Design of Scheduling Algorithms for End-to-End Backlog Minimization in Wireless Multi-hop Networks under K-hop Interference Models

Shizhen Zhao, *Student Member, IEEE,* and Xiaojun Lin, *Senior Member, IEEE*

Abstract—**In this paper, we study the problem of link scheduling for multi-hop wireless networks with per-flow delay constraints under the K-hop interference model. Specifically, we are interested in algorithms that maximize the asymptotic decay-rate of the probability with which the maximum end-to-end backlog among all flows exceeds a threshold, as the threshold becomes large. We provide both positive and negative results in this direction. By minimizing the drift of the maximum end-to-end backlog in the converge-cast on a tree, we design an algorithm, Largest-Weight-First (LWF), that achieves the optimal asymptotic decay-rate for the overflow probability of the maximum end-to-end backlog as the threshold becomes large. However, such a drift minimization algorithm may not exist for general networks. We provide an example in which no algorithm can minimize the drift of the maximum end-to-end backlog. Finally, we simulate the LWF algorithm together with a well known algorithm (the back-pressure algorithm) and a large-deviations optimal algorithm in terms of the sum-queue (the P-TREE algorithm) in converge-cast networks. Our simulation shows that our algorithm performs significantly better not only in terms of asymptotic decay-rate, but also in terms of the actual overflow probability.**

I. Introduction

In this paper, we study the link-scheduling problem for multi-hop wireless networks to improve the end-to-end delay performance. In such networks, each flow transmits packets from source to destination in a multi-hop fashion. We assume that each flow has a fixed route from the source to destination. Due to the shared nature of the wireless medium, there are both intra-flow interference (i.e., links at different hops of a flow may interfere with each other) and inter-flow interference (i.e., links of different flows may interfere with each other) in the system. Further, packets from multiple flows may compete for the service at a common link. Hence, it becomes a challenging problem to determine how link transmissions and packet transmissions should be scheduled in order to minimize some notion of end-to-end delay, subject to interference constraints.

When only the stability of a network is concerned, it is well-known that the back-pressure algorithm [2] is throughput-optimal, i.e., it can stabilize a multi-hop wireless system under the largest set of offered-load vectors. However, stability only means that the backlog in the system remains finite, and is inadequate for many delay-sensitive applications that require stringent end-to-end delay guarantees. In fact, it has been observed that the back-pressure algorithm can have very poor end-to-end delay performance [3]. On the other hand, It is predicted that delay-sensitive traffic, e.g., video streaming, will consume two-thirds of the overall mobile data traffic by 2016 [4]. Therefore, it is important to study finer performance metrics for the end-to-end delay and design scheduling algorithms that are optimal for these metrics.

Characterizing the end-to-end delay in multi-hop wireless networks is usually a very challenging problem. Although there has been a considerable body of work on the delay performance of single-hop wireless networks (e.g., [5][6]), results on the end-to-end delay performance for multi-hop wireless networks are more limited [7]. In multi-hop networks, the packet arrivals at downstream links are the departures from upstream links. Hence, the statistics of the arrival processes at downstream links are often unknown beforehand. As a result, the end-to-end delay performance in multi-hop systems is much more difficult to characterize and optimize than single-hop systems.

As is typical in the literature [5][8], we use the end-to-end backlog of a flow, i.e., the total backlog of the flow over all links along its path, as a measure for its end-to-end delay performance. There are different approaches for analysing the end-to-end backlog. For a tandem network, the work in [9] provides a scheduling algorithm that is sample-path optimal for minimizing the total end-to-end backlog. Such sample-path optimality results attain the strongest sense of optimality. However, they are also the most demanding, and hence their applicability is the most restrictive. In fact, for more general topologies, e.g., a tree topology with converge-cast, it can be shown that there may not even exist sample-path optimal algorithms [10]. Other approaches study weaker notions of optimality. For example, [11][12] study the expected value of the total end-to-end backlog among all flows. These studies typically provide upper- and lower-bounds of the expected total end-to-end backlog. However, it is usually difficult to identify which algorithm attains the smallest expected end-to-end backlog. Another approach is to use large-deviations theory to characterize the probability that some function of the end-to-end backlog exceeds a large threshold. Our prior work [7] has applied such a large-deviations approach to a tree-network with converge-cast for minimizing the large-

An earlier version of this paper [1] has appeared in the 31st IEEE International Conference on Computer Communications (IEEE INFOCOM), 2012. This work was partially supported by NSF through grants CNS-0643145, CNS-0721477 and CNS-0721484, by a grant from the Purdue Research Foundation, and by the Bilsland Dissertation Fellowship at Purdue University.

Shizhen Zhao and X. Lin are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette (email: {zhao147, linx}@purdue.edu).

deviations decay rate of the probability that the sum of the end-to-end backlog over all flows exceeds a large threshold. However, similar to [9][11][12], [7] only considers the total end-to-end backlog among all flows. Such a metric fails to account for the fairness issue across different flows, which may lead to network starvation when the channel condition of one flow is significantly worse than that of other flows.

In this paper, we are interested in the *maximum* end-to-end backlog among all flows. Note that in many scenarios the maximum end-to-end backlog among all flows is more useful than the total sum. For example, consider again a converge-cast on a tree where all nodes send packets to the root of the tree. This setting can model, e.g., a video surveillance system where all cameras send captured video to a central monitoring station, or a cellular uplink with multi-hop relays where all mobiles send data to the base-station in a multi-hop fashion. Suppose that we need to ensure that the delay of every video frame or every packet is small. In this case, it is more important to minimize the *maximum* end-to-end backlog among all flows, rather than the sum of the backlog of all flows[1]. Unfortunately, the maximum end-to-end backlog turns out to be more challenging to minimize than the sum of the end-to-end backlog among all flows. As we observe in [9], in order to minimize the end-to-end backlog for a single flow, one needs to give priority to links closer to the destination. On the other hand, to minimize the maximum end-to-end backlog among multiple flows, one needs to give priority to flows with the largest end-to-end backlog. The key difficulty is that these two priorities are not always consistent with each other. In another prior work [8], these two priorities are asymptotically attained by a family of $\alpha\beta$-scheduling algorithms, which are then shown to be optimal in the large-deviations sense as one asymptotically takes the parameters $\alpha \rightarrow 0$ and $\beta \rightarrow \infty$. However, in practice the algorithms approaching the limit can have large overflow probabilities when the overflow threshold of interest is small. Hence, it remains a challenge to find algorithms for minimizing the maximum end-to-end backlog among multiple flows that are not only large-deviations optimal, but also have good performance at small overflow thresholds of interest.

In this paper, we provide both positive and negative results for this open problem. On the positive side, we first focus on a converge-cast on a tree topology under the K-hop interference model. We provide a new large-deviations optimal algorithm, called Largest-Weight-First (LWF), for minimizing the maximum end-to-end backlog among all flows. Our proposed algorithm intelligently combines together the two priorities that we discussed above in a non-asymptotic manner. As a result, the LWF algorithm is not only large-deviations optimal, but also significantly reduces the overflow probability at small thresholds of interest. To the best of our knowledge, this is the first such optimal algorithm for minimizing the maximum end-to-end backlog among all flows in a converge-cast scenario.

The optimality of LWF is shown based on one of our earlier results in [14] that, under suitable conditions, an algorithm that minimizes the drift of a Lyapunov function at every time in every fluid sample path (FSP) is also large-deviations optimal for minimizing the probability that the Lyapunov function value exceeds a large threshold. Taking the maximum end-to-end backlog among all flows as the Lyapunov function, we show that the LWF algorithm minimizes its drift for converge-cast on a tree at every time in every FSP. Therefore, it must be large-deviations optimal. Given that no sample-path optimal algorithms exist for a general converge-cast setting, our result illustrates the potential power and flexibility with the drift minimizing criterion. (We caution, however, that it is not straightforward at all to find such drift minimizing algorithms and to verify the drift minimizing property, which we will elaborate below and in Section III.)

Given the success of the LWF algorithm in converge-cast scenarios, we are then interested in developing similar optimal algorithms for more general network topology settings. In particular, we would like to know whether we can find algorithms (similar to LWF) that minimize the drift of the maximum end-to-end backlog in more general settings. It is along this direction that we report two negative, but important, results. First, we show that the LWF algorithm is not drift-minimizing for a tree topology that is not converge-cast. In fact, we show that it is not even throughput optimal in such a setting. Second (and perhaps more importantly), we find a tree topology that is not a converge-cast, where we show that there exists no algorithm that can possibly be drift-minimizing at every time in every FSP. Hence, these results indicate that, while the drift-minimizing criterion is more flexible and powerful than the sample-path optimality criterion, it also has its own limitations. For more general settings, we may have to search for other criteria for designing large-deviations optimal scheduling algorithms.

II. System Model

A. Model

We model the topology of a wireless multi-hop network by a graph $G = G(V, \mathcal{E})$, where V is the set of nodes and $\mathcal{E}$ is the set of directed edges that represent physical links. There are F single-path flows in the network. Each flow $f = 1, 2, ..., F$ corresponds to a fixed path, which consists of a subset of the physical links that it traverses. However, for ease of exposition, we will also view a path as a subgraph of G, consisting of the nodes and the edges belonging to the path. Let the subgraph be denoted by P^f. Let the path collection $\mathcal{F} = \{P^f | f = 1, 2, ..., F\}$ be the collection of all P^f. A network $(G, \mathcal{F})$ is defined as a network topology G equipped with a path collection $\mathcal{F}$.

The transmission on one link may interfere with other links. We consider the following K-hop interference model: two links interfere with each other if and only if it takes no greater than K hops to jump from one link to another link [15]. In other words, there exists a path between them such that the number of intermediate links is no greater than $K - 1$. Let R_l be the capacity of physical link l, i.e., R_l is the maximum amount of data that can be transmitted over link l in a time slot if l is active and no other interfering links are active.

[1] For the same reason, our objective is also different from the studies that minimize the draining time, e.g., [13], which consider all flows' performance as a whole.

Consider the network $(G, \mathcal{F})$. For each flow f, we label the link at the i-th hop from the destination node as $l_i^f, i = 1, 2, ..., n^f$, where n^f is the total number of links on its path P^f. Note that, with such a convention, l_1^f is the link next to the destination node, while $l_{n^f}^f$ is the link next to the source node. Packets of flow f arrive at link $l_{n^f}^f$, travel multiple hops to l_1^f, and then depart from the system. We use $P^f(l_i^f)$ to denote the sub-path starting from link l_i^f and ending at its destination node. Obviously, $P^f(l_{n^f}^f) = P^f$. Note that, it is possible to assign multiple labels l_i^f to the same physical link if the link is used by more than one flow. In the rest of the paper, it is more convenient to view these multiple labels on the same physical link as separate logical links, one for each flow. For flow f, the logical link at the i-th hop from its destination is l_i^f. The capacity of each logical link is the same as its underlying physical link. Two logical links interfere with each other if and only if they share the same physical link or their corresponding physical links interfere. We denote by $\mathcal{E}^L$ (the superscript "L" stands for "logical links") the set of logical links in network $(G, \mathcal{F})$. For each logical link l_i^f, we use $\mathcal{I}(l_i^f)$ to denote the set of all logical links interfering with l_i^f. In the rest of the paper, when we refer to links with labels, we will mean logical links.

Let $A^f(t)$ be the amount of data offered to the source node of flow f at time t. We assume that $A^f(t) \leq M$ for any t, f. Moreover, we assume that $A^f(t)$ is *i.i.d.*[2] across time. Let $\lambda^f = E[A^f(t)]$ denote the arrival rate, and $\vec{\lambda} = [\lambda^f, f = 1, 2, ..., F]$. The capacity region of $(G, \mathcal{F})$ is defined as the largest set of offered load $\vec{\lambda}$ that the network can support. We are interested in the case when the arrival rate vector $\vec{\lambda}$ is strictly inside the capacity region. We use $X_i^f(t)$ to denote the queue length of flow f at link l_i^f. Let $E_i^f(t)$ denote the actual amount of data transmitted over link l_i^f at time t. Obviously, $E_i^f(t) \leq \min\{X_i^f(t), R_{l_i^f}\}$. The queue length at link l_i^f is then updated in the following way.

$$X_i^f(t{+}1) = \begin{cases} X_i^f(t) + E_{i+1}^f(t) - E_i^f(t), \text{if}\, i = 1, 2, ..., n^f - 1, \\ X_i^f(t) + A^f(t) - E_i^f(t), \quad \text{if}\, i = n^f. \end{cases} \tag{1}$$

Then, the total end-to-end backlog of flow f, $X^f(t) = \sum_{i=1}^{n^f} X_i^f(t)$, is governed by

$$X^f(t+1) = X^f(t) + A^f(t) - E_1^f(t). \tag{2}$$

B. Design Objective

In this paper, we are interested in designing a scheduling algorithm to minimize the maximum end-to-end backlog among all flows. Specifically, given an algorithm that can stabilize the system (and thus a steady-state distribution exists), we consider the steady state probability that the maximum end-to-end backlog[3] over all flows exceeds a threshold B, i.e.,

$$\mathbb{P}\left[\max_f \left\{X^f(+\infty)\right\} \geq B\right]. \tag{3}$$

If the scheduling algorithm cannot stabilize the system (and thus the queue length will grow unboundedly), we take

$$\mathbb{P}\left[\max_f \left\{X^f(+\infty)\right\} \geq B\right] = 1.$$

We then want to minimize the quantity in (3). This objective is extremely useful in practice. For example, for multiple data flows driven by real-time applications, such as video streaming, visual web conference, etc., it is very important to keep the end-to-end backlog among all flows small to meet the stringent delay constraints. Unfortunately, this quantity is in general mathematically intractable. Instead, since the above probability is small, we can focus on its asymptotic decay rate when B becomes large. Specifically, we can define the following two quantities:

$$-I \triangleq \liminf_{B\to\infty} \frac{1}{B} \log \left(\mathbb{P}\left[\max_f \left\{X^f(+\infty)\right\} \geq B\right]\right) \tag{4}$$

$$-J \triangleq \limsup_{B\to\infty} \frac{1}{B} \log \left(\mathbb{P}\left[\max_f \left\{X^f(+\infty)\right\} \geq B\right]\right) \tag{5}$$

Then, for large B, we have $e^{-IB+o(B)} \leq \mathbb{P}\left[\max_f \left\{X^f(+\infty)\right\} \geq B\right] \leq e^{-JB+o(B)}$. Therefore, we can instead focus on finding a scheduling algorithm that maximizes I and J.

III. End-to-End Backlog Minimization in Converge-cast Networks

In this section, we are interested in a converge-cast scenario on a tree topology. For a converge-cast, the root O of the tree is the destination node of all flows in the network. In order to emphasize that we are dealing with converge-cast in a tree topology, we add subscript "T" to the notations G, $\mathcal{F}$ and $\mathcal{E}^L$. Specifically, we use $G_T(V_T, \mathcal{E}_T)$ to denote the graph, $\mathcal{F}_T$ to denote the path collection, and $\mathcal{E}_T^L$ to denote the set of all logical links, respectively, for the tree topology. We will propose a large-deviations optimal algorithm for such a converge-cast network $(G_T, \mathcal{F}_T)$ to minimize the maximum end-to-end backlog among all flows.

A. Largest-Weight-First Algorithm

We first propose the Largest-Weight-First (LWF) algorithm for the converge-cast network $(G_T, \mathcal{F}_T)$. For each logical

[2] This assumption can be relaxed. The key requirement for the results in this paper to hold is that the arrival processes satisfy a sample-path large-deviations principle. As an example, finite-state irreducible Markov chains satisfy the sample-path large deviations principle. See Section II-E in [14] for related discussion.

[3] The results of this work can also be generalized to study the case with the maximum *weighted* end-to-end backlog, i.e., to minimize the overflow probability $\mathbb{P}[\max_f \{c_f X^f(+\infty)\} \geq B]$, where c_f is a positive weight chosen for each flow f. Specifically, if we change the link weights (see Eqn. (6)) of the LWF algorithm to $q_i^f(t) = c_f \eta_i^f(t) \sum_{j=i}^{n^f} X_j^f(t)$, the LWF algorithm will be drift-minimizing for the maximum weighted end-to-end backlog. Note that when $c_f = 1/\lambda_f$, the overflow event $\max_f \{c_f X^f(+\infty)\} \geq B$ may have a closer relationship to the delay performance of flow f [16].

link $l_i^f \in \mathcal{E}_T^L$, we define the usage efficiency $\eta_i^f(t) \triangleq \min\left\{\frac{X_i^f(t)}{R_{l_i^f}}, 1\right\}$. We assign a weight for link l_i^f as follows:

$$q_i^f(t) = \eta_i^f(t) \sum_{j=i}^{n^f} X_j^f(t). \tag{6}$$

Note that by multiplying the usage efficiency, the above definition reduces the weight for those links whose backlog is less than their capacity. We then use a greedy algorithm to compute a feasible schedule. Specifically, we first schedule the link with the largest weight, and delete this link and all the other links that interfere with this link. Then, we schedule the link with the largest weight from the remaining links, and again delete this link and all the other links that interfere with this link. We repeat this process until there are no remaining links.

We briefly describe the intuition behind this algorithm. Recall that link l_1^f is closest to the destination O, and $l_{n^f}^f$ is closest to the source of flow f. In Eqn. (6), a link closer to the destination tends to have a larger weight than another upstream link from the same flow. Further, links from flows with larger end-to-end backlog will also tend to have larger weights. Hence, the weight definition (especially the term "$\sum_{j=i}^{n^f} X_i^f(t)$") in the LWF algorithm can be viewed as a way to combine two priorities, i.e., giving priority to those flows with larger end-to-end backlog and to those links closer to the destination. Note that giving priority to the flow with the largest end-to-end backlog is natural since we want to minimize the maximum end-to-end backlog among all flows. Further, the idea that giving priority to links closer to destination can help to drain packets faster has been reported for a simple linear topology with only one flow [9]. Our proposed algorithm can be viewed as a generalization to the multi-flow setting, which combines these two ideas together. However, we note that when there are two different priorities, they may not always be compatible with each other. Hence, it is not at all clear why (6) is the right way to combine these two priorities. The proof of optimality presented next also requires new techniques and follows very different lines as that in the prior work [7][9].

We note that the weight definition in (6) also contains the usage efficiency "$\eta_i^f(t)$". This usage efficiency $\eta_i^f(t)$ is essentially the fraction of capacity that can be utilized whenever the link l_i^f is scheduled. $\eta_i^f(t) = 1$ means that there is no capacity wasted for the link l_i^f if this link is scheduled. Intuitively, if a link has very low usage efficiency, we should not give this link a high priority no matter how close this link is to the destination. As readers will see in Lemma 6 and Section III-C4, this usage efficiency is critical to ensure the optimality of the LWF algorithm.

The LWF algorithm is formally described in Algorithm 1. In this algorithm, we use $\vec{\gamma(t)} = [\gamma_l(t), l \in \mathcal{E}_T^L]$ to denote the scheduling decision at time t. $\gamma_l(t) = 1$ if the logical link l is scheduled, and $\gamma_l(t) = 0$ otherwise.

To implement the LWF algorithm, we need the queue-length information for all links. Hence, the LWF algorithm can best be viewed as a centralized algorithm that uses a separate control channel to gather queue-length information, compute the schedule, and then distribute the decision back to each link. This is a reasonable setting when such a central station and control channel is available, e.g., in a cellular system with multi-hop relays. In our analysis, we assume that the LWF algorithm has the up-to-date queue-length information for every link in every time slot. This assumption simplifies the analysis, and the results can be viewed as an upper bound for other more practical settings. Further, as readers will see in the simulation section, even when such an assumption is relaxed, the LWF algorithm still performs very well in practice.

```
At time slot $t$, calculate the weight for each logical link $l$.
Let $\mathcal{E}_r = \mathcal{E}_T^L, \vec{\gamma(t)} = \vec{0}$.
while $\mathcal{E}_r \neq \emptyset$ do
    Find a logical link $l \in \mathcal{E}_r$ with the largest weight.
    Set $\gamma_l(t) = 1$, $\mathcal{E}_r = \mathcal{E}_r \setminus (\mathcal{I}(l) \cup \{l\})$.
end
The scheduling decision is given by $\vec{\gamma(t)}$.
```

Algorithm 1: Largest-Weight-First(LWF) algorithm

B. Mathematical Preliminaries

In this section, we will be mainly interested in the tree network $(G_T, \mathcal{F}_T)$ and its subnetworks. Consider the following subnetwork of $(G_T, \mathcal{F}_T)$ given by a subset of flows $\mathcal{A} \subseteq \{1, 2, ..., F\}$ and a vector $z_{\mathcal{A}} = [z^f, f \in \mathcal{A}]$, where each z^f is an integer between 1 and n^f. We denote such a subnetwork by $\mathcal{N}(\mathcal{A}, z_{\mathcal{A}})$. For each flow $f \in \mathcal{A}$, the route in the subnetwork is given by a subpath of P^f starting from the z^f-th hop and ending at the root O, i.e., $P^f(l_{z^f}^f)$. The subnetwork path collection is then given by $\{P^f(l_{z^f}^f) | f \in \mathcal{A}\}$. The subnetwork topology is a graph consisting of all links traversed by at least one path $P^f(l_{z^f}^f), f \in \mathcal{A}$, and all vertices adjacent to these links, which can be represented by $\bigcup_{f \in \mathcal{A}} P^f(l_{z^f}^f)$. Further, we use $\mathcal{E}_T^L(\mathcal{A}, z_{\mathcal{A}})$ to denote the set of all logical links of the subnetwork $\mathcal{N}(\mathcal{A}, z_{\mathcal{A}})$.

1) Fluid Sample Paths: Fix T. For any B and any realization ω of the arrival process, define the following scaled quantities in the time interval $[0, T]$ as

$$\begin{aligned}
a^{f,B}(\omega, t) &= \frac{1}{B}\sum_{\tau=0}^{Bt-1} A^f(\omega, \tau), \\
x_i^{f,B}(\omega, t) &= \frac{1}{B} X_i^f(\omega, Bt), \\
x^{f,B}(\omega, t) &= \frac{1}{B} X^f(\omega, Bt), \\
e_i^{f,B}(\omega, t) &= \frac{1}{B}\sum_{\tau=0}^{Bt-1} E_i^f(\omega, \tau),
\end{aligned} \tag{7}$$

for $t = \frac{m}{B}, m = 0, 1, ..., BT$, and by linear interpolation[4], otherwise. For ease of exposition, we denote $Z^B(\omega, t) =$

[4] Here, by linear interpolation, we mean that between $t_1 = m/B$ and $t_2 = (m+1)/B$, we set $a^{f,B}(\omega, t) = a^{f,B}(\omega, t_1) + \frac{a^{f,B}(\omega,t_2) - a^{f,B}(\omega,t_1)}{t_2 - t_1}(t - t_1)$ for all $t \in (t_1, t_2)$. The linear interpolation for other quantities is defined analogously.

$[a^{f,B}(\omega,t), x_i^{f,B}(\omega,t), x^{f,B}(\omega,t), e_i^{f,B}(\omega,t), f = 1,2,...,F, i = 1,2,...,n^f]$.

Due to the assumption of bounded arrivals and departures, $Z^B(\omega,t)$ is Lipschitz continuous with respect to $t \in [0,T]$. For the fixed T, we take any sequence $[Z^{B_1}(\omega_1,t), Z^{B_2}(\omega_2,t), ...]$ of such scaled processes as $B_i \to \infty$. There must exist a subsequence $[Z^{B_{k_1}}(\omega_{k_1},t), Z^{B_{k_2}}(\omega_{k_2},t), ...]$ that converges uniformly to a limit $Z(t) = [a^f(t), x_i^f(t), x^f(t), e_i^f(t)]$ over the compact interval $[0,T]$. Any such limit $Z(t)$ is called a fluid sample path (FSP) [14][17]. Note that in general, there may exist more than one FSPs out of the same (possibly non-convergent) sequence of scaled processes $Z^{B_i}(\omega_i,t)$.

Remark 1. We note that the notion of convergence in the definition of FSP is not related to the probability measure, and hence is different from other probabilistic notion of convergence, e.g., fluid limit [18]. Specifically, the sequence of realizations $[\omega_1, \omega_2, ...]$ can be chosen arbitrarily, and they may not correspond to those with high probability. In contrast, fluid limits are limiting processes of almost-surely (i.e., with probability 1) convergent sequences. Intuitively, the fluid limit captures the mean behavior of the system, e.g., the fluid limit of the arrival process $a^f(t)$ will be $\lambda_f t$. In contrast, in an FSP, the arrival process $a^f(t)$ will typically deviate from the mean arrival process $\lambda_f t$. Indeed, as is typical in large-deviations theory, we are interested in "rare" events that are away from the mean value [14]. The relationship between an FSP and the original probability space is through the notion of large-deviations rate-function discussed below.

In order to study the decay-rate of the steady-state probability (3), we can instead study the decay rate of the probability $P\left[\left\{\omega : \max_{f=1,...,F}\{x^{f,B}(\omega,T)\} \geq 1\right\}\right]$ as $B \to \infty$ [14]. Since the arrival process is *i.i.d.*, the scaled arrival process $a^B(\omega,t)$ satisfies a large deviation principle with some rate function $I_a^T(\cdot)$ (Chapter 1.2 in [19]). This means that, for any set Γ of arrival sample paths, the probability that $a^B(\omega,t) = [a^{f,B}(\omega,t), f = 1,...,F]$ falls into Γ satisfies $\lim_{B\to\infty} \frac{1}{B}\mathbb{P}(\omega | a^B(\omega,t) \in \Gamma) = -\inf_{a'\in\Gamma} I_a^T(a')$. In the typical large-deviations literature, if we can additionally verify that the mapping from $a^B(\omega,t)$ to $x^{f,B}(\omega,t)$ is continuous under a given scheduling algorithm, we can then apply the contraction principle [19] and obtain the exact decay-rate of the overflow probability by finding the "most likely path to overflow". Note that when this approach works, one can potentially characterize the decay-rate for arbitrary overflow events and arbitrary scheduling algorithms. However, there are significant difficulties in applying this approach in our work with multiple multi-hop flows. First, it is usually a non-trivial task to verify the continuity of the mapping from $a^B(\omega,t)$ to $x^{f,B}(\omega,t)$. Second, finding such a "most likely path to overflow" involves solving a high-complexity multi-dimensional calculus-of-variations problem, which is also challenging. Further, note that we are interested in the optimal algorithm with the largest decay-rate. Even if we can find the decay-rate for each scheduling algorithm, it is unclear how to search for the algorithm with the largest decay-rate. In this paper, we use a different approach, first proposed in [14], to circumvent the above difficulties. The result of [14] proposes a drift minimizing criterion, which can be used to directly find an algorithm that can attain the largest possible decay-rate. We will go into details in Section III-C.

Since the convergence to an FSP is uniform, the FSP $Z(t)$ is also Lipschitz-continuous, and therefore, its derivative exists almost everywhere (a.e.) over $[0,T]$. Define the Lyapunov function $V(x(t)) \triangleq \max_f x^f(t)$. It is easy to check that $V(x(t))$ is also Lipschitz-continuous, and thus is differentiable a.e. with respect to t. Denote by $\mathcal{T}$ the set of all time instances where the FSP or the Lyapunov function $V(x(t))$ is not differentiable with respect to t. Then $\mathcal{T}$ is of measure 0. In the rest of this paper, we will restrict our analysis to those $t \notin \mathcal{T}$, and we call such a time instant a regular time.

At any regular time t, we define $\alpha^f(t) = \frac{d}{dt}a^f(t)$ and $\mu_i^f(t) = \frac{d}{dt}e_i^f(t)$. (We note that $\alpha^f(t)$ may not be equal to λ^f because the arrival rate of an FSP may deviate from the mean arrival rate λ^f.) Then, we can derive the following equations for an FSP from equation (1) and (2) (refer to [7] for details):

$$\frac{d}{dt}x_i^f(t) = \begin{cases} \mu_{i+1}^f(t) - \mu_i^f(t), & \text{if } i = 1,2,...,n^f - 1, \\ \alpha^f(t) - \mu_i^f(t), & \text{if } i = n^f. \end{cases} \tag{8}$$

$$\frac{d}{dt}x^f(t) = \alpha^f(t) - \mu_1^f(t). \tag{9}$$

Eqn. (8) and (9) can be interpreted as the limits of (1) and (2) as $B \to \infty$.

As for $V(x(t))$, define $\mathcal{M}(t) = \{f | x^f(t) = \max_{f'}\{x^{f'}(t)\}\}$ as the set of flows that have the largest end-to-end backlog in the FSP at time t. Then,

$$\frac{d}{dt}V(x(t)) = \max_{f\in\mathcal{M}(t)}\{\alpha^f(t) - \mu_1^f(t)\}. \tag{10}$$

In addition, we have the following lemma that imposes additional constraints for $\mu_i^f(t)$.

Lemma 1. *(Proposition 1 in [7]) Under any algorithm, any FSP $(a^f(t), x_i^f(t), x^f(t), e_i^f(t))$ must satisfy the following flow constraint for each flow f:*

$$\mu_i^f(t) \leq \begin{cases} \mu_{i+1}^f(t), & \textit{if } i = 1,2,...,n^f - 1 \textit{ and } x_i^f(t) = 0, \\ \alpha^f(t), & \textit{if } i = n^f \textit{ and } x_i^f(t) = 0. \end{cases} \tag{11}$$

Further, suppose that $l_{i_1}^{f_1}, l_{i_2}^{f_2}, ..., l_{i_y}^{f_y}$ are links that interfere with each other. Then, any FSP must also satisfy the following link constraints:

$$\sum_{k=1}^{y} \frac{\mu_{i_k}^{f_k}(t)}{R_{l_{i_k}^{f_k}}} \leq 1 \tag{12}$$

$$\mu_i^f(t) \geq 0, \textit{ for all } i, f. \tag{13}$$

Lemma 1 states that regardless of the scheduling algorithm, the service rates $\mu_i^f(t)$ must satisfy the constraints (11)-(13). Specifically, the first part of (11) states that, when the backlog $x_i^f(t)$ is 0, we need the upstream link l_{i+1}^f to serve at least as many packets as the downstream link l_i^f. A similar interpretation holds for the second part of (11). In (12), $\frac{\mu_i^f(t)}{R_{l_i^f}}$ can be viewed as the fraction of time that link l_i^f is activated. The sum of the fraction of time must be no greater than 1 for mutually interfering links.

C. Optimality of the LWF Algorithm

We will use the techniques of [14] to prove that the LWF algorithm achieves the largest asymptotic decay-rate of the maximum end-to-end backlog overflow probability. We would like to prove the following result.

Theorem 2. *The LWF algorithm achieves the optimal decay-rate for the maximum end-to-end backlog overflow probability, i.e., for any scheduling algorithm π, we have*

$$\begin{aligned}&\limsup_{B\to\infty}\frac{1}{B}\log\left(\mathbb{P}^{LWF}\left[\max_{f=1,2,\ldots,F}\{X^f(+\infty)\}\geq B\right]\right)\\ \leq\ &\liminf_{B\to\infty}\frac{1}{B}\log\left(\mathbb{P}^{\pi}\left[\max_{f=1,2,\ldots,F}\{X^f(+\infty)\}\geq B\right]\right),\end{aligned}\tag{14}$$

where $\mathbb{P}^{LWF}$ and $\mathbb{P}^{\pi}$ denote the stationary distributions under algorithms LWF and π, respectively.

To prove Theorem 2, we use the result of Theorem 8 from [14]. Consider a scheduling algorithm π_0. Suppose that the Lyapunov function $V(x(t)) = \max_f\{x^f(t)\}$ satisfies Assumptions 1-6 of [14] under π_0. In the rest of this paper, we will simply say that such an algorithm π_0 satisfies Assumptions 1-6. Below, we restate Theorem 8 from [14] for reference.

Theorem 3. *Suppose that an algorithm π_0 satisfies Assumptions 1,2,3,4,5 and 6. Then, the algorithm π_0 attains the optimal decay-rate in the sense of (14).*

Thus, to prove Theorem 2, we only need to justify Assumptions 1-6 of [14] for the LWF algorithm. In our technical report [20], we list Assumptions 1-3, 5 and 6 (except Assumption 4), and verify that these assumptions hold for the LWF algorithm. Hence, here we only focus on Assumption 4.

Before stating Assumption 4, we first introduce the concept of the "possible drift" of the Lyapunov function $V(x(t)) = \max_f x^f(t)$ under a given scheduling vector $\hat{\mu}$. Note that given an FSP $Z(t) = (a^f(t), x_i^f(t), x^f(t), e_i^f(t))$ and a specific regular time t, the drift of $V(x(t))$ at time t following the FSP is simply given by $\frac{d}{dt}V(x(t))$ in (10), where the corresponding decision of the scheduling algorithm produces $\mu(t) = \frac{d}{dt}e(t)$. However, if we were to choose another scheduling decision vector $\hat{\mu} \neq \frac{d}{dt}e(t)$, the *possible drift* of $V(x(t))$ would be different. Specifically, starting from $x(t)$, consider what happens when a feasible scheduling decision vector[5] $\hat{\mu} = [\hat{\mu}_i^f]$ is applied over the time-interval $[t, t+\delta]$, assuming that the same scaling as in (7) is taken. Note that whenever $x_i^f(t) = 0$, the number of packets served by a downstream link l_i^f cannot exceed the number of packets served by its upstream link l_{i+1}^f. Thus, let $\tilde{\mu} = [\tilde{\mu}_i^f]$ be determined from both $x(t)$ and $\hat{\mu}$ as follows

$$\tilde{\mu}_i^f = \begin{cases}\min\{\hat{\mu}_i^f, \tilde{\mu}_{i+1}^f\}, & \text{if } i < n^f \text{ and } x_i^f(t) = 0,\\ \min\{\hat{\mu}_i^f, \alpha^f(t)\}, & \text{if } i = n^f \text{ and } x_i^f(t) = 0,\\ \hat{\mu}_i^f, & \text{otherwise.}\end{cases}\tag{15}$$

Then, it is easy to check that the trajectory of $x_i^f(t)$ would follow $x_i^f(t+\delta;\hat{\mu}) = x_i^f(t) + (\tilde{\mu}_{i+1}^f - \tilde{\mu}_i^f)\delta$ for sufficiently small δ, where we have used $x_i^f(t+\delta;\hat{\mu})$ to denote the queue evolution in the immediate future of t under the scheduling decision vector $\hat{\mu}$. Then, the possible drift at time-slot t under the scheduling decision vector $\hat{\mu}$ would be given by

$$\lim_{\delta\to 0^+}\frac{V(x(t+\delta;\hat{\mu})) - V(x(t))}{\delta} = \max_{f\in\mathcal{M}(t)}\{\alpha^f(t) - \tilde{\mu}_1^f\}.\tag{16}$$

Thus, we refer to the right hand side of (16) as the possible drift under the scheduling vector $\hat{\mu}$. Note that, if the scheduling decision vector $\hat{\mu}_i^f$ is chosen to be exactly $\frac{d}{dt}e_i^f(t) = \mu_i^f(t)$, then the possible drift would be identical to the real drift (see (10)) along the original FSP. Now, we are ready to present Assumption 4.

Assumption 4. *(Drift Minimization Assumption) For any FSP $Z(t) = (a^f(t), x_i^f(t), x^f(t), e_i^f(t))$ under the algorithm π_0, the following holds for all regular times t.*

$$\begin{aligned}\frac{d}{dt}V(x(t)) \quad &= \quad \min_{\hat{\mu}}\max_{f\in\mathcal{M}(t)}\{\alpha^f(t) - \tilde{\mu}_1^f\} \qquad (17)\\ \textit{subject to} \quad &\quad \hat{\mu} \textit{ is a feasible scheduling decision vector,}\\ &\quad \hat{\mu} \textit{ and } \tilde{\mu} \textit{ satisfy } (15).\end{aligned}$$

Note that the right hand side of (17) can be viewed as the minimum possible drift over all feasible $\hat{\mu}$. Thus, Assumption 4 states that the decisions under the algorithm π_0 minimize the drift of the Lyapunov function $V(x(t))$ at every time t for every FSP.

Theorem 4. *The LWF algorithm satisfies the drift minimization assumption.*

Since we have verified that the LWF algorithm satisfies Assumptions 1,2,3,5 and 6 (see our technical report [20]), Theorem 2 follows directly from Theorem 3 and Theorem 4. We next focus on proving Theorem 4. Our strategy is as follows. We first derive a lower bound to the optimization problem on the right-hand-side of (17). We then show that the drift of the LWF algorithm achieves this lower bound.

1) Lower Bound: We now derive an lower bound for the optimal value of the optimization problem in (17). First, from our construction (15), it is easy to verify that any $\tilde{\mu}$ that satisfies the constraints in (17) must also satisfy the following properties (see details in [20]):

$$\sum_{k=1}^{y}\frac{\tilde{\mu}_{i_k}^{f_k}}{R_{l_{i_k}^{f_k}}} \leq 1,\tag{18}$$

for all $l_{i_1}^{f_1}, \ldots, l_{i_y}^{f_y}$ that interfere with each other; and

$$\tilde{\mu}_i^f \geq 0, \quad \text{for all } \ i, f;\tag{19}$$

$$\tilde{\mu}_i^f \leq \begin{cases}\tilde{\mu}_{i+1}^f, & \text{if } i < n^f \text{ and } x_i^f(t) = \frac{d}{dt}x_i^f(t) = 0,\\ \alpha^f(t), & \text{if } i = n^f \text{ and } x_i^f(t) = \frac{d}{dt}x_i^f(t) = 0,\end{cases}\tag{20}$$

where $\frac{d}{dt}x_i^f(t)$ is computed based on (8) using the scheduling vector $\mu(t)$ (rather than $\tilde{\mu}$).

For the given FSP $Z(t) = (a^f(t), x_i^f(t), x^f(t), e_i^f(t))$ in

[5] A scheduling decision vector $\hat{\mu} = [\hat{\mu}_i^f]$ is said to be feasible if there exists a sequence of schedules such that, when the scaling in (7) is taken, the limiting service rate of link l_i^f at time t is equal to $\hat{\mu}_i^f$.

Assumption 4, we define another set $\mathcal{M}_0(t) \subseteq \mathcal{M}(t)$ as follows. Recall that $\mu_i^f(t) = \frac{d}{dt}e_i^f(t)$ is the service rate of the LWF algorithm given the system input $a^f(t)$ and the system state $x_i^f(t)$ and $x^f(t)$. Denote by $\mathcal{M}_0(t)$ the set of flows in $\mathcal{M}(t)$ that have the maximum growth rate, i.e.,

$$\mathcal{M}_0(t) \quad = \quad \left\{ f \in \mathcal{M}(t) : \frac{d}{dt}x^f(t) = \max_{f' \in \mathcal{M}(t)} \left\{ \frac{d}{dt}x^{f'}(t) \right\} \right\},$$

where $\frac{d}{dt}x^f(t) = \alpha^f(t) - \mu_1^f(t)$. Now, define

$$D^*(t) \quad = \quad \min_{\tilde{\mu}} \max_{f \in \mathcal{M}_0(t)} \{\alpha^f(t) - \tilde{\mu}_1^f\} \qquad (21)$$

$$\text{subject to} \qquad \tilde{\mu}_i^f \text{ satisfies (18)(19) and (20).}$$

Clearly, $D^*(t)$ is a lower bound to the minimum drift on the right-hand-side of (17) since in $D^*(t)$ we take the maximum over a smaller set of f's and the constraints for $\tilde{\mu}$ in (21) is a necessary condition for the constraints in (17). (See details in our technical report [20].)

2) Achievable Drift of the LWF Algorithm: Based on the definition of $\mathcal{M}_0(t)$, we can easily compute the drift $D_{\text{LWF}}(t)$ under the LWF algorithm as follows:

$$D_{\text{LWF}}(t) = \frac{d}{dt}V(x(t)) = \alpha^f(t) - \mu_1^f(t), \text{for any } f \in \mathcal{M}_0(t).$$

In order to show Theorem 4, it suffices to show that

$$D_{\text{LWF}}(t) \le D^*(t). \qquad (22)$$

The inequality holds trivially when $x^f(t) = \frac{d}{dt}x^f(t) = 0$ for all $f \in \mathcal{M}_0(t)$, in which case all queues are close to empty at time t and in the immediate future of time t. It is then easy to verify that $D_{\text{LWF}}(t) = 0$ and $D^*(t) \ge 0$ (see details in [20]). Thus, we next focus on the case where $x^f(t) > 0$ or $\frac{d}{dt}x^f(t) = \alpha^f(t) - \mu_1^f(t) > 0$ for all $f \in \mathcal{M}_0(t)$ (recall that all flows $f \in \mathcal{M}_0(t)$ have the same values of $x^f(t)$ and $\frac{d}{dt}x^f(t)$). We need to introduce additional concepts and lemmas.

3) Mathematical Preliminaries for Theorem 4: Given an FSP $Z(t) = (a^f(t), x_i^f(t), x^f(t), e_i^f(t))$ under the LWF algorithm, where $x^f(t) > 0$ or $\frac{d}{dt}x^f(t) = \alpha^f(t) - \mu_1^f(t) > 0$ for all $f \in \mathcal{M}_0(t)$, we first introduce the concept of a *barrier* $b^f(t)$ for $f \in \mathcal{M}_0(t)$, which is defined as follows:

$$b^f(t) = \operatorname*{argmin}_i \{i | x_i^f(t) > 0 \text{ or } \frac{d}{dt}x_i^f(t) > 0\}. \qquad (23)$$

Since we already assume that either $x^f(t) > 0$ or $\frac{d}{dt}x^f(t) > 0$ holds for each flow $f \in \mathcal{M}_0(t)$, it is easy to check that the set $\{i | x_i^f(t) > 0 \text{ or } \frac{d}{dt}x_i^f(t) > 0\}$ is always non-empty, and thus the barrier $b^f(t)$ is always well defined. Intuitively, the "barrier" of a flow f is the closest logical link to the destination that has very large queue length in the immediate future (recall from (7) that the queue length is approximately $B\big(x_{b^f}^f(t) + \delta \frac{d}{dt}x_{b^f}^f(t)\big)$ at time $B(t+\delta)$, and B is large). For the barrier links, we have the following lemma.

Lemma 5. *Given an FSP $Z(t) = (a^f(t), x_i^f(t), x^f(t), e_i^f(t))$ under the LWF algorithm, where $x^f(t) > 0$ or $\frac{d}{dt}x^f(t) = \alpha^f(t) - \mu_1^f(t) > 0$ for all $f \in \mathcal{M}_0(t)$, let the set of barriers be $b^f(t)$, $f \in \mathcal{M}_0(t)$. For any $\epsilon > 0$, there exist $\tilde{\delta}(\epsilon) > 0$ and $J > 0$, such that, for all $0 < \delta < \tilde{\delta}(\epsilon)$, the following holds*

1) *$x^f(t+\delta) > x^{\tilde{f}}(t+\delta) + J\delta$ for any $f \in \mathcal{M}_0(t), \tilde{f} \notin \mathcal{M}_0(t)$;*
2) *for any $f \in \mathcal{M}_0(t)$, we have*

$$x_i^f(t+\delta) > J\delta, \text{ if } i = b^f(t),$$

$$x_i^f(t+\delta) < \frac{\epsilon J}{4L}\delta, \text{ if } i = 1, 2, ..., b^f(t) - 1,$$

where L is the total number of physical links.

Proof: Please refer to our technical report [20]. ■

Lemma 5 reveals the trend of the queue evolution in the immediate future after time t along the FSP. The first part of Lemma 5 states that all the flows in $\mathcal{M}_0(t)$ will have larger end-to-end backlog than that of the flows not in $\mathcal{M}_0(t)$, and the second part states that the total backlog between the barrier link and the destination is negligibly small compared to the backlog of the barrier link.

Based on Lemma 5, we can then study the scheduling decisions of the LWF algorithm. Specifically, we consider the FSP together with the original discrete-time system. Let $Z^B(\omega_B, t) = (a^{f,B}(\omega_B, t), x_i^{f,B}(\omega_B, t), x^{f,B}(\omega_B, t), e_i^{f,B}(\omega_B, t))$ be the sequence of scaled processes that converge uniformly to $Z(t) = (a^f(t), x_i^f(t), x^f(t), e_i^f(t))$. Using Lemma 5, we then have the following lemma.

Lemma 6. *Consider the FSP $Z(t)$ in Lemma 5. Given any $\epsilon \in (0,1)$ and the corresponding $\tilde{\delta}(\epsilon)$ (given by Lemma 5), for any fixed $\delta \in (0, \frac{\tilde{\delta}(\epsilon)}{2})$, there exist $\tilde{B}(\delta) > 0$, such that for all $B > \tilde{B}(\delta)$ and all time slots $t_0 \in (B(t+\delta), B(t+2\delta))$, the following holds*

1) *The weight of any logical link $l_{\tilde{i}}^{\tilde{f}} \notin \mathcal{E}_T^L(\mathcal{M}_0(t), b_{\mathcal{M}_0}(t))$ is strictly smaller than the weight of any logical link $l_{b^f}^f, f \in \mathcal{M}_0(t)$, i.e.,*

$$q_{\tilde{i}}^{\tilde{f}}(\omega_B, t_0) < q_{b^f}^f(\omega_B, t_0).$$

2) *Consider one specific link $l_{b^f}^f, f \in \mathcal{M}_0(t)$. Let $\mathcal{I}'(l_{b^f}^f)$ be the set of all logical links in $\mathcal{E}_T^L(\mathcal{M}_0, b_{\mathcal{M}_0})$ that interfere with link $l_{b^f}^f$. Then, at least one link in $\mathcal{I}'(l_{b^f}^f) \bigcup \{l_{b^f}^f\}$ should be scheduled at time instance t_0.*
Moreover, suppose that any two links in $\mathcal{I}'(l_{b^f}^f)$ interfere with each other. Then, if any link $l_{\hat{i}}^{\hat{f}} \in \mathcal{I}'(l_{b^f}^f) \bigcup \{l_{b^f}^f\}$ is scheduled, the usage efficient of link $l_{\hat{i}}^{\hat{f}}$ must satisfy

$$\eta_{\hat{i}}^{\hat{f}}(\omega_B, t_0) > 1 - \epsilon.$$

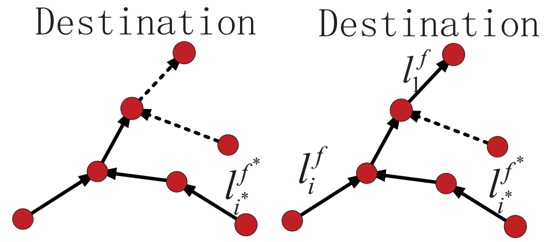
Proof: Please refer to our technical report [20]. ■

The rigorous proof of Lemma 6 has to deal with the original discrete-time system. However, it is easier to explain the intuition in the sense of FSP. In the immediate future of time t, those barrier links always have none zero backlog. Therefore, in the original discrete-time system, their backlog must be larger than their capacity, and thus their usage efficiency must be 1. Hence, the weight of each barrier link is approximately equal to the maximum end-to-end backlog. For those logical links not in $\mathcal{E}_T^L(\mathcal{M}_0, b_{\mathcal{M}_0})$, it is easy to check that their weights are strictly smaller than the maximum end-to-end backlog. To see this, note that either the corresponding flow is not in $\mathcal{M}_0$; or if it is in $\mathcal{M}_0$, at least the backlog of the barrier link must be subtracted from the weight of the link. Therefore, they must have smaller weights than those barrier links, as is stated in part (1). The results in part (2) are direct corollaries of part (1). For the first part of (2), if none of the links in $\mathcal{I}'(l^f_{b^f})$ is scheduled, link $l^f_{b^f}$ will have larger weight than the rest of its interfering links not in $\mathcal{E}^L_T(\mathcal{M}_0, b_{\mathcal{M}_0})$. Then link $l^f_{b^f}$ should be scheduled. As for the second part of (2), applying the same argument, we know that for link $l^{\hat{f}}_{\hat{i}} \in \mathcal{I}'(l^f_{b^f})$ to be activated, it must have weight larger than that of $l^f_{b^f}$. Note that if we ignore the usage efficiency term, the weights of link $l^{\hat{f}}_{\hat{i}}$ and link $l^f_{b^f}$ are comparable because they are both approximately equal to the maximum end-to-end backlog in FSP, and further, link $l^f_{b^f}$ has usage efficiency equal to 1. Hence, $l^{\hat{f}}_{\hat{i}}$ must also have usage efficiency close to 1, otherwise it cannot be scheduled.

Remark 2. We note that the condition that "any two links in $\mathcal{I}'(l^f_{b^f})$ interfere with each other" in part (2) is critical. If this condition does not hold, then it is possible to schedule two links $l^{\hat{f}}_{\hat{i}}$ and $l^{\hat{f}_2}_{\hat{i}_2}$ in $\mathcal{I}'(l^f_{b^f})$. According to the LWF algorithm, the first link $l^{\hat{f}}_{\hat{i}}$ needs to have a weight larger than that of $l^f_{b^f}$ in order to be scheduled. After the link $l^{\hat{f}}_{\hat{i}}$ is scheduled, The LWF algorithm removes all logical link interfering with $l^{\hat{f}}_{\hat{i}}$ (including $l^f_{b^f}$). As a result, the second link $l^{\hat{f}_2}_{\hat{i}_2}$ does not need to have a weight larger than that of $l^f_{b^f}$, and thus its usage efficiency could be smaller than $1-\epsilon$. In the following lemma, we show that for a special barrier link, the above condition always holds.

Lemma 7. *Let $l^{f^*}_{i^*}$ be the farthest logical link from the destination in the subnetwork $\mathcal{N}(\mathcal{M}_0(t), b_{\mathcal{M}_0}(t))$ (note that this definition implies that $i^* = b^{f^*}(t)$). Then, under the K-hop interference model, any two links in $\mathcal{I}'(l^{f^*}_{i^*})$ interfere with each other.*

The detailed proof can be found in Appendix A. We use Fig. 1 to illustrate Lemma 7. Both Fig. 1(a) and Fig. 1(b) show the same subnetwork $\mathcal{N}(\mathcal{M}_0(t), b_{\mathcal{M}_0}(t))$. In Fig. 1(a), $l^{f^*}_{i^*}$ is the furthest link away from the destination. Assuming a 2-hop interference model, we mark by solid lines all links that interfere with link $l^{f^*}_{i^*}$ in Fig. 1(a). We can easily check that all (solid) links interfering with $l^{f^*}_{i^*}$ also interfere with each other. We emphasize the importance of the condition that $l^{f^*}_{i^*}$ is the furthest link from the destination. For example, in Fig. 1(b) we pick another link l^f_i that is *not* furthest away from the destination and also mark all of its interfering links by solid lines. Even though links $l^{f^*}_{i^*}$ and l^f_1 both interfere with l^f_i, they do not interfere with each other.



(a) All links in $\mathcal{I}'(l^{f^*}_{i^*})$ interfere with each other.

(b) Both $l^{f^*}_{i^*}$ and l^f_1 are in $\mathcal{I}'(l^f_i)$, but they do not interfere with each other.

Fig. 1. Two examples for Lemma 7 under 2-hop interference model.

4) Proof of Theorem 4: We are now ready to prove Theorem 4. Recall the earlier discussion that we only need to show (22) for the non-trivial case where $x^f(t) > 0$ or $\frac{d}{dt}x^f(t) = \alpha^f(t) - \mu^f_1(t) > 0$ for all $f \in \mathcal{M}_0(t)$. Suppose that (22) does not hold. Then, there must exist $\tilde{\mu}^f_i$ satisfying (18)(19) and (20), such that

$$\alpha^f(t) - \tilde{\mu}^f_1 < D_{\text{LWF}}(t) = \alpha^f(t) - \mu^f_1(t), \text{ for all } f \in \mathcal{M}_0(t).$$

Therefore, $\mu^f_1(t) < \tilde{\mu}^f_1$ for all $f \in \mathcal{M}_0(t)$. Thus, we can find $\epsilon > 0$ such that

$$(1-\epsilon)^{-1}\mu^f_1(t) < \tilde{\mu}^f_1, \text{ for all } f \in \mathcal{M}_0(t). \tag{24}$$

Let $l^{f^*}_{i^*}$ be the farthest logical link from the destination in the subnetwork $\mathcal{N}(\mathcal{M}_0(t), b_{\mathcal{M}_0}(t))$, and let all of its interfering links in $\mathcal{E}^L_T(\mathcal{M}_0, b_{\mathcal{M}_0}(t))$ be $l^{f_1}_{i_1}, l^{f_2}_{i_2}, ..., l^{f_y}_{i_y}$. According to Lemma 7, we know that all the links $l^{f^*}_{i^*}, l^{f_1}_{i_1}, l^{f_2}_{i_2}, ..., l^{f_y}_{i_y}$ interfere with each other. We also note that $\tilde{\mu}^f_i$ satisfies the constraint (18). Thus, we must have

$$\frac{\tilde{\mu}^{f^*}_{i^*}}{R_{l^{f^*}_{i^*}}} + \sum_{k=1}^{y} \frac{\tilde{\mu}^{f_k}_{i_k}}{R_{l^{f_k}_{i_k}}} \le 1.$$

Further, $\tilde{\mu}^f_i$ satisfies both (20) and (24). Thus, we have

$$\tilde{\mu}^f_i \ge \tilde{\mu}^f_{i-1} \ge ... \ge \tilde{\mu}^f_1 > (1-\epsilon)^{-1}\mu^f_1(t),$$

and

$$1 \ge \frac{\tilde{\mu}^{f^*}_{i^*}}{R_{l^{f^*}_{i^*}}} + \sum_{k=1}^{y} \frac{\tilde{\mu}^{f_k}_{i_k}}{R_{l^{f_k}_{i_k}}} > \frac{1}{1-\epsilon}\left(\frac{\mu^{f^*}_1(t)}{R_{l^{f^*}_{i^*}}} + \sum_{k=1}^{y} \frac{\mu^{f_k}_1(t)}{R_{l^{f_k}_{i_k}}}\right),$$

and

$$\frac{\mu^{f^*}_1(t)}{R_{l^{f^*}_{i^*}}} + \sum_{k=1}^{y} \frac{\mu^{f_k}_1(t)}{R_{l^{f_k}_{i_k}}} < 1-\epsilon. \tag{25}$$

On the other hand, for the above ϵ, according to Lemma 6 part (2), there exist $\tilde{\delta}(\epsilon)$ and $\tilde{B}(\delta)$ such that, for $0 < \delta < \frac{\tilde{\delta}(\epsilon)}{2}$ and $B > \tilde{B}(\delta) > 0$, in the time period $(B(t+\delta), B(t+2\delta))$, at least one of the links $l^{f^*}_{i^*}, l^{f_1}_{i_1}, l^{f_2}_{i_2}, ..., l^{f_y}_{i_y}$ should be scheduled. Further, according to Lemma 7, these links interfere with each other. Then, exactly one of these links must be scheduled by the LWF algorithm at each time instant. We use $\gamma_{l^f_i}(\tau)$ to represent the scheduling decision of the logical link l^f_i at time-slot τ. Specifically, $\gamma_{l^f_i}(\tau) = 1$ if the logical link l^f_i is scheduled at time τ, and $\gamma_{l^f_i}(\tau) = 0$ otherwise. Then, we must have

$$\sum_{\tau=\lceil B(t+\delta)\rceil}^{\lfloor B(t+2\delta)\rfloor} \gamma_{l^{f^*}_{i^*}}(\tau) + \sum_{k=1}^{y} \sum_{\tau=\lceil B(t+\delta)\rceil}^{\lfloor B(t+2\delta)\rfloor} \gamma_{l^{f_k}_{i_k}}(\tau) = g(B,\delta),$$

where $\lceil w \rceil$ is the smallest integer that is no smaller than w, $\lfloor w \rfloor$ is the largest integer that is no larger than w, and $g(B,\delta) = \lfloor B(t+2\delta) \rfloor - \lceil B(t+\delta) \rceil + 1$. Further, according to Lemma 6, whenever one of these links is scheduled, its usage efficiency must be larger than $1-\epsilon$. We then have,

$$\begin{aligned} g(B,\delta) \quad &\geq \sum_{\tau=B(t+\delta)}^{B(t+2\delta)} \frac{E_{i^*}^{f^*}(\tau)}{R_{l_{i^*}^{f^*}}} + \sum_{k=1}^{y} \sum_{\tau=B(t+\delta)}^{B(t+2\delta)} \frac{E_{i_k}^{f_k}(\tau)}{R_{l_{i_k}^{f_k}}} \\ &> \quad (1-\epsilon) g(B,\delta). \end{aligned}$$

Dividing both sides by $B\delta$ and using the scaling in (7), we obtain,

$$\begin{aligned} \frac{g(B,\delta)}{B\delta} \geq \frac{1}{\delta} \Bigg(\frac{o(B)}{B} + \frac{e_{i^*}^{f^*,B}(t+2\delta) - e_{i^*}^{f^*,B}(t+\delta)}{R_{l_{i^*}^{f^*}}} \\ + \sum_{k=1}^{y} \frac{e_{i_k}^{f_k,B}(t+2\delta) - e_{i_k}^{f_k,B}(t+\delta)}{R_{l_{i_k}^{f_k}}} \Bigg) > (1-\epsilon) \frac{g(B,\delta)}{B\delta}. \end{aligned}$$

We first fix δ, and let $B \to \infty$. We obtain

$$\begin{aligned} 1 \quad \geq \quad & \frac{1}{\delta} \Bigg(\frac{e_{i^*}^{f^*}(t+2\delta) - e_{i^*}^{f^*}(t+\delta)}{R_{l_{i^*}^{f^*}}} \\ & + \sum_{k=1}^{y} \frac{e_{i_k}^{f_k}(t+2\delta) - e_{i_k}^{f_k}(t+\delta)}{R_{l_{i_k}^{f_k}}} \Bigg) > 1-\epsilon. \end{aligned}$$

Since this is true for all δ, we then let $\delta \to 0$, and obtain

$$\frac{\mu_{i^*}^{f^*}(t)}{R_{l_{i^*}^{f^*}}} + \sum_{k=1}^{y} \frac{\mu_{i_k}^{f_k}(t)}{R_{l_{i_k}^{f_k}}} \geq 1-\epsilon. \qquad (26)$$

Finally, recall from the definition (23) of $b_f(t)$ that, $\frac{d}{dt} x_i^f(t) = \mu_{i+1}^f(t) - \mu_i^f(t) = 0$ for all $f \in \mathcal{M}_0(t)$ and $i = 1, ..., b_f(t) - 1$. We then have $\mu_1^{f^*}(t) = \mu_{i^*}^{f^*}(t), \mu_1^{f_k}(t) = \mu_{i_k}^{f_k}(t), k = 1, 2, ..., y$. Then, Eqn. (26) becomes

$$\frac{\mu_1^{f^*}(t)}{R_{l_{i^*}^{f^*}}} + \sum_{k=1}^{y} \frac{\mu_1^{f_k}(t)}{R_{l_{i_k}^{f_k}}} \geq 1-\epsilon,$$

which contradicts Eqn. (25). Therefore, the inequality (22) and the result of Theorem 4 must hold.

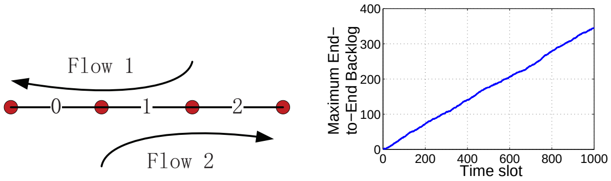
Remark 3. From the above proof, one may get the impression that as long as an algorithm made the "right" decisions in the neighborhood of the link $l_{i^*}^{f^*}$, it would have achieved the same performance guarantee as the LWF algorithm. However, the challenge is to make the "right" decisions even before the algorithm knows which flow-link pair (f^*, i^*) is the most critical. Thus, it is non-trivial to find the LWF algorithm and to show that it satisfies the desired properties summarized in Lemmas 5-7.

IV. Maximum End-to-End Backlog Minimization in General Networks

Given the success of the LWF algorithm and the drift-minimizing criterion in converge-case networks, we next seek to find such an optimal algorithm under more general network topology settings. Unfortunately, the LWF algorithm is not large-deviations optimal for a tree topology that is not a converge-cast. In fact, it is not even throughput optimal (see Section IV-A). We then study all other possible algorithms in Section IV-B. We prove an even stronger result that given the Lyapunov function $V(x(t)) = \max_f \{x^f(t)\}$, there exists no drift-minimizing algorithm for the simple linear network in Fig. 2(a).

A. Throughput Sub-Optimality of the LWF Algorithm

We consider the network in Fig. 2(a). We have a linear network with four nodes and three links. Two flows are active in this network. The routes of the two flows are $(1, 0)$ and $(1, 2)$, respectively. Assume that the capacity of each link is 1, *i.e.*, at each time slot, one link can transmit at most one packet. As for the interference model, we use the 1-hop interference model (i.e., $K = 1$) here.



(a) Network topology. (b) Max end-to-end backlog.

Fig. 2. A 2-flow example. Flow 1 traverses link 1 and link 0; Flow 2 traverses link 1 and link 2.

We simulate the LWF algorithm when both flows have i.i.d. arrivals. For each flow, there is a newly-arrival packet with probability 0.6 at each time slot. Hence, the arrival rates are 0.6 for both flows. Note that this system is stabilizable (i.e., the queues do not increase unboundedly) if we schedule link 1 twice every 3 time slots and schedule link 0 and link 2 together once every 3 time slots. However, the system is unstable under the LWF algorithm. In Fig. 2(b), we simulate the system for 1000 time slots based on the LWF algorithm. We can see that the maximum end-to-end backlog increases unboundedly.

This example thus indicates that the LWF algorithm is not even throughput-optimal for a tree network that is not a converge-cast. As a result, it cannot be large-deviations optimal in this setting either.

B. Drift Minimization in General Networks

Given that the LWF algorithm is not optimal for a tree topology that is not converge-cast, our next hope is that there may still be other algorithms that are optimal for these more general settings. Recall that we establish the optimality of LWF by showing that LWF satisfies the drift minimization property. Clearly, LWF cannot minimize the drift for the network setting in Fig. 2(a) (otherwise it would have been throughput optimal). Then, a natural question is whether there exist other algorithms that can minimize the drift for the network in Fig. 2(a). Unfortunately, we will prove next that no algorithm can minimize the drift in that network for the Lyapunov function $V(x(t)) = \max_f \{x^f(t)\}$. Hence, in order to design optimal scheduling algorithms, we must find new criteria (other than drift minimizing).

Theorem 8. *For the network in Fig. 2(a), no algorithm can minimize the drift at every regular time for every FSP.*

In order to prove Theorem 8, we need the following lemma.

Lemma 9. *Given any FSP $Z(t) = (a^f(t), x_i^f(t), x^f(t), e_i^f(t))$ for the network shown in Fig. 2(a), let $(a^1(t), a^2(t))$ denote the arrival process in FSP and let $\alpha^f(t) = \frac{d}{dt}a^f(t), f = 1, 2$. Assume that $x_i^f(0) = 0$, for all $f = 1, 2, i = 1, 2$ and that the algorithm π can minimize the drift at every regular time of the FSP. Then, if $|\alpha^1(t) - \alpha^2(t)| \leq \frac{1}{2}$ almost everywhere(a.e.) in $[0, T]$, we must have $x_2^1(t) = x_2^2(t)$ and $x_1^1(t) = x_1^2(t) = 0$ in $[0, T]$*[6].

Proof: Please refer to our technical report [20]. ■

The condition $|\alpha^1(t) - \alpha^2(t)| \leq \frac{1}{2}$ states that the rates of the two flows do not differ too much. If such an assumption is satisfied for an FSP, then there exists a feasible scheduling decision vector μ, such that $\alpha^1(t) - \mu_1^1(t) = \alpha^2(t) - \mu_1^2(t)$. Under this assumption, a drift-minimizing algorithm will be able to balance the end-to-end backlog for the two flows.

Now we are ready to prove theorem 8. The high-level idea of our proof is as follows. Recall that, given a sequence of scaled process $Z^B(\omega_B, t) = (a^{f,B}(\omega_B, t), x_i^{f,B}(\omega_B, t), x^{f,B}(\omega_B, t), e_i^{f,B}(\omega_B, t))$, there may be multiple subsequences that converge to different FSPs. Note that once an algorithm is given, for each B the corresponding scaled process will be completely determined, independent of which subsequence is taken to yield the FSP. Thus, for the algorithm to be drift-minimizing, its decision in each scaled process $Z^B(\omega_B, t)$ must be *simultaneously* drift-minimizing in the limit for all FSPs, regardless of which subsequence is taken. Thus, our strategy is to create a scenario such that, if the algorithm is drift-minimizing for one set of subsequences and the associated FSPs, then it cannot be drift-minimizing for another FSP taking along a further subsequence. Specifically, our steps to construct this contradictory scenario are as follows:

Step 1: We construct a sequence of FSPs indexed by j, each of which has a certain property on the arrival process (we will refer to these FSPs as Type-1 FSPs later on). For the j-th FSP, we construct a sequence of processes $\omega_{jk}, k = 1, 2, ...$ that converges to it with the scaling parameter $B_{j,k} = jk$.

Step 2: Then, assuming that an algorithm π is drift-minimizing for all the Type-1 FSPs, we show that the resulting queue length must be increasing in time. Thus, for the j-th FSP, we will be able to pick a process ω_{j,k_j} such that the *unscaled* queue length at time $B_{j,k_j}T = jk_jT$ is larger than $\frac{1}{25}B_{j,k_j}T$.

Step 3: Next, we take a subsequence of the chosen processes ω_{j,k_j} in Step 2 and form a new FSP using the scaling $B_j = B_{j,k_j} = jk_j$ (which we will refer to as a Type-2 FSP). Again, once an algorithm π is given, for each ω, the corresponding process will be completely determined, independently of which subsequence is taken to produce the various FSPs. Thus, from the construction above, the queue-length in the new Type-2 FSP should also be large than $\frac{1}{25}T$ at time T.

Step 4: However, we will show that, in order for the algorithm π to be drift-minimizing for the new Type-2 FSP, its queue-length should diminish to zero. This results into a contradiction. We will therefore conclude that no algorithm can be drift-minimizing for all FSPs.

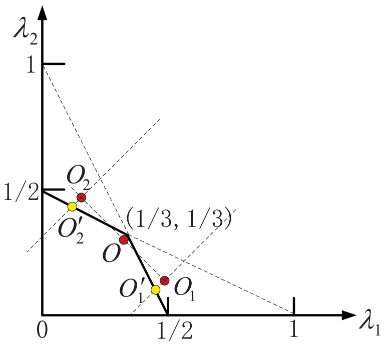


Fig. 3. The capacity region of the network in Fig. 2(a). O_1, O_2 are outside the capacity region; O'_1, O'_2 are at the boundary of the capacity region; $O = \frac{O_1+O_2}{2}$ is inside the capacity region.

We next explain the desired high-level properties of the two types of FSPs. Consider the capacity region of the network in Fig. 2(a). Let the arrival rate of flow i ($i = 1, 2$) be λ_i. Under the one-hop interference model, the capacity region can be characterized by (see Fig. 3)

$$2\lambda_1 + \lambda_2 \leq 1, \lambda_1 + 2\lambda_2 \leq 1.$$

For a Type-1 FSP, the instantaneous arrival rate $(\alpha^1(t), \alpha^2(t))$ is either at point $O_1 = (\frac{1}{2}, \frac{1}{8})$ or at point $O_2 = (\frac{1}{8}, \frac{1}{2})$. Then, we show that in order for the algorithm π to minimize the drift at every regular time, its service rate vector must be either at point $O'_1 = (\frac{11}{24}, \frac{1}{12})$ or at point $O'_2 = (\frac{1}{12}, \frac{11}{24})$. Thus, the queues for both flows will increase in time. On the other hand, for the Type-2 FSP, the instantaneous arrival rate $(\alpha^1(t), \alpha^2(t))$ is at the mid-point $O = \frac{O_1+O_2}{2} = (\frac{5}{16}, \frac{5}{16})$. Since O is inside the capacity region, we will show that, in order for the algorithm π to minimize the drift, the queue length of both flows should decrease to zero. Thus, we will obtain the afore-mentioned contradiction.

Proof of Theorem 8: We now present the full proof. Before we follow the Steps 1-4 outlined earlier, we describe the set of realizations of the arrival process that we will use in our construction. Like in Section IV-A, we assume that a new packet arrives to a flow with a given probability p, i.i.d. across time and flows.

We define two matrices M_1 and M_2.

$$M_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

We construct a sequence of realizations $\omega_{j,k}$, indexed by $j = 1, 2, ...$ and $k = 1, 2, ...$ Specifically, for the realization $\omega_{j,k}$,

[6] Note that throughout the paper we have used the superscript to denote the index of the flow. For example, the superscripts in $x_1^1(t), x_1^2(t), a^1(t), a^2(t)$ denote flow 1 and flow 2, respectively.

its discrete-time arrival sequence is given by

$$\begin{aligned} A(\omega_{j,k}) \quad = \quad & [\overbrace{\underbrace{M_1M_1\cdots M_1}_{k\text{-repetition}}\underbrace{M_2M_2\cdots M_2}_{k\text{-repetition}}}^{\text{1st group}}\underbrace{M_1M_1\cdots M_1}_{k\text{-repetition}} \\ & \underbrace{M_2M_2\cdots M_2}_{k\text{-repetition}}\cdots\overbrace{\underbrace{M_1M_1\cdots M_1}_{k\text{-repetition}}\underbrace{M_2M_2\cdots M_2}_{k\text{-repetition}}}^{j\text{-th group}}]. \end{aligned}$$

We can see that $A(\omega_{j,k})$ has two rows and jkT (here $T=16$) columns. The two rows of $A(\omega_{j,k})$ correspond to the two flows in Fig. 2(a), and each column corresponds to one time slot. Readers can easily see that the average arrival rates in each k-repetition interval are either at $O_1 = (\frac{1}{2}, \frac{1}{8})$ or $O_2 = (\frac{1}{8}, \frac{1}{2})$ in Fig. 3, but the long-term average rate is at $O = (\frac{5}{16}, \frac{5}{16})$. We note that, although the arrival sequence seems to exhibit high correlation over time, such a choice is still consistent with our original assumption that the arrival process $A(t)$ is i.i.d. in time. This is because for FSP we are interested in rare events. Thus, it is legitimate to consider samples $\omega_{j,k}$ that are "rare" when we are constructing an FSP (see also the definition of FSP in Section III-B1).

Step 1: Fix $T=16$, we first construct a sequence of Type-1 FSPs over $[0,T]$ indexed by j. For each j, we choose the sequence of arrival processes $A(\omega_{j,1}), A(\omega_{j,2}), \ldots$ We scale the arrival process $A(\omega_{j,k})$ by $B_{j,k} = jk$ using Eqn. (7), i.e.,

$$a^{B_{j,k}}(\omega_{j,k}, t) = \frac{1}{B_{j,k}} \sum_{\tau=0}^{B_{j,k}t-1} A(\omega_{j,k}, \tau), t \in [0,T].$$

Note that in each $\omega_{j,k}$, each k-repetition of matrix M_1 (or M_2) takes $8k$ time-slots. Thus, for a fixed j, as k increases, each $8k$-slot interval in $\omega_{j,k}$ is scaled back by the scaling factor $B_{j,k} = jk$. As a result, the limiting FSP will be a piece-wise linear function of time, where each piece is of the length $\frac{T}{2j} = \frac{8}{j}$. More precisely, the arrival process $(a^{1,(j)}(t), a^{2,(j)}(t)), t \in [0,T]$ of the j-th limiting FSP is given by:

$$a^{1,(j)}(t) = \left\{ \begin{array}{rl} \frac{1}{2}t - \frac{3}{16}\frac{Ti}{j}, & \text{if } \frac{Ti}{j} \le t < \frac{T(2i+1)}{2j}, \\ \frac{1}{8}t + \frac{3}{16}\frac{Ti}{j} + \frac{3}{16}\frac{T}{j}, & \text{if } \frac{T(2i+1)}{2j} \le t < \frac{T(i+1)}{j}, \end{array} \right. \tag{27}$$

$$a^{2,(j)}(t) = \left\{ \begin{array}{rl} \frac{1}{8}t + \frac{3}{16}\frac{Ti}{j}, & \text{if } \frac{Ti}{j} \le t < \frac{T(2i+1)}{2j}, \\ \frac{1}{2}t - \frac{3}{16}\frac{Ti}{j} - \frac{3}{16}\frac{T}{j}, & \text{if } \frac{T(2i+1)}{2j} \le t < \frac{T(i+1)}{j}, \end{array} \right.$$

where $i = 0, 1, \ldots, j-1$. Note that the arrival rates in each interval of length $\frac{8}{j}$ are given by:

$$\alpha^{1,(j)}(t) = \frac{d}{dt} a^{1,(j)}(t) = \left\{ \begin{array}{ll} \frac{1}{2}, & \text{if } \frac{Ti}{j} \le t < \frac{T(2i+1)}{2j}, \\ \frac{1}{8}, & \text{if } \frac{T(2i+1)}{2j} \le t < \frac{T(i+1)}{j}, \end{array} \right.$$

$$\alpha^{2,(j)}(t) = \frac{d}{dt} a^{2,(j)}(t) = \left\{ \begin{array}{ll} \frac{1}{8}, & \text{if } \frac{Ti}{j} \le t < \frac{T(2i+1)}{2j}, \\ \frac{1}{2}, & \text{if } \frac{T(2i+1)}{2j} \le t < \frac{T(i+1)}{j}, \end{array} \right.$$

where $i = 0, 1, \ldots, j-1$, which correspond to either $O_1 = (\frac{1}{2}, \frac{1}{8})$ or $O_2 = (\frac{1}{8}, \frac{1}{2})$ in Fig. 3.

Step 2: We assume that an algorithm π is drift-minimizing for all the Type-1 FSPs. We now compute the queue evolution $x_i^{f,(j)}(t)$ and $x^{f,(j)}(t)$ under the arrival process $a^{(j)}(t)$ and the algorithm π. Note that $(\alpha^{1,(j)}(t), \alpha^{2,(j)}(t))$ always satisfies the conditions in Lemma 9. Therefore, $x_2^1(t) = x_2^2(t)$ and $x_1^1(t) = x_1^2(t) = 0$ in $[0,T]$ under the algorithm π. In this case, the optimization problem in (17) for the minimum drift becomes

$$\begin{array}{cl} \min\limits_{\hat{\mu} \ge 0} & \max\{\alpha_1^{(j)}(t) - \tilde{\mu}_1^1, \alpha_2^{(j)}(t) - \tilde{\mu}_1^2\} \\ \text{sub to} & \hat{\mu}_1^2 + \hat{\mu}_2^2 + \hat{\mu}_2^1 \le 1, \\ & \hat{\mu}_1^1 + \hat{\mu}_1^2 + \hat{\mu}_2^2 \le 1, \\ & \tilde{\mu}_1^1 = \min\{\hat{\mu}_1^1, \tilde{\mu}_2^1\}, \tilde{\mu}_2^1 = \hat{\mu}_2^1, \\ & \tilde{\mu}_1^2 = \min\{\hat{\mu}_1^2, \tilde{\mu}_2^2\}, \tilde{\mu}_2^2 = \hat{\mu}_2^2. \end{array} \tag{28}$$

If the algorithm π minimizes the drift for the j-th Type-1 FSP, we must have that $(\mu_1^{1,(j)}(t), \mu_1^{2,(j)}(t))$ equals the optimal solution of $(\tilde{\mu}_1^1, \tilde{\mu}_1^2)$ in (28). Therefore, by solving (28) we obtain

$$\mu_1^{1,(j)}(t) = \left\{ \begin{array}{ll} \frac{11}{24}, & \text{if } \frac{Ti}{j} \le t < \frac{T(2i+1)}{2j}, \\ \frac{1}{12}, & \text{if } \frac{T(2i+1)}{2j} \le t < \frac{T(i+1)}{j}, \end{array} \right. \quad i = 0, \ldots, j-1,$$

$$\mu_1^{2,(j)}(t) = \left\{ \begin{array}{ll} \frac{1}{12}, & \text{if } \frac{Ti}{j} \le t < \frac{T(2i+1)}{2j}, \\ \frac{11}{24}, & \text{if } \frac{T(2i+1)}{2j} \le t < \frac{T(i+1)}{j}, \end{array} \right. \quad i = 0, \ldots, j-1.$$

We note that $(\mu_1^{1,(j)}(t), \mu_1^{2,(j)}(t))$ corresponds to either O'_1 or O'_2 in Fig. 3.

Based on the values of $\alpha^{f,(j)}(t)$ and $\mu_1^{f,(j)}(t)$, it is easy to check that $x^{f,(j)}(T) = \frac{1}{24}T, f = 1, 2$. Note that for any fixed j, the sequence of scaled quantities $Z^{B_{j,k}}(\omega_{j,k}, t)$ has a subsequence that converges uniformly to the above FSP on $[0,T]$. Hence, there must exist $k_j > j$, such that

$$x^{f,B_{j,k_j}}(\omega_{j,k_j}, T) > \frac{1}{25}T, f = 1, 2. \tag{29}$$

Thus, before the scaling, the queue length at time $B_{j,k_j}T$ will be greater than $\frac{1}{25}B_{j,k_j}T$, i.e., $X^f(\omega_{k_j}, B_{j,k_j}T) > \frac{1}{25}B_{j,k_j}T$.

Step 3: Next, we take the realizations ω_{j,k_j} obtained above, and use the scaling $B_j = B_{j,k_j} = jk_j$. There will be a subsequence that converges uniformly to another FSP $Z^{(*)}(t) = [a^{f,(*)}(t), x_i^{f,(*)}(t), x^{f,(*)}(t), e_i^{f,(*)}(t)]$. Recall that, once the algorithm π is given, for each ω, the corresponding process will be completely determined. Hence, using (29), we can conclude that, in this new FSP, we must also have

$$x^{f,(*)}(T) = \lim_{j \to \infty} x^{f,B_{j,k_j}}(\omega_{j,k_j}, T) \ge \frac{1}{25}T. \tag{30}$$

The arrival process $a^{f,(*)}(t)$ of this new FSP, however, has a different form. Specifically, as j increases, the length of each k_j-repetition of M_1 (or M_2) increases as $8k_j$, but the scaling factor $B_j = jk_j$ increases even faster (due to the additional factor of j). Intuitively, each linear piece in (27) now becomes shorter and shorter. Therefore, the limiting arrival process will become a line given by the following

$$a^{f,(*)}(t) = \frac{5}{16}t, f = 1, 2.$$

The corresponding arrival rate vector is precisely $O = (\frac{5}{16}, \frac{5}{16})$ in Fig. 3, which is inside the capacity region. We use the above new FSP as the Type-2 FSP outlined earlier.

Step 4: However, we will show that, in order for the algorithm π to be drift-minimizing for the above Type-2 FSP, its queue-length should diminish to zero. Assume that the algorithm π also minimizes the drift for this Type-2 FSP. According to Lemma 9, in order to compute the scheduling decision vector in this case, we need to solve the following optimization problem:

$$\begin{aligned} \min_{\hat{\mu}\geq 0} \quad & \max\{\alpha^{1,(*)}(t)-\tilde{\mu}_1^1, \alpha^{2,(*)}(t)-\tilde{\mu}_1^2\} \qquad (31)\\ \text{sub to} \quad & \hat{\mu}_1^2+\hat{\mu}_2^2+\hat{\mu}_2^1 \leq 1,\\ & \hat{\mu}_1^1+\hat{\mu}_1^2+\hat{\mu}_2^2 \leq 1,\\ & \tilde{\mu}_1^1=\min\{\hat{\mu}_1^1,\tilde{\mu}_2^1\}, \tilde{\mu}_2^1=\min\{\alpha^{1,(*)}(t),\hat{\mu}_2^1\},\\ & \tilde{\mu}_1^2=\min\{\hat{\mu}_1^2,\tilde{\mu}_2^2\}, \tilde{\mu}_2^2=\min\{\alpha^{2,(*)}(t),\hat{\mu}_2^2\}. \end{aligned}$$

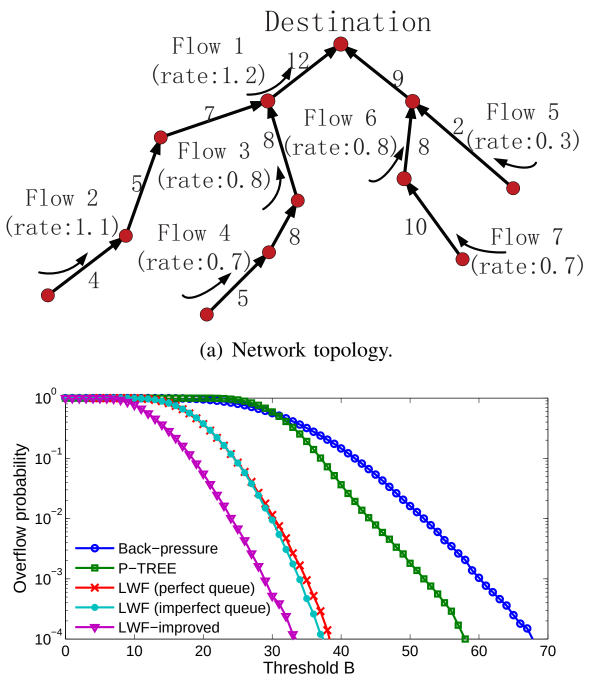
Solving (31) gives $\tilde{\mu}_1^f=\frac{5}{16}, f=1,2$. Since the algorithm π is drift-minimizing, its decision must also be

$$\mu_1^{f,(*)}(t)=\alpha^{f,(*)}(t)=\frac{5}{16}.$$

As a result, $x^{f,(*)}(T)=0$ for $f=1,2$, which contradicts (30). The result of Theorem 8 then follows.

V. Simulation

In this section, we present our simulation results for the topology shown in Fig. 4(a) under the 2-hop interference model. This topology contains 12 nodes. The number near each link represents its capacity. There are 7 flows in the network. The number of packets arriving at each flow per time slot is chosen to be Poisson distributed. The number near each flow indicates the arrival rate. We can verify that the offered-load vector has already exceeded 0.9 of the capacity region.



(a) Network topology.

(b) Queue overflow probability.

Fig. 4. Comparison between LWF, back-pressure, and P-TREE algorithms.

We are interested in the overflow probability of the maximum end-to-end backlog, i.e., $P\left[\max_{f=1,2,\ldots,7}\{X^f(t)\}\geq B\right]$. We simulate the system under different scheduling algorithms: LWF, back-pressure [2], and P-TREE [7].

We give a brief overview about these algorithms. In the back-pressure algorithm [2], the weight of each logical link is given by the difference between the backlog at this link and its subsequent link. For example, the weight of link l_i^f at time instance t is $X_i^f(t)-X_{i-1}^f(t)$. (Here, we use the convention that $X_0^f(t)=0$). Then, we schedule the set of non-interfering logical links that maximizes the total weight. As for the P-TREE algorithm [7], it is a large-deviations optimal algorithm for minimizing the total backlog of all flows, which gives priority to those links nearer to the destination and links that have larger capacity.

We will compute the above algorithms with the proposed LWF algorithm, along with the following two modified versions. First, note that in our analysis of the LWF algorithm, we have assumed perfect queue-length information in every time slot. However, in practice, the delivery of queue-length information may suffer from delay and loss. Therefore, we simulate the performance of the LWF algorithm both with and without perfect queue-length information. In the latter case, the weight of each logical link l at time t is based on the queue-length information $r_l(t)$ slots ahead, where $[r_l(t)]$ is a set of i.i.d. random variables uniformly distributed in [0,20]. Second, we note that the proposed LWF algorithm may waste some capacity. Specifically, if a logical link with usage efficiency smaller than 1 is scheduled, then some capacity of the corresponding physical link is wasted. We could have made use of this part of capacity to serve other logical links (from different flows) to further improve the performance. Thus, we also simulate an improved version of the LWF algorithm (with perfect queue-length) as follows. In the improved LWF algorithm, for each physical link that is scheduled to serve a logical link with usage efficiency less than 1, we use the remaining capacity to serve additional logical links that correspond to the same physical link, according to the weight of each link (logical links with larger weights are served first).

In Fig. 4(b), we compare the above algorithms, and plot the overflow probability $P\left[\max_{f=1,2,\ldots,7}\{X^f(t)\}\geq B\right]$ versus the threshold B with the y-axis in the log scale. We observe that our LWF algorithm performs best not only in terms of decay rate, but also in terms of actual overflow probability. The performances of the P-TREE algorithm[7] and the back-pressure algorithm are both significantly worse. This is because, in our network setting, while flow 1 and flow 2 have similar rates, they are at different depth: flow 1 is at depth 1, and flow 2 is at depth 4. In the P-TREE algorithm, priority is given to flow 1 as it is closer to the destination. Similarly, in the back-pressure algorithm, the link l_1^1 is more likely to have larger weight, since flow 1 only has one hop. Hence, flow 1 again has a higher

[7] Although in Fig. 4(b), the tail decay rates of LWF and other algorithms do not differ greatly, it is not difficult to construct examples where their difference is larger. The simplest example would be a system with a single link and two flows. In that case, the P-TREE algorithm reduces to first-come-first-serve, which can perform poorly for minimizing the larger backlog among the two flows. For details, please refer to our technical report [20].

priority. For both P-TREE and back-pressure, giving priority to flow 1 takes away the packet transmission opportunity of flow 2. Therefore, their performances are poor in terms of the maximum end-to-end backlog. An interesting observation is that the LWF algorithm with imperfect queue-length information has similar performance as the LWF algorithm with perfect queue-length information. We conjecture that the underlying reason for this insensitivity is that, since both the departure rate and the arrival rate are bounded, the weight of each link at some time t will not differ too much from its weight at time $t - r_l(t)$. Thus, whenever the weights are large, Lemma 5 and Lemma 6 would still approximately hold for the LWF algorithm with imperfect queue-length information. This suggest that the tail overflow probability will not differ much, especially when the backlog threshold B is large. The numerical results in Fig. 4 (b) suggest that even for medium values of backlog thresholds, the overflow probability will also be quite similar. Finally, we note that the improved LWF algorithm indeed achieves better performance, although its asymptotic decay-rate remains the same.

VI. Conclusion

We study the scheduling problem for multi-hop wireless network under the K-hop interference model. We first focus on the case of converge-cast on a tree topology. Using a large-deviation framework, we design a new LWF algorithm and show that it is large-deviations optimal for minimizing the maximum end-to-end backlog across flows. We prove the large-deviation optimality of the LWF algorithm by showing that it minimizes the drift at every time in every FSP. Then, we study large-deviations optimal algorithms in a more general setting. We provide a negative result that drift minimizing algorithms do not exist for some topologies. Finally, the simulation results indicate that the proposed LWF algorithm significantly outperforms other algorithms in the literature not only in terms of the asymptotic decay rate, but also in terms of the actual overflow probability.

The negative result in Section IV-B reveals the limitation of the “drift-minimizing” criterion in [14]. There are a number of interesting questions for future work. First, given that drift-minimizing algorithms do not exist for some topologies, can we still find the algorithm(s) with the optimal decay-rate? Can we develop new criteria that are more general than the drift-minimizing criterion? Second, perhaps achieving the optimal decay rate is too difficult, and we may be content with finding algorithms with good decay-rates. Then, what will be a useful structure for such algorithms? In particular, even though drift-minimizing algorithms are not possible, can we find those with “good” drifts so that they can still lead to good decay-rate? A deeper understanding to these questions will help us develop low-delay algorithms for more general settings.

We note an interesting connection between the results presented in this paper and the local-pooling condition reported in the literature [21]. Indeed, our proposed LWF algorithm shares some similarity to the GMS (Greedy Maximal Scheduling) algorithm [22] and the LQF (Longest Queue First) algorithm [21] in the literatures. The latter has been shown to be throughput-optimal for single-hop networks satisfying the local-pooling condition. Further, our construction of the counter example also seems to suggest a connection to a local-pooling condition as well. Thus, it would be highly appealing if these results can be generalized from tree networks to any network that satisfies a local-pooling type of condition. However, we caution that existing results on the local pooling condition mainly focus on single-hop networks. In the literature, there has been limited success in extending the local polling condition to multi-hop networks. For example, [23] only studies a linear network with a single destination. In another work [24], while the authors present a definition of “multi-hop local pooling condition,” only a very limited set of topologies can be verified to satisfy the condition. Further, these results only account for throughput optimality, not queueing performance. In contrast, our optimality result for the LWF algorithm holds for the overflow probability under a more general multi-hop setting (i.e., tree networks with converge-cast). In our future work, we will carefully study whether we can define a suitable notion of multi-hop local pooling that can be easily verified for network topologies beyond trees, and show that the LWF type of algorithms can also be large-deviations optimal.

References

[1] S. Zhao and X. Lin, “On the design of scheduling algorithms for end-to-end backlog minimization in multi-hop wireless networks,” in *IEEE INFOCOM*, Orlando, FL, March 2012.

[2] L. Tassiulas and A. Ephremides, “Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multi-hop Radio Networks,” *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[3] L. Bui, R. Srikant, and A. L. Stolyar, “Novel Architectures and Algorithms for Delay Reduction in Back-Pressure Scheduling and Routing,” in *Infocom Mini-Conference*, April 2009.

[4] “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016,” *Cisco*, Feb. 2012. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.

[5] L. Ying, R. Srikant, A. Eryilmaz, and G. E. Dullerud, “A Large Deviations Analysis of Scheduling in Wireless Networks,” *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 5088–5098, Nov. 2006.

[6] V. J. Venkataramanan and X. Lin, “On Wireless Scheduling Algorithms for Minimizing the Queue-Overflow Probability,” *IEEE/ACM Trans. on Networking*, vol. 18, no. 3, June 2010.

[7] ——, “Low-Complexity Scheduling Algorithm for Sum-Queue Minimization in Wireless Convergecast,” in *IEEE INFOCOM*, Shanghai, China, April 2011.

[8] V. J. Venkataramanan, X. Lin, L. Ying, and S. Shakkottai, “On Scheduling for Minimizing End-to-end Buffer Usage Over Multihop Wireless Networks,” in *IEEE INFOCOM*, San Diego, CA, March 2010.

[9] L. Tassiulas and A. Ephremides, “Dynamic Scheduling for Minimum Delay in Tandem and Parallel Constrained Queueing Models,” *Annals of Operation Research*, vol. 48, pp. 333–355, December 1994.

[10] S. Hariharan and N. B. Shroff, “On Optimal Dynamic Scheduling for Sum-Queue Minimization in Trees,” in *WIOPT*, Princeton, NJ, May 2011.

[11] M. J. Neely, “Delay Analysis for Max Weight Opportunistic Scheduling in Wireless Systems,” *IEEE Transactions on Automatic Control*, vol. 54, no. 9, pp. 2137–2150, September 2009.

[12] ——, “Delay Analysis for Maximal Scheduling with Flow Control in Wireless Networks with Bursty Traffic,” *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1146–1159, August 2009.

[13] B. Hajek and R. G. Ogier, “Optimal dynamic routing in communication networks with continuous traffic,” *Networks*, vol. 14, no. 3, pp. 457–487, 1984.

[14] V. J. Venkataramanan and X. Lin, "On the Queue-Overflow Probability of Wireless Systems: A New Approach Combining Large Deviation with Lyapnov Functions," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6367–6392, 2013.

[15] G. Sharma, N. B. Shroff, and R. R. Mazumdar, "On the Complexity of Scheduling in Wireless Networks," in *ACM MOBICOM*, New York, NY, September 2006.

[16] C. Zhao and X. Lin, "On the overflow probability of distributed scheduling algorithms," *Computer Networks*, vol. 55, no. 1, January 2011.

[17] A. L. Stolyar, "Control of end-to-end delay tails in a multiclass network: Lwdf discipline optimality," *Annals of Applied Probability*, pp. 1151–1206, 2003.

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

[19] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*, 2nd ed. New York: Springer-Verlag, 1998.

[20] S. Zhao and X. Lin, "Design of Scheduling Algorithms for End-to-End Backlog Minimization in Wireless Multi-hop Networks under K-hop Interference Models," *Technical Report, https://engineering.purdue.edu/%7elinx/papers.html*, 2013.

[21] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Advances in Applied probability*, pp. 505–521, 2006.

[22] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1132–1145, 2009.

[23] X. Kang, J. J. Jaramillo, and L. Ying, "Stability of longest-queue-first scheduling in linear wireless networks with multihop traffic and one-hop interference," in *IEEE CDC*, Florence, Italy, December 2013.

[24] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop local pooling for distributed throughput maximization in wireless networks," in *IEEE INFOCOM*, Phoenix, AZ, April 2008.

Appendix

A. Proof of Lemma 7

Proof: Given the logical link $l_{i^*}^{f^*}$, let all of its interfering links in $\mathcal{E}_T^L(\mathcal{M}_0, b_{\mathcal{M}_0}(t))$ be $l_{i_1}^{f_1}, l_{i_2}^{f_2}, ..., l_{i_y}^{f_y}$. In order to prove Lemma 7, we consider any two links $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$, where m, n are two distinct indices ranging from 1 to y. Consider the two paths from $l_{i^*}^{f^*}$ to $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$, respectively. Let V be the farthest node from link $l_{i^*}^{f^*}$ that appears in both paths (see Fig. 5). Let a be the number of links between link $l_{i^*}^{f^*}$ and node V; let b be the number of links between node V and link $l_{i_m}^{f_m}$; and let c be the number of links between node V and link $l_{i_n}^{f_n}$. Since $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$ interfere with $l_{i^*}^{f^*}$ under K-hop interference model, the number of links (which is one less than the number of hops) between $l_{i_m}^{f_m}$ (or $l_{i_n}^{f_n}$) and $l_{i^*}^{f^*}$ must be smaller than K, i.e.,

$$a + b < K, a + c < K.$$

Then, we consider the two paths from the root to $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$. It is easy to check that at least one path traverses node V. Without loss of generality, we assume that the path from the root to link $l_{i_m}^{f_m}$ traverses node V. Then, we must have $b \leq a$. Otherwise, $l_{i_m}^{f_m}$ is farther from the root than $l_{i^*}^{f^*}$, which contradicts the assumption that $l_{i^*}^{f^*}$ is the farthest link from the root. Then, the total number of hops between $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$ is $b + c \leq a + c < K$. Therefore, $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$ interfere with each other.

Repeating the argument for each pair of links in $l_{i_1}^{f_1}$, $l_{i_2}^{f_2}$, ..., $l_{i_y}^{f_y}$, we know that all these links interfere with each other.

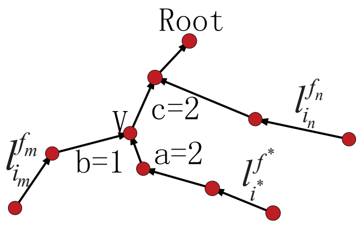


Fig. 5. An example showing that links $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$ also interfere with each other. Here, $K = 5$. Both $l_{i_m}^{f_m}$ and $l_{i_n}^{f_n}$ interfere with $l_{i^*}^{f^*}$ since $a + b < 5$ and $a + c < 5$, and $l_{i_m}^{f_m}$ interferes with $l_{i_n}^{f_n}$ since $b + c < 5$.

Shizhen Zhao received his B.S. from Shanghai Jiao Tong University, China in 2010, and is pursuing a Ph.D. degree at the School of ECE, Purdue University, West Lafayette, IN, USA. His research interests are in the analysis, control and optimization in wireless networks and smart grid.

Xiaojun Lin received his B.S. from Zhongshan University, Guangzhou, China, in 1994, and his M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, in 2000 and 2005, respectively. He is currently an Associate Professor of Electrical and Computer Engineering at Purdue University.

Dr. Lin's research interests are in the analysis, control and optimization of wireless and wireline communication networks. He received the IEEE INFOCOM 2008 best paper and 2005 best paper of the year award from Journal of Communications and Networks. His paper was also one of two runner-up papers for the best-paper award at IEEE INFOCOM 2005. He received the NSF CAREER award in 2007. He was the Workshop co-chair for IEEE GLOBECOM 2007, the Panel co-chair for WICON 2008, the TPC co-chair for ACM MobiHoc 2009, and the Mini-Conference co-chair for IEEE INFOCOM 2012. He is currently serving as an Area Editor for (Elsevier) Computer Networks Journal, and has served as a Guest Editor for (Elsevier) Ad Hoc Networks journal.