On the Design of Scheduling Algorithms for End-to-End Backlog Minimization in Multi-hop Wireless Networks

Shizhen Zhao and Xiaojun Lin
School of ECE, Purdue University, West Lafayette, IN, USA
Email: {zhao147, linx}@purdue.edu

***Abstract*—In this paper, we study the problem of link scheduling for multi-hop wireless networks with per-flow delay constraints. 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 significantly performs 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 wireless interference, there are both intra-flow constraints (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 interference 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 one is only concerned about the stability of the system, it is well-known that the back-pressure algorithm [1] 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 [7]. 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. We note that 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 works on the delay performance of single-hop wireless networks (e.g., [8][9]), results on the end-to-end delay performance for multi-hop wireless networks are more limited [6]. 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 [8][13], 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. Most existing results either focus on minimizing the end-to-end backlog of a single flow or on the total end-to-end backlog among all flows. For the single flow case (and therefore the system can be viewed as a tandem network), the work in [2] provides a scheduling algorithm that is sample-path optimal for minimizing the 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 with multiple competing flows, e.g., a tree topology with converge-cast, one can show that there may not even exist sample-path optimal algorithms [5]. Alternately, the works in [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. Finally, one may use large-deviations theory to characterize and compare the exponential decay rate with which the probability that some function of the end-to-end backlog exceeds a threshold approaches zero, when the overflow threshold becomes large. Our prior work [6] has applied such a large-deviations approach to a tree-network with converge-cast for minimizing the large-deviations decay rate of the probability that the sum of the end-to-end backlog over all flows exceeds a large threshold. There, we propose the P-TREE algorithm,

which is shown to be large-deviations optimal. Both the P-TREE algorithm and the algorithm in [2] share the same intuition: for either a single flow or for a converge-cast on a tree topology, one should give priority to links closer to the destination, which helps to reduce the total end-to-end backlog more quickly.

In this paper, we study a different but important setting compared to these prior studies (especially [6]). Specifically, 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 practically 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. 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 in [2][6], in order to minimize the end-to-end backlog of 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, one needs to give priority to flows whose end-to-end backlog is large. The key difficulty is that these two priorities are not always consistent with each other! In another prior work [13], these two priorities are asymptotically attained together in the limit by a sequence of scheduling algorithms, which are then shown to be asymptotically optimal in the large-deviations sense. However, in practice the algorithms approaching the limit can have large overflow probabilities when the overflow threshold of interest is not very large. 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 provide a new large-deviations optimal algorithm, called Largest-Weight-First (LWF), for minimizing the maximum end-to-end backlog among all flows of a converge-cast on a tree topology. 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 in small thresholds of interest. To the best of our knowledge, this is the first such optimal algorithms for minimizing the maximum end-to-end backlog among all flows in any converge-cast scenario.

The optimality of LWF is shown based on one of our earlier results in [3] that, under suitable conditions, an algorithm that minimizes the drift of a Lyapunov function at every time in every fluid sample paths (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 a 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 such 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. Further, the algorithm and the techniques that we use are both new.)

Given the success of the LWF algorithm in converge-cast scenarios, we are then interested in developing similar optimal algorithms for more general problem 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 a negative, but fundamental, result. We find a tree topology that is not a converge-cast, where we show that there exist no algorithms 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 to design large-deviations optimal scheduling algorithms.

The rest of the paper is organized as follows. Section II defines the system model. We propose the LWF algorithm for converge-cast networks and prove its optimality in Section III. Then, the negative result for more general networks is reported in Section IV. Simulation results are provided in Section V. In Section VI, we conclude.

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 corresponds to a fixed path, which consists of a subset of the physical links that it transverses. However, for ease of exposition, we will also view a path as a subgraph of G, consisting of the nodes and the edges belong 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's. A network $(G, \mathcal{F})$ is defined as a network topology G equipped with a *path collection* $\mathcal{F}$.

The transmission on one physical link may interfere with other physical links. We consider the following one-hop interference[1] model: each physical link interferes with all other

[1]The results in this paper can also be generalized to the K-hop interference model

physical links that share a common node. This means that among those physical links that are adjacent to a common node, at most one of them can transmit at a time. This model has also been used in [2][6]. Let R_l be the capacity of physical link l, i.e., R_l is the 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.

Consider 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 in 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, then depart from the system. We use $P_f(l_i^f)$ to denote the path starting from link l_i^f and ending at the its destination node. Obviously, $\mathcal{P}_f(l_{n_f}^f) = \mathcal{P}_f$. Note that, it is possible to assign multiple labels to the same physical link if the link is used by more than one flows. 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 $\mathcal{E}_L^{(G,\mathcal{F})}$ the set of all 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 $A_f(t)$ is uniformly bounded by M, i.e., $A_f(t) \leq M$ 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 that the arrival rate vector $\vec{\lambda}$ is strictly inside of 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.

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

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

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

[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, the finite-state irreducible Markov chains satisfy the sample-path large deviations principle. See Section II-E in [3] for related discussion.

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, we want to minimize the steady-state probability that the max-backlog over all flows exceeds a threshold B, i.e.,

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

This is extremely useful in practice. For example, for multiple flow real-time data transmission applications, such as video streaming, visual web conference, e.t.c., it is very important to keep the end-to-end backlog small to meet the harsh delay constraints. Unfortunately, this quantity is in general mathematically intractable. Instead, since the above probability is small, we can instead focus on its asymptotic decay rate when B becomes large. Specifically, we can define the following two quantities:

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

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

Our objective is to find a scheduling algorithm that maximizes I and J.

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

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. To avoid any confusion, we use $G_T(V, \mathcal{E})$ to denote the tree topology, and $\mathcal{F}_T$ to denote the *path collection* for converge-cast. 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 link $l_i^f \in \mathcal{E}_L^{(G_T,\mathcal{F}_T)}$, 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 at link l_i^f for flow f as

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

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 is no remaining link. 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 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 drain packets faster has been reported for a simple linear topology with only one flow [2]. 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 the prior work[2][6].

The LWF algorithm can be formally described in Algorithm 1. In this algorithm, we use $\vec{\gamma(t)} = [\gamma_l(t), l \in \mathcal{E}_L^{(G_T,\mathcal{F}_T)}]$ 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.

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

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

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.

B. Mathematical Preliminaries

1) Capacity Region: For networks with tree topology under the one-hop interference model, the capacity region is in a simple form. Suppose that $l_{i_1}^{f_1}, l_{i_2}^{f_2}, ..., l_{i_{y_v}}^{f_{y_v}}$ are all the logical links adjacent to node v, then the interior of the capacity region can be represented by a set of linear inequalities,

$$\sum_{k=1}^{y_v} \frac{\lambda_{f_k}}{R_{l_{i_k}^{f_k}}} < 1, \text{ for any node } v. \tag{7}$$

Later in this section, we will also be interested in the capacity regions of a subnetwork of $(G_T, \mathcal{F}_T)$. 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. For each flow $f \in \mathcal{A}$, the route in the subnetwork is given by a sub-path 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 *path collection* of this subnetwork 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 $\underset{f\in\mathcal{A}}{\cup} P_f(l_{z_f}^f)$ (Recall that we view $P_f(\cdot)$ as a graph. A finite union of graphs is defined as follows. Given $G_i = \{V_i, E_i\}$, the union of G_i is $\underset{i}{\cup} G_i = (\underset{i}{\cup} V_i, \underset{i}{\cup} E_i)$). In the rest of the paper, we will use $C(\mathcal{A}, z_{\mathcal{A}})$ to denote the capacity region of the above subnetwork and use int$C(\mathcal{A}, z_{\mathcal{A}})$ to denote its interior. Further, we use $\mathcal{E}(\mathcal{A}, z_{\mathcal{A}})$ to denote the set of all logical links of the above subnetwork.

2) Fluid Sample Paths: Given B and T, define the following scaled quantities in the time interval $[-T, 0]$ as

$$a_f^B(t) = \frac{1}{B} \sum_{\tau=0}^{B(T+t)} A_f(\tau), x_i^{f,B}(t) = \frac{1}{B} X_i^f(B(T+t)),$$
$$x_f^B(t) = \frac{1}{B} X_f(B(T+t)), e_i^{f,B}(t) = \frac{1}{B} \sum_{\tau=0}^{B(T+t)} E_i^f(\tau). \tag{8}$$

for $t = \frac{m}{B} - T, m = 0, 1, ..., BT$, and by linear interpolation otherwise. Due to the assumption of bounded arrivals and departures, $(a_f^B(t), x_i^{f,B}(t), x_f^B(t), e_i^{f,B}(t)), f = 1, 2, ..., F, i = 1, 2, ..., n_f$ are all Lipschitz continuous. Fix T and take any sequence of such scaled processes as $B \to \infty$. There must exist a subsequence that converges uniformly over the compact interval $[-T, 0]$. Any such limit is called a fluid sample path(FSP). In other words, $(a_f(t), x_i^f(t), x_f(t), e_i^f(t))$ is called an FSP if there exists a subsequence of $(a_f^B(t), x_i^{f,B}(t), x_f^B(t), e_i^{f,B}(t))$ that converges to it uniformly over $[-T, 0]$. Note that in general, there may exist more than one FSPs out of the same sequence of scaled processes.

Using the scaled quantities, the probability in (3) can be rewritten as $P\left[\max_{f=1,2,...,F} \left\{x_f^B(+\infty)\right\} \geq 1\right]$. Our interest is its decay-rate as $B \to \infty$. Since the arrival process is i.i.d., the scaled arrival process $a^B(t)$ satisfies a large deviation principle with some rate function $I_a^T(\cdot)$ (Chapter 1.2 in [14]). This means that, for any set Γ of arrival sample paths, the probability that $a^B(t)$ falls into Γ satisfies: $\lim_{B\to\infty} \frac{1}{B} \mathbb{P}(a^B(t) \in \Gamma) = -\inf_{a\in\Gamma} I_a^T(a)$. In the typical large-deviation literature, if we can additionally verify that the mapping from $a^B(t)$ to $x_f^B(t)$ is continuous under a given scheduling algorithm,

we can then apply the contraction principle [14] and obtain the decay-rate of the overflow probability by finding the "most likely path to overflow". However, there are significant difficulties in applying this approach in multi-hop network. First, it is usually very hard to verify the continuity of the mapping from $a^B(t)$ to $x_f^B(t)$. Second, finding such "most likely path to overflow" involves solving a high-complexity multi-dimensional calculus-of-variations problem. In this paper, we use a different approach which is first proposed in [3] to circumvent these difficulties. The result of [3] propose a drift minimizing criterion, which is sufficient for an algorithm to attain the largest decay-rate. We will go into details in Section III-C.

Since the convergence to an FSP is uniform, the FSP $(a_f(t), x_i^f(t), x_f(t), e_i^f(t))$ is also Lipschitz continuous, and therefore, its derivative exists almost everywhere (a.e.) over $[-T, 0]$. 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 $\mathcal{Z}$ the set of all time instances where the FSP or the Lyapunov function $V(x(t))$ is not differentiable with respect to t. Then $\mathcal{Z}$ is of measure 0. In the rest of this paper, we will restrict our analysis to those $t \notin \mathcal{Z}$, 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 $\mu_i^f(t) = \frac{d}{dt}e_i^f(t)$. Then, we can derive the following equations for an FSP from equation (1) and (2) (refer to [6] for detailed derivation):

$$\frac{d}{dt}x_i^f(t) = \begin{cases} \mu_{i+1}^f(t) - \mu_i^f(t), & \text{if } i = 1, 2, ..., n_f - 1, \\ \alpha_f(t) - \mu_i^f(t), & \text{if } i = n_f. \end{cases} \tag{9}$$

$$\frac{d}{dt}x_f(t) = \alpha_f(t) - \mu_1^f(t). \tag{10}$$

Eqn. (9) and (10) 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,

$$\frac{d}{dt}V(x(t)) = \max_{f \in \mathcal{M}(t)} \{\alpha_f(t) - \mu_1^f(t)\}. \tag{11}$$

In addition, we have the following lemma that imposes additional constraints for $\mu_1^f(t)$.

Lemma 1. *(Proposition 1 in [6]) Under any algorithms, 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:*

$$\mu_i^f(t) \le \begin{cases} \mu_{i+1}^f(t), & \text{if } i = 1, 2, ..., n_f - 1 \text{ and } x_i^f(t) = 0, \\ \alpha_f(t), & \text{if } i = n_f \text{ and } x_i^f(t) = 0. \end{cases} \tag{12}$$

For each node v, suppose that $l_{i_1}^{f_1}, l_{i_2}^{f_2}, ..., l_{i_{y_v}}^{f_{y_v}}$ are all the links that are adjacent to node v. Then, any FSP must also satisfy the following node constraints:

$$\sum_{k=1}^{y_v} \frac{\mu_{i_k}^{f_k}}{R_{l_{i_k}^{f_k}}} \le 1, \text{ for all nodes } v, \tag{13}$$

$$\mu_i^f \ge 0, \text{ for all } i, f. \tag{14}$$

Lemma 1 can be viewed as follows. The variables $\mu_i^f(t)$ can be viewed as the service rate of link l_i^f. The first part of (12) states that, when the backlog $x_i^f(t)$ is 0, we need the upstream link l_{i+1}^f to serve as many packets as the downstream link l_i^f. A similar interpretation holds for the second part of (12). In (13), $\frac{\mu_i^f(t)}{R_{l_i^f}}$ 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 around each node v.

C. Optimality of the LWF Algorithm

We will use the techniques of [3] 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 among all flows, i.e., for any scheduling algorithm π, we have*

$$\begin{aligned} &\limsup_{B\to\infty} \frac{1}{B} \log \left(\mathbb{P}^{LWF} \left[\max_{f=1,2,...,F} \{X_f(+\infty)\} \ge B \right] \right) \\ \le\ &\liminf_{B\to\infty} \frac{1}{B} \log \left(\mathbb{P}^{\pi} \left[\max_{f=1,2,...,F} \{X_f(+\infty)\} \ge B \right] \right), \end{aligned} \tag{15}$$

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

To prove Theorem 2, we use the result of Theorem 8 from [3], which we restated here for reference.

Theorem 3. *Let π_0 be a scheduling algorithm that satisfies Assumptions 1,2,3,4,5 and 6 of [3]. Then the algorithm π_0 attains the optimal decay-rate in the sense of (15).*

To prove Theorem 2, we need to justify the Assumptions 1,2,3,4,5 and 6 of [3] for the LWF algorithm. Due to space constraints, we list these assumptions (except Assumption 4) in the technical report [10]. Compared to Assumption 4, Assumptions 1,2,3,5,6 are relatively easy to verify. We verify Assumptions 1,2,3,5,6 for the LWF algorithm in our technical report [10]. Next, we will focus on Assumption 4, which is restated below.

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

$$\begin{aligned} \frac{d}{dt}V(x(t)) &= \min_{\tilde{\mu}} \max_{f \in \mathcal{M}(t)} \{\alpha_f(t) - \tilde{\mu}_1^f\} \\ \text{subject to} &\quad \tilde{\mu} \text{ satisfies } (12)(13)(14). \end{aligned} \tag{16}$$

In Eqn. (16), $x(t)$, $\alpha(t)$ are fixed for a given FSP at the time t. $\tilde{\mu} = [\tilde{\mu}_i^f]$ represents a feasible scheduling decision (which is not necessarily the same as the decision according to the FSP). Hence, $\max_{f \in \mathcal{M}(t)} \{\alpha_f(t) - \tilde{\mu}_1^f\}$ is the drift of the Lyapunov function if the scheduling decision were $\tilde{\mu}$. By taking the minimization over all feasible $\tilde{\mu}$'s, (16) essentially states that

the drift of the Lyapunov function at each time under algorithm π_0 is also the minimum possible.

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

Since we have verified that the LWF algorithm satisfies 1,2,3,5 and 6 in our technical report, then Theorem 2 follows directly from Theorem 4. Therefore, It is sufficient to prove Theorem 4. To begin with, we need to understand what should be the optimal solution of (16).

1) Optimal Solution of (16): Clearly, to minimize the objective function in (16), we only need to serve flows in $\mathcal{M}(t)$ as much as possible. Note that in the optimization problem (16), minimizing the objective is equivalent to maximizing $\tilde{\mu}_1^f(t)$ for all $f \in \mathcal{M}(t)$ since α_f is fixed by the given FSP. From constraint (12), we note that there is no constraint relating $\tilde{\mu}_i^f$ and $\tilde{\mu}_j^f, j > i$, if $x_i^f(t) > 0$. An intuitive explanation is the following. $x_i^f(t) > 0$ means that the queue length in the original discrete time system is very large (the queue length is approximately $Bx_i^f(t)$, and B is large). Therefore, link l_i^f always has enough packets to transmit. In contrast, in case of $x_i^f(t) = 0$, we need link l_{i+1}^f to serve as many packets as link l_i^f. Hence, we can intuitively view the link that has non-zero backlog in the FSP and that is closest to the destination of flow f as a "barrier" for flow f. To achieve the optimal solution of (16), we only need to consider those links between the destination and the barrier. To be specific, for each flow $f \in \mathcal{M}(t)$, define the barrier of flow f as

$$b_f(t) = \begin{cases} \underset{i}{\text{argmin}}\{i | x_i^f(t) > 0\}, & \text{if } x_f(t) > 0, \\ n_f, & \text{if } x_f(t) = 0. \end{cases} \tag{17}$$

Given $\mathcal{A} \subseteq \mathcal{M}(t)$, let $b_{\mathcal{A}}(t) = [b_f(t), f \in \mathcal{A}]$. Note that if the growth rate of the end-to-end backlog of flow f is g, we must have $\tilde{\mu}_1^f = \alpha_f(t) - g$. Further, all $\tilde{\mu}_i^f$ for $i \leq b_f$ must be no less than μ_1^f due to constraints in (12). Hence, $[(\alpha_f(t) - g), f \in \mathcal{A}]$ must be supportable by the subnetwork with topology $\underset{f\in\mathcal{A}}{\cup} P_f(l_{b_f}^f)$. Therefore, we can show that the following value $g_{\mathcal{A}}(t)$ gives the minimal possible growth rate of the maximum backlog among all flows $f \in \mathcal{A}$. To be precise, if $x_f(t) > 0$ for some $f \in \mathcal{A}$, then

$$g_{\mathcal{A}}(t) \triangleq \inf\left\{g | \left((\alpha_f(t) - g)^+\right)_{f\in\mathcal{A}} \in C(\mathcal{A}, b_{\mathcal{A}}(t))\right\}.$$

Otherwise, if $x_f(t) = 0$ for all $f \in \mathcal{A}$, then

$$g_{\mathcal{A}}(t) \triangleq \inf\left\{g \geq 0 | \left((\alpha_f(t) - g)^+\right)_{f\in\mathcal{A}} \in C(\mathcal{A}, b_{\mathcal{A}}(t))\right\}. \tag{18}$$

Here, $(u)^+ \triangleq \max\{u, 0\}$.

Consider $g_{\mathcal{M}(t)}(t)$ for $\mathcal{M}(t)$. For simplicity, we use the notation $g(t)$ to stand for $g_{\mathcal{M}(t)}(t)$. The following lemma states that $g(t)$ attains the optimality in (16). See our technical report[10] for the detailed proof.

Lemma 5. *$g(t)$ is the optimal solution of the optimization problem (16), i.e.,*

$$\min_{\substack{\tilde{\mu} \text{ satisfies} \\ (12)(13)(14)}} \max_{f\in\mathcal{M}(t)} \{\alpha_f(t) - \tilde{\mu}_1^f(t)\} = g(t).$$

2) Proof of the Optimality of LWF Algorithm: Next, we will show that the LWF algorithm achieves the minimum drift $g(t)$. The key difficulty is that the LWF algorithm does not know the value of $\alpha(t)$ before hand. However, by excavating some inherent properties of the LWF algorithm, we could show that the LWF algorithm always achieve the optimality.

Given an FSP, assume that $\mu(t)$ is the service rate under the LWF algorithm. Denote by $\mathcal{M}_0(t)$ the set of flows in $\mathcal{M}(t)$ that have the maximum growth rate, i.e.,

$$\mathcal{M}_0(t) = \left\{f \in \mathcal{M}(t) | \frac{d}{dt}x_f(t) = \max_{f'\in\mathcal{M}(t)} \left\{\frac{d}{dt}x_{f'}(t)\right\}\right\},$$

where $\frac{d}{dt}x_f(t) = \alpha_f(t) - \mu_1^f(t)$.

Then, all flows in $\mathcal{M}_0(t)$ have the same end-to-end backlog and the same derivative in the corresponding FSP. If $x_f(t) = \frac{d}{dt}x_f(t) = 0$ for one flow $f \in \mathcal{M}_0(t)$, then $x_f(t) = \frac{d}{dt}x_f(t) = 0$ holds for all flows. In this case, the max growth rate under the LWF algorithm is 0. Note that $g(t) \leq 0$ because of Lemma 5. However, $g(t) \geq 0$ since the backlog is zero (see (18)). Hence, $g(t) = 0$ and Theorem 4 holds trivially. Hence, we next focus on the case $x_f(t) > 0$ or $\frac{d}{dt}x_f(t) > 0$ for all $f \in \mathcal{M}_0(t)$.

Consider $g_{\mathcal{M}_0(t)}(t)$ for $\mathcal{M}_0(t)$. Again, for simplicity, we use $g_0(t)$ to stand for $g_{\mathcal{M}_0(t)}(t)$. It is easy to check that $g_0(t) \leq g(t)$.

Recall we have defined the barrier for flow $f \in \mathcal{M}(t)$. Now we need to introduce the concept of *potential barrier* $b'_f(t)$ for $f \in \mathcal{M}_0(t)$. $b'_f(t)$ is the first link that has the potential to become a barrier for flow $f \in \mathcal{M}_0(t)$, which is formally defined as follows:

$$b'_f(t) = \underset{i}{\text{argmin}}\{i | x_i^f(t) > 0 \text{ or } \frac{d}{dt}x_i^f(t) > 0\}. \tag{19}$$

Since we already assume that $x_f(t) > 0$ or $\frac{d}{dt}x_f(t) > 0$ for each flow $f \in \mathcal{M}_0(t)$, it is easy to check that the potential barrier is well defined. Further, let $b'_{\mathcal{M}_0} = [b'_f, f \in \mathcal{M}_0]$.

We consider the FSP together with the original discrete-time system. Let $(a_f^B(t), x_i^{\tilde{f},B}(t), x_f^B(t), e_i^{f,B}(t))$ be the sequence of scaled processes that converge uniformly to $(a_f(t), x_i^f(t), x_f(t), e_i^f(t))$. We then find the following result which is the key in the proof of the optimality of the LWF algorithm.

Lemma 6. *For any $0 < \epsilon < 1$, there exists $\tilde{\delta}(\epsilon) > 0$. For any fixed $0 < \delta < \frac{\tilde{\delta}(\epsilon)}{2}$, there exist $\tilde{B}(\delta) > 0$, such that for all $B > \tilde{B}(\delta)$ and all time slots $t_0 \in (B(t+T+\delta), B(t+T+2\delta))$, we have*

1) *The weight of any logical link $l_{\tilde{i}}^{\tilde{f}} \notin \mathcal{E}(\mathcal{M}_0, b'_{\mathcal{M}_0})$ is strictly smaller than the weight of any logical link $l_{b'_f}^f, f \in \mathcal{M}_0(t)$, i.e., $q_{\tilde{i}}^{\tilde{f}}(t_0) < q_{b'_f}^f(t_0)$.*
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 the logical links in $\mathcal{E}(\mathcal{M}_0, b'_{\mathcal{M}_0})$ that interfere with link $l^f_{b'_f}$. Then at least one link in $\mathcal{I}'(l^f_{b'_f}) \bigcup \{l^f_{b'_f}\}$ should be scheduled at time instance t_0. Moreover, if all the links in $\mathcal{I}'(l^f_{b'_f})$ interfere with each other, then the usage efficient of any link $l^{\hat{f}}_{\hat{i}} \in \mathcal{I}'(l^f_{b'_f}) \bigcup \{l^f_{b'_f}\}$(if it is scheduled) must satisfy $\eta^{\hat{f}}_{\hat{i}}(t_0) > 1-\epsilon$.

The rigorous proof of Lemma 6 has to deal with the original discrete-time system, which is shown in our technical report [10]. However, it is easier to explain the intuition in the sense of FSP. In the immediate future of time t, those potential 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 potential barrier links is approximately equal to the maximum end-to-end backlog. For those logical links not in $\mathcal{E}(\mathcal{M}_0, b'_{\mathcal{M}_0})$, it is easy to check that their weights are strictly smaller than the maximum end-to-end backlog. Because, either the corresponding flow is not in $\mathcal{M}_0$, or it is in $\mathcal{M}_0$, but at least the backlog of the potential barrier link must be subtracted from the weight of the link. Therefore, they must have smaller weights than those potential 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 $\mathcal{E}(\mathcal{M}_0, b'_{\mathcal{M}_0})$. 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 weight of link $l^{\hat{f}}_{\hat{i}}$ and $l^f_{b'_f}$ 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.

Now we are ready to prove Theorem 4. By contradiction, suppose that the LWF algorithm does not minimize the drift. Then, we must have $\alpha_f(t) - \hat{\mu}^f_1(t) = \max_{f' \in \mathcal{M}(t)} \{\alpha_{f'}(t) - \hat{\mu}^{f'}_1(t)\} > g(t) \geq g_0(t)$, and thus, $\alpha_f(t) - g_0(t) > \hat{\mu}^f_1(t)$ for any $f \in \mathcal{M}_0(t)$. From the definition of $g_0(t)$, we can then obtain $\left(\mu^f_1(t), f \in \mathcal{M}_0(t)\right) \in \text{int}\, C(\mathcal{M}_0(t), b_{\mathcal{M}_0(t)}(t))$. Noting that $b'_f(t) \leq b_f(t)$, it is easy to check from the definition that $\text{int}\, C(\mathcal{M}_0(t), b_{\mathcal{M}_0(t)}(t)) \subseteq \text{int}\, C(\mathcal{M}_0(t), b'_{\mathcal{M}_0(t)}(t))$. Therefore, $\left(\mu^f_1(t), f \in \mathcal{M}_0(t)\right) \in \text{int}\, C(\mathcal{M}_0(t), b'_{\mathcal{M}_0(t)}(t))$. Consider a leaf link $l^{f^*}_{i^*}$ in the subtree $\bigcup_{f \in \mathcal{M}_0} P_f(l^f_{b'_f(t)})$. Let all of its interfering links in $\mathcal{E}(\mathcal{M}_0, b'_{\mathcal{M}_0}(t))$ be $l^{f_1}_{i_1}, l^{f_2}_{i_2}, ..., l^{f_y}_{i_y}$. Then according to Eqn. (7), there exist $\epsilon > 0$ such that

$$\frac{\mu^{f^*}_1}{R_{l^{f^*}_{i^*}}} + \sum_{k=1}^{y} \frac{\mu^{f_k}_1}{R_{l^{f_k}_{i_k}}} < 1 - \epsilon. \tag{20}$$

For the above ϵ, according to Lemma 6, we know that for $0 < \delta < \frac{\tilde{\delta}(\epsilon)}{2}$ and $B > \tilde{B}(\delta) > 0$, in the time period $(B(t+T+\delta), B(t+T+2\delta))$, 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. On the other hand, these links interfere with each other. Then, exactly one of these links must be scheduled at one time instance. Further, their usage efficiency is larger than $1-\epsilon$. Hence, we can show that(refer to [10] for details)

$$\frac{\mu^{f^*}_{i^*}}{R_{l^{f^*}_{i^*}}} + \sum_{k=1}^{y} \frac{\mu^{f_k}_{i_k}}{R_{l^{f_k}_{i_k}}} \geq 1 - \epsilon. \tag{21}$$

Finally, since the derivatives of the backlog of links (except potential barrier links) in $\mathcal{E}(\mathcal{M}_0, b'_{\mathcal{M}_0})$ stay at zero, we have $\mu^{f^*}_1 = \mu^{f^*}_{i^*}, \mu^{f_k}_1 = \mu^{f_k}_{i_k}, k = 1, 2, ..., y$. Then, Eqn. (21) contradicts to Eqn. (20).

Therefore, the LWF algorithm satisfies the drift minimization assumption, and thus achieves the largest decay rate for the overflow probability of the maximum end-to-end backlog.

IV. Drift Minimization in General Networks

In the previous section, we establish the large-deviations optimality of LWF in minimizing the maximum end-to-end backlog for converge-cast by showing that LWF satisfies the drift minimization assumption. Naturally, our next question is that whether we can develop drift-minimizing algorithm (similar to LWF) that is optimal for more general network topologies and traffic profiles. In this direction, we will show some negative, but fundamental, results. First, we can show that LWF is not drift minimizing even for a tree topology that is not a converge-cast (like Fig. 1. Refer to our technical report [10] for more details). Then, a natural question is whether there exists other algorithms that can minimize the drift for general networks. Unfortunately, we will prove next that no algorithm could minimize the drift in the network show in Fig. 1. Hence, in order to design optimal scheduling algorithms, we must find new criterions (other than drift minimizing).

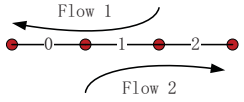


Fig. 1. A counter example of LWF algorithm.

The network shown in Fig. 1 is a tandem 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. We have the following result.

Theorem 7. *For the network in Fig. 1, no algorithm can minimize the drift in every regular time for every FSP.*

We note that Theorem 7 is quite strong. It holds even when we include the possibility of non-causal algorithms. The basic idea of proving Theorem 7 is to construct a sequence of FSPs,

and then find a contradiction if we want to minimize the drift for every FSP. To proceed, we need the following lemma.

Lemma 8. *Given any FSP for the network shown in Fig. 1. Let* $a(t) = \begin{bmatrix} a_1(t) & a_2(t) \end{bmatrix}^T$ *denote the arrival process in FSP and let* $\alpha(t) = \frac{d}{dt}a(t)$. *Assume that* $x_i^f(0) = 0, f = 1, 2, i = 1, 2$ *and that 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* $[0, T]$.

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, then there exists service rate μ satisfying constraints (12)(13)(14), such that $\alpha_1(t) - \mu_1^1(t) = \alpha_2(t) - \mu_1^2(t)$. Under this assumption, the drift-minimizing FSP will be able to balance the end-to-end backlog for the two flows. The detailed proof is shown in our technical report [10].

Now we are ready to prove Theorem 7.

Proof: We prove by contradiction. Assume that there exists an algorithm π that minimizes the drift for the above network at every regular time in every FSP.

The actual arrival process is a random process defined on space $\Omega \times \mathbb{N}$. We represent it by $A(\omega, t) = \begin{bmatrix} A_1(\omega, t) \\ A_2(\omega, t) \end{bmatrix}$. For each $\omega \in \Omega$, $A(\omega) = \{A(\omega, t), t = 0, 1, 2, ...\}$ can be seen as a $2 \times \infty$ matrix.

For convenience, we define two matrices M_1 and M_2.

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

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

Using these two matrices, we construct a sequence of realizations of the arrival process as follows, where the k-th $(k = 1, 2, ...)$ realization ω_k is given by

$$A(\omega_k) = [\underbrace{M_1 M_1 \cdots M_1}_{k} \underbrace{M_2 M_2 \cdots M_2}_{k} \underbrace{M_1 M_1 \cdots M_1}_{k} \cdots].$$

We assume that there are no packets in the network at the initial time($t = 0$). Next, we define scaled versions of the system based on Eqn. (8). Here, we let $T = 16$. Define a matrix $B = \{B_{k,j}\}, k = 1, 2, ..., j = 1, 2, ...$, where $B_{k,j} = kj$. It is easy to check that, for any fixed j, as $k \to \infty$, $a^{B_{k,j}}(\omega_k, t)$ converge uniformly to a piecewise linear function $a^{(j)}(t) = \begin{bmatrix} a_1^{(j)}(t) & a_2^{(j)}(t) \end{bmatrix}$, and it is easy to check that (refer to [10] for details)

$$\alpha_1^{(j)}(t) = \frac{d}{dt}a_1^{(j)}(t) = \begin{cases} \frac{1}{2}, & \text{if } \frac{Ti}{j} \leq t < \frac{T(2i+1)}{2j}, \\ \frac{1}{8}, & \text{if } \frac{T(2i+1)}{2j} \leq t < \frac{T(i+1)}{j}, \end{cases}$$

$$\alpha_2^{(j)}(t) = \frac{d}{dt}a_2^{(j)}(t) = \begin{cases} \frac{1}{8}, & \text{if } \frac{Ti}{j} \leq t < \frac{T(2i+1)}{2j}, \\ \frac{1}{2}, & \text{if } \frac{T(2i+1)}{2j} \leq t < \frac{T(i+1)}{j}, \end{cases}$$

where $i = 0, 1, ..., j - 1$.

Note that this FSP satisfies the conditions in Lemma 8. Therefore, $x_2^1(t) = x_2^2(t)$ and $x_1^1(t) = x_1^2(t) = 0$ in $[0, T]$ under algorithm π. By solving Eqn. (16), we can infer the queue evolution of the drift-minimizing FSP as $x_1^{(j)}(T) = x_2^{(j)}(T) = \frac{1}{24}T$ (refer to [10] for details). Note that $\lim_{k\to\infty} x_1^{B_{k,j}}(\omega_k, T) = x_1^{(j)}(T)$. Therefore, for any j, there exists k_j, such that

$$x_1^{B_{k_j,j}}(\omega_{k_j}, T) > \frac{1}{25}T. \tag{22}$$

Now, we consider another sequence of scaled arrival processes $a^{B_{k_j,j}}(\omega_{k_j}, t)$, where k_j is chosen according to the value that leads to Eqn. (22). Let $a^{(*)}(t) = \begin{bmatrix} a_1^{(*)}(t) & a_2^{(*)}(t) \end{bmatrix}$, where $a_1^{(*)}(t) = a_2^{(*)}(t) = \frac{5}{16}t$.

We can show that $a^{B_{k_j,j}}(\omega_{k_j}, t)$ converges uniformly to $a^{(*)}(t)$ as $j \to \infty$ (refer to [10] for details) such that

$$\alpha_1^{(*)}(t) = \frac{d}{dt}a_1^{(*)}(t) = \frac{5}{16}, \alpha_2^{(*)}(t) = \frac{d}{dt}a_2^{(*)}(t) = \frac{5}{16}.$$

For this FSP, drift-minimization would lead to $x_1^{(*)}(T) = x_2^{(*)}(T) = 0$ (refer to [10] for details). Note that $\lim_{j\to\infty} x_1^{B_{k_j,j}}(\omega_{k_j}, T) = x_1^{(*)}(T) = 0$. Therefore, there exists J, such that $x_1^{B_{k_J,J}}(\omega_{k_J}, T) < \frac{1}{25}T$.

This contradicts to Eqn. (22). Hence, the result of the theorem must hold. ■

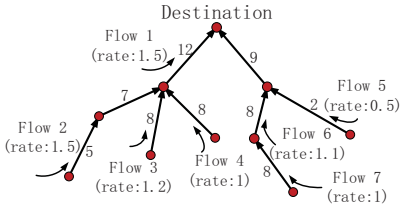
V. Simulation

In this section, we present our simulation results for the topology shown in Fig. 2.(a). This topology contains 10 nodes, with one node as the root, 2 nodes at depth 1, 5 nodes at depth 2, and 2 nodes at depth 3. The number near each link represents its capacity. There are 7 flows in the network. The number of packets arrived at each flow per time slot admits to Poisson distribution. The number near each flow indicates the arrival rate. One can verify that the resulting offered-load vector has already exceeded 0.9 of the optimal capacity region according to Eqn. (7).

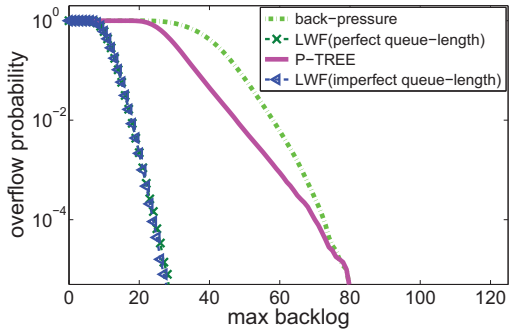
We are interested in the overflow probability of the maximum end-to-end backlog, i.e., $P\left[\max_{f=1,2,...,7}\{X_f(t)\} \geq B\right]$. We simulate the system under three different scheduling algorithms: LWF, back-pressure, and P-TREE.

We give a brief overview about these algorithms. In back-pressure algorithm[1], 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 $X_0^f(t) = 0$). And then we schedule those logical links that maximizes the total weight. As for the P-TREE algorithm [6], 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.

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



(a) Network topology.



(b) Simulation result.

Fig. 2. Comparison between LWF, back-pressure, and P-TREE algorithms.

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 is based on the queue-length information r_l time slots before, where $r = [r_l]$ is a set of random variables uniformly distributed in [0,20].

In Fig. 2.(b), we plot the overflow probability $P\left[\max_{f=1,2,\ldots,7}\{X_f(t)\} \geq B\right]$ vs 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 performance of P-TREE algorithm and back-pressure algorithm are both significantly worse. This is because, in our network setting, while flow 1 and flow 2 have the same rate, they are at different depth: flow 1 is at depth 1, and flow 2 is at depth 3. In P-TREE algorithm, priority is given to flow 1 as it is closer to 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 priority. For both P-TREE and back-pressure, giving priority to flow 1 hurts the packet transmission of flow 2. Therefore, their performance are poor in terms of the maximum end-to-end backlog. Finally, note that the LWF algorithm with imperfect queue-length information still performs very well. This indicates that although our theoretical analysis is based on the assumption of perfect queue-length information, the insights of the algorithm obtained can still be effective even in practical settings with imperfect queue-length information. Additional simulation results (including a comparison with [13]) is available in our technical report [10].

VI. Conclusion

We study the scheduling problem for multi-hop wireless network under one-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.

Acknowledgement: The work has been partially supported by NSF through grants CNS-0643145, CNS-0721477 and CNS-0721484, and by the Purdue Research Foundation.

References

[1] L. Tassiulas and A. Ephremides, "*Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks*," IEEE Transactions on Automatic Control, vol. 37, no. 12, pp. 1936-1949, December 1992.

[2] 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, 1993.

[3] V.J.Venkataramanan and X. Lin, "*On the Queue-Overflow Probability of Wireless Systems: A New Approach Combining Large Deviation with Lyapnov Functions*," submitted to IEEE Trans. Info. Theory, 2009. [online]. Available: https://engineering.purdue.edu/%7elinx/publications.html.

[4] C. Joo, X. Lin, and N. B. Shroff, "*Understanding the Capacity Region of the Greedy Maximal Scheduling Algorithm in Multi-hop Wireless Networks*," IEEE/ACM Transactions on Networking, vol. 17, no. 4, pp. 1132-1145, August 2009.

[5] S. Hariharan and N. B. Shroff, "*On Optimal Dynamic Scheduling for Sum-Queue Minimization in Trees*," IEEE WIOPT, May 2011

[6] V. J. Venkataramanan and X. Lin, "*Low-Complexity Scheduling Algorithm for Sum-Queue Minimization in Wireless Convergecast*," in IEEE INFOCOM, Shanghai, China, April 2011.

[7] 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, April 2009.

[8] 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, November 2006.

[9] V. J. Venkataramanan and X. Lin, "*On Wireless Scheduling Algorithms for Minimizing the Queue-Overflow Probability*," IEEE Transactions on Networking, vol. 18, no. 3, June 2010.

[10] S. Zhao and X. Lin, "*Scheduling Algorithm for End-to-End Backlog Minimization in Wireless Multi-hop Networks*," Technical Report, Purdue University, 2011. [Online]. Available: http://web.ics.purdue.edu/%7ezhao147/.

[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, Sept. 2009.

[12] M. J. Neely, "*Delay Analysis for Maximal Scheduling with Flow Control in Wireless Networks with Bursty Traffic*," IEEE Transactions on Networking, vol. 17, no. 4, pp. 1146-1159, August 2009.

[13] 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.

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