Low-Complexity Scheduling Algorithm for Sum-Queue Minimization in Wireless Convergecast

V. J. Venkataramanan and Xiaojun Lin
School of Electrical and Computer Engineering, Purdue University, West Lafayette
Email: {vvenkat,linx}@purdue.edu

***Abstract*—We consider the problem of link scheduling for efficient convergecast in a wireless system. While there have been many results on scheduling algorithms that attain the maximum possible throughput in such a system, there have been few results that provide scheduling algorithms that are optimal in terms of some quality-of-service metric such as the probability that the end-to-end buffer usage exceeds a large threshold. Using a large deviations framework, we design a novel and low complexity algorithm that attains the optimal asymptotic decay rate for the overflow probability of the sum-queue (i.e. the total queue backlog in the entire system) as the overflow threshold becomes large. Simulations show that this algorithm has better performance than well known algorithms such as the standard back-pressure algorithm and the multihop version of greedy maximal matching (combined with back-pressure). Our proposed algorithm performs better not only in terms of the asymptotic decay rate at large overflow thresholds, but also in terms of the actual probability of overflow for practical range of overflow thresholds.**

I. Introduction

We consider the link scheduling problem in a wireless multihop network for convergecast. In a convergecast situation, there is a central node to which each node in the network forwards data. We assume that each node uses a fixed path to route data to the central node. The same path is used irrespective of whether a node is relaying data for other nodes or is transmitting its own data. This scenario results in a tree topology for the overall system of flows. Such convergecast problems arise commonly in sensor networks where there is a central node that collects data, such as temperature readings or audio/video signals, from all other nodes in the network.

Since the communication medium is wireless, managing interference becomes important in order to achieve effective data transfer. For example, allowing all nodes to transmit at the same time would lead to a significant number of collisions whereas allowing only one node to talk at a time is inefficient. Scheduling algorithms play the crucial role of determining which links can be activated at a certain time. A first order requirement made on any desirable scheduling algorithm is that of throughput optimality. That is, if the desired scheduling algorithm can not stabilize the system for an offered load, then no other algorithm can stabilize the system for the same offered load. There has been a substantial amount of work in this area of research starting with the seminal work of Tassiulas and Ephremides [1]. The backpressure algorithm [1], [2], exponential rule [3], log rule [4], [5] and α-algorithms [6] are several throughput optimal algorithms known to date.

This work was supported in part by NSF awards CNS-0643145, CNS-0721477, CNS-0721484, and a grant from Purdue Research Foundation.

However, throughput-optimality alone is not sufficient when performance metrics such as mean-delay, probability of delay violation or probability of buffer overflow are considered. For example, the well known throughput-optimal back pressure algorithm suffers from large delays [7], [8]. Many throughput-optimal algorithms make their scheduling decisions based on the backlog in the system, which in turn depends on past scheduling decisions and arrival rates. Such cross-dependency results in system dynamics that are difficult to analyze. Due to this reason, the behavior of throughput-optimal algorithms in terms of finer QoS performance metrics, such as those mentioned before, is difficult to quantify. For *single-hop* traffic, several techniques have been used to characterize such finer QoS metrics, including mean-delay analysis [9], [10], heavy traffic analysis [3], [11] and large deviations [6], [12]–[15]. For *multi-hop* traffic, however, the problem becomes even more complex due to the coupling between the departure process of a node and the arrival process of the downstream node. Hence, the available results are even more limited. [16], [17] study mean-delay performance in the presence of multi-hop traffic. While [16] provides lower bounds on the mean-delay, it does not immediately reveal which algorithm is optimal. [17] provides an algorithm that is order-optimal for mean-delay. While the algorithm achieves the optimal order when the number of nodes is large, for small or moderate size systems, the algorithm may not be close to optimal. [18] studies a specific tandem network topology with a single flow and proposes an algorithm that is sample path optimal for the sum-queue (i.e., the total backlog of the single flow).

Instead of analyzing the mean-delay performance as in [16], [17], we seek to design a scheduling algorithm for the tree topology to minimize the probability that the sum-queue of the system (i.e., the total backlog of all flows) exceeds a large threshold. However, it appears difficult to apply the technique of [18] in this more general tree topology and to derive the exact probability for sum-queue overflow. Hence, we employ a large deviations approximation which captures the asymptotic rate of decay of this probability as the overflow threshold increases [13].

Specifically, we focus on the so-called one-hop interference model (as in [19] and [20]) and design a low-complexity scheduling algorithm called P-TREE that attains the maximum rate of decay for the probability of overflow of the sum backlog. (Note that although we do not explicitly prove the throughput-optimality of the P-TREE algorithm, it can also be established through the mathematical techniques used in this paper.) The philosophy behind the P-TREE algorithm is to ensure that as much data is driven out of the system as soon as possible. The algorithm achieves this by giving priority to links that are closer to the destination node and to links that have a larger capacity. The details of this algorithm are provided in Section IV. A non-rigorous explanation of the P-TREE algorithm is the following. The algorithm considers for activation all the links attached to the root, then it considers all the links at depth 2 from the root, and so on, till it has reached the deepest leaf nodes. At each depth, the algorithm first eliminates from consideration all links that might interfere with links already activated at a lower depth. Then, from the remaining links, the algorithm activates those links which will lead to a maximum net transfer of data across that depth. When applied to the tandem topology studied in [18], our algorithm reduces to the algorithm used in [18]. However, we emphasize that our algorithm works for the more general tree topology.

Simulation results show that the algorithm significantly reduces the probability of sum-queue overflow even when the overflow threshold is not very large. It performs much better than both the backpressure algorithm and the multi-hop version of the low-complexity greedy maximal matching (GMM) algorithm. Take L to be the number of links in the system. For the scenario we consider (1-hop interference), it is known that the back-pressure algorithm has a complexity of $O(L^3)$ [21] and the greedy maximal matching algorithm has a complexity of $O(L \log L)$ [19]. In comparison, the P-TREE algorithm has an even lower complexity of $O(L)$.

The large-deviations optimality of the P-TREE algorithm is based on a result from our earlier work that, under suitable assumptions, an algorithm that minimizes the drift of a Lyapunov function at every time in every fluid sample path is large-deviations optimal for minimizing the probability that the Lyapunov function overflows [13]. However, as we will discuss later, it is not trivial to come up with the P-TREE algorithm and to verify that it minimizes the drift of the sum-queue in every fluid sample paths. As readers will see in Section V, such verification involves novel techniques that uncover non-trivial insights on the dynamics of the P-TREE algorithm. These techniques are of independent interest and may be useful for other settings as well.

The rest of the paper is organized as follows. Section II defines the system model and Section III introduces the large deviations preliminaries. Section IV presents the P-TREE scheduling algorithm, which is shown to be decay-rate optimal for the sum-queue overflow probability in Section V. Simulation results are provided in Section VI. Then we conclude.

II. System Model

As mentioned in the introduction, the convergecast problem leads to a tree topology for the flows in the network. The root of the tree is the destination node for all flows in the network. Each flow in the network originates at some node of the tree (other than the root) and follows the shortest path to the root. There can be only one flow originating at each node. All nodes in the tree other than the root and leaf nodes will be referred to as interior nodes. Each node (except the root) in the network can be associated with a link that connects the node to its parent node. Since this association is a one-to-one mapping, we will use a unique identifier $\boldsymbol{l}$ to refer to both the link and its corresponding node. We will use $C(\boldsymbol{l})$ to denote the number of children of node $\boldsymbol{l}$ and C to denote the number of children of the root. Let $\mathcal{L}$ denote the set of all links/nodes in the network.

For ease of exposition, we use a vector $\boldsymbol{l}$ to identify a link (node), which can be explained through the following recursive procedure. Consider that we have labelled a node as a vector $\boldsymbol{l} = (l_1, \ldots, l_{D(\boldsymbol{l})})$ (with $D(\boldsymbol{l})$ denoting the dimension of $\boldsymbol{l}$). Then, the task of labelling its children nodes is accomplished as follows. The children nodes are ordered according to their link capacities. The one with the highest link capacity is labelled $(l_1, \ldots, l_{D(\boldsymbol{l})}, 1)$, the next is labelled $(l_1, \ldots, l_{D(\boldsymbol{l})}, 2)$ and so on. In the future, we will use the notation $< \boldsymbol{l}, i >$ to denote the vector $(l_1, \ldots, l_{D(\boldsymbol{l})}, i)$ and $< \boldsymbol{l}, i, j >$ to denote $<< \boldsymbol{l}, i >, j >$. To start off this procedure, we label the root node with a null vector. Hence, the vectors of dimension one, i.e., $1, 2, \ldots, C$, represent the nodes at depth 1 arranged in decreasing order of their link capacities. Please see Fig. 1 for an example of the labeling scheme.

The interference model that we consider is the so-called one-hop interference model [19], [20]. This means that a node can either receive or transmit during a time-slot but not both. Further, it can only receive from one of its children nodes at a time. This interference model has been used to study Bluetooth, UWB, and FH-CDMA systems [19], [22], [23]. We assume that time is slotted and the link capacity is fixed at all time. Let $F_{\boldsymbol{l}}$ denote the capacity of link $\boldsymbol{l}$, i.e., $F_{\boldsymbol{l}}$ is the amount of data that can be transmitted over link $\boldsymbol{l}$ in a time-slot provided that interfering links are silent.

The queue associated with link $\boldsymbol{l}$, denoted by $X_{\boldsymbol{l}}(t)$, is maintained by node $\boldsymbol{l}$. Let $E_{\boldsymbol{l}}(t)$ denote the amount of data transmitted over link $\boldsymbol{l}$ in time-slot t. We impose the constraint that $E_{\boldsymbol{l}}(t) \leq X_{\boldsymbol{l}}(t)$. Let $A_{\boldsymbol{l}}(t)$ denote the amount of data generated by node $\boldsymbol{l}$ in time-slot t. We assume that $A_{\boldsymbol{l}}(t)$ is *i.i.d.* in time[*] and that there is a bound M on the maximum amount of data that any node can generate in a time-slot. Let $\hat{\lambda}_{\boldsymbol{l}} \triangleq \mathbf{E}[A_{\boldsymbol{l}}(t)]$ be the expected arrival rate. We assume that $\hat{\boldsymbol{\lambda}}$ is such that the system is stabilizable, i.e., there exists some scheduling algorithm that can stabilize the system. The queue

[*]This assumption can be relaxed. It suffices that $A_{\boldsymbol{l}}(t)$ satisfies a sample path LDP [13].

evolution is then as follows

$$
\begin{aligned}
X_{\boldsymbol{l}}(t+1) &= X_{\boldsymbol{l}}(t) + A_{\boldsymbol{l}}(t) + \sum_{l=1}^{C(\boldsymbol{l})} E_{<\boldsymbol{l},l>}(t) - E_{\boldsymbol{l}}(t) \\
&\quad \text{if } \boldsymbol{l} \text{ is not a leaf} \\
X_{\boldsymbol{l}}(t+1) &= X_{\boldsymbol{l}}(t) + A_{\boldsymbol{l}}(t) - E_{\boldsymbol{l}}(t) \text{ if } \boldsymbol{l} \text{ is a leaf.} \qquad (1)
\end{aligned}
$$

The root maintains no queue since it is the destination node for all flows. Further, $X_{\boldsymbol{l}}(t) \geq 0$ for all t and links $\boldsymbol{l}$. From (1), we can derive

$$
\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(t+1) = \sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(t) + \sum_{\boldsymbol{l}\in\mathcal{L}} A_{\boldsymbol{l}}(t) - \sum_{l=1}^{C} E_{l}(t). \qquad (2)
$$

Equation (2) states that the sum queue is governed by a simple queueing equation where the arrival is the sum of the arrivals at each node in the tree and the service is the sum of service given to the links connected to the root. Note that the service given to any other link in the system will not change the sum queue since it is simply an internal transfer of data.

A. Performance Objective

In this paper, we are interested in designing a scheduling algorithm to minimize the total buffer occupancy in the network in the following sense. We want to minimize the steady-state probability that the total buffer occupancy exceeds a threshold B. The precise mathematical quantity that we want to minimize is given by

$$
\mathbf{P}\left[\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(+\infty) \geq B\right]. \qquad (3)
$$

In general this quantity is mathematically intractable. We instead use the following large deviations quantities

$$
\begin{aligned}
-I &\triangleq \liminf_{B\to\infty} \frac{1}{B} \log\left(\mathbf{P}\left[\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(+\infty) \geq B\right]\right) \qquad (4) \\
-J &\triangleq \limsup_{B\to\infty} \frac{1}{B} \log\left(\mathbf{P}\left[\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(+\infty) \geq B\right]\right) \qquad (5)
\end{aligned}
$$

to provide an approximation of (3). Note that for large B, we have $e^{-IB+o(B)} \leq \mathbf{P}\left[\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(+\infty) \geq B\right] \leq e^{-JB+o(B)}$.

The quantities I and J can be determined by the so-called fluid-sample-paths (FSPs) [13] described next.

III. Large Deviations Preliminaries

We first define the concept of fluid sample paths.

A. Fluid Sample Paths

For a fixed B and T, define the following scaled quantities in the time interval $[-T, +\infty)$:

$$
a_{\boldsymbol{l}}^{B}(t) = \frac{1}{B}\sum_{\tau=0}^{B(T+t)} A_{\boldsymbol{l}}(\tau), \quad x_{\boldsymbol{l}}^{B}(t) = \frac{1}{B} X_{\boldsymbol{l}}(B(T+t)),
$$
$$
e_{\boldsymbol{l}}^{B}(t) = \frac{1}{B}\sum_{\tau=0}^{B(T+t)} E_{\boldsymbol{l}}(\tau). \qquad (6)
$$

The probabilities in (4) and (5) can now be rewritten as $\mathbf{P}[\sum_{\boldsymbol{l}\in\mathcal{L}} x_{\boldsymbol{l}}^{B}(+\infty) \geq 1]$. Denote by $\boldsymbol{a}^{B}(t)$ the vector $[a_{\boldsymbol{l}}^{B}(t)]_{\boldsymbol{l}\in\mathcal{L}}$. The vectors $\boldsymbol{x}^{B}(t)$ and $\boldsymbol{e}^{B}(t)$ are defined similarly. Due to bounded arrival and departure rates, the quantities $(\boldsymbol{a}^{B}(t), \boldsymbol{x}^{B}(t), \boldsymbol{e}^{B}(t))$ are Lipschitz continuous. Hence, there exists a subsequence over which they converge uniformly over compact intervals (u.o.c.). Any such limit over the interval $[-T, 0]$ is called a fluid-sample-path (FSP). In other words, $(\boldsymbol{a}(t), \boldsymbol{x}(t), \boldsymbol{e}(t))$ is called a FSP if for some $T > 0$ there exists a sequence $(\boldsymbol{a}^{B}(t), \boldsymbol{x}^{B}(t), \boldsymbol{e}^{B}(t))$ that converges to it u.o.c. over $[-T, 0]$.

Note that FSPs are different from fluid limits [13]. Fluid limits are limiting processes to which $(\boldsymbol{a}^{B}(t), \boldsymbol{x}^{B}(t), \boldsymbol{e}^{B}(t))$ converges with probability 1. Hence, fluid limits capture the *mean* behavior of the system. In contrast, convergence to an FSP does not need to be with probability 1. Hence, an FSP is more general and captures large-deviations behavior that deviates from the mean.

B. Large Deviations Principle

Since the arrival process is *i.i.d.* in time, the sequence of scaled processes $\boldsymbol{a}^{B}(t)$ satisfies a sample path large deviations principle with some rate function $I_{a}^{T}(\cdot)$. What this means is the following. Let $\Phi_{a}[-T, 0]$ be the space of component-wise non-decreasing functions $\boldsymbol{a}(t)$ on $[-T, 0]$ with $\boldsymbol{a}(-T) = 0$. Let this space be equipped with the essential supremum norm [24, p176]. For any set Γ of trajectories in $\Phi_{a}[-T, 0]$, the probability that the sequence of scaled arrival processes $\boldsymbol{a}^{B}(t)$ fall into the set Γ satisfies: $\lim_{B\to\infty} \frac{1}{B} \log \mathbf{P}[\boldsymbol{a}^{B}(t) \in \Gamma] = -\inf_{\boldsymbol{a}\in\Gamma} I_{a}^{T}(\boldsymbol{a})$. That is, the decay-rate of the probability $\mathbf{P}[\boldsymbol{a}^{B}(t) \in \Gamma]$ is determined by the trajectory $\boldsymbol{a}$ in Γ with the least cost $I_{a}^{T}(\boldsymbol{a})$.

Under a given scheduling algorithm, if the mapping from the arrival process $\boldsymbol{a}^{B}(t)$ to the queue process $\boldsymbol{x}^{B}(t)$ is continuous, then one can apply the contraction principle [24] and conclude that the sequence of scaled queue processes $\boldsymbol{x}^{B}(t)$ satisfies a large deviations principle as well. That is, the rate of decay of the probability of overflow $\mathbf{P}[\sum_{\boldsymbol{l}\in\mathcal{L}} \boldsymbol{x}_{\boldsymbol{l}}^{B}(0) \geq 1]$ is determined by the minimum cost $I_{a}^{T}(\boldsymbol{a})$ among all trajectories $\boldsymbol{a}(t)$ that causes the queue to grow from $\boldsymbol{a}(-T) = 0$ at $t = -T$ to overflow at $t = 0$. The trajectory that attains this minimum-cost-to-overflow is called the "most likely path to overflow." Further, as $T \to +\infty$, one may show that this minimum-cost-to-overflow approaches the decay rate of the steady-state probability of overflow $\mathbf{P}[\sum_{\boldsymbol{l}\in\mathcal{L}} \boldsymbol{x}_{\boldsymbol{l}}^{B}(+\infty) \geq 1]$.

However, for the system that we are interested in, this approach encounters several difficulties [13]. First, for many scheduling algorithms it is very difficult to verify the continuity of the mapping from $\boldsymbol{a}^{B}(t)$ to $\boldsymbol{x}^{B}(t)$ and hence the contraction principle may not hold. Second, even if the contraction principle can be applied, it is difficult to find the minimum-cost-to-overflow because we have to solve a multi-dimensional calculus-of-variations problem. Third, even if we can compute the minimum-cost-to-overflow for a given algorithm, it is unclear how to optimize *across algorithms* to find the optimal algorithm.

In our earlier work [13], we establish a new result (Theorem 8 in [13]) that circumvents these difficulties. This result is re-stated in this paper as Theorem 3. Roughly speaking, this Theorem states that under certain assumptions on a Lyapunov function $V(\boldsymbol{x})$, if the scheduling algorithm π_0 minimizes the drift of the Lyapunov function $V(\boldsymbol{x}(t))$ at each point in time in every fluid sample path, then the algorithm π_0 must be large deviations decay-rate optimal for $\mathbf{P}[V(\boldsymbol{x}^B(+\infty)) \geq 1]$. This result has the following intuitive explanation: For any given FSP under algorithm π_0, since the algorithm π_0 minimizes the drift of the Lyapunov function at every time, it is plausible that the algorithm π_0 will also minimize the value of the Lyapunov function at the end of the FSP. We note however that this statement is not trivial to hold: minimizing the drift at each point in time is a myopic property, which may not always lead to globally optimal behavior! In our prior work [13], we have rigorously quantified the conditions under which the above statement holds (see also Theorem 3 in Section V). Once these conditions are satisfied, we can see that for any FSP that leads to overflow under algorithm π_0, the corresponding FSP (with the same arrival process $\boldsymbol{a}(t)$ and thus the same cost $I_a^T(\boldsymbol{a})$) under any other algorithm must also overflow. Hence, the minimum-cost-to-overflow (and thus the decay rate of the overflow probability) under algorithm π_0 must be no smaller than that under any other algorithms.

Hence, our problem becomes that of finding an algorithm (i.e., the P-TREE algorithm in Section IV) and verifying that it is drift-minimizing for the Lyapunov function $V(\boldsymbol{x}(t)) = \sum_{\boldsymbol{l}\in\mathcal{L}} \boldsymbol{x}_{\boldsymbol{l}}(t)$ at each time in every fluid sample path. This, however, is not a trivial task. Although it is not difficult to identify the minimum drift at a given point in an FSP, it is often much more difficult to find an algorithm *in the original discrete-time system* that attains the minimum drift. This is because drift minimization in FSP, even over an infinitesimally small interval δ, corresponds to the cumulative effect over time interval $B\delta$ in the original discrete-time system. However, in the original discrete-time system, an algorithm cannot know the "future" in the interval $B\delta$ before hand. As a result, it is not always easy to design a discrete-time algorithm that minimizes the drift in each time in every FSP. This discrepancy between discrete-time and fluid-scaled continuous-time was discussed in [25] for fluid limits, where the authors establish conditions under which minimizing the drift in discrete-time is sufficient for minimizing the drift in fluid limits. However, our Lyapunov function $\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(t)$ does not satisfy the conditions in [25], and hence the techniques there do not apply. Further, as readers will see, the P-TREE algorithm that we will propose in Section IV does not minimize the drift of the Lyapunov function $\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(t)$ in discrete-time either. For instance, if a link has insufficient data, i.e., $X_{\boldsymbol{l}}(t) < F_{\boldsymbol{l}}$, then that link is not considered for activation in that time-slot even though serving that link might drain more packets from the system. Nonetheless, we will develop new techniques that confirm that the P-TREE algorithm indeed minimizes the drift of the Lyapunov function at each time in every FSP. These techniques reveal non-trivial insights on the dynamics of the system under the P-TREE algorithm.

Finally, we emphasize that our task of proving the drift minimizing property for *fluid sample paths* is distinct from the more common proofs in the literature for proving that certain algorithms are drift minimizing for the *fluid limit*. As stated previously, fluid limits only capture the *mean* behavior where-as FSPs capture behaviors that deviate from the mean as well. Proving an algorithm to be drift minimizing for FSP is inherently more difficult since drift minimization must be shown for any conceivable system behavior.

IV. P-TREE SCHEDULER

We next describe P-TREE, a simple priority based scheduling algorithm tailored for the tree network. The algorithm is based on two guiding principles. First, we give priority to links that are closer to the root (destination) node and secondly, we give priority to links that carry more data per timeslot. The intuition is that by following the two principles, we hope to move data out of the network as fast as possible and hence reduce the total buffer occupancy in the network.

Only links that have enough data to fully utilize the capacity, i.e., $X_{\boldsymbol{l}}(t) \geq F_{\boldsymbol{l}}$, are considered for activation in time-slot t[†]. Let the set of such links be denoted by $\mathcal{A}(t)$. To choose the link for activation, the algorithm first considers links $\boldsymbol{l}$ of dimension 1 (i.e., those links directly connected to the root). It chooses to activate link $l^* = \min\{l | 1 \leq l \leq C, l \in \mathcal{A}(t)\}$. Recall that the links of dimension 1 are numbered in decreasing order of link capacity. Hence, l^* is the link with the largest capacity among links of dimension 1 that are considered for activation. Then the algorithm considers all interior nodes at depth 1, then all interior nodes at depth 2, and so on, till it has considered all interior nodes. Each time the algorithm considers an interior node $\boldsymbol{l}$, it performs the following:

If link $\boldsymbol{l}$ is activated, then none of the links$< \boldsymbol{l}, l >$ $(l = 1, \ldots, C(\boldsymbol{l}))$ will be activated (due to interference). Otherwise link $< \boldsymbol{l}, l^* >$ is activated where $l^* = \min\{l | 1 \leq l \leq C(\boldsymbol{l}), < \boldsymbol{l}, l > \in \mathcal{A}(t)\}$. Again, if we recall the structure used to label links, we can see that the above optimization problem is choosing the link with the largest capacity among all contending links.

This algorithm can be considered as a generalization of the algorithm π_0 specified in [18] where the authors establish that for a tandem topology (i.e., a tree with no branching) the algorithm is sample path optimal in terms of the sum queue. In comparison, in this work we will show that the P-TREE algorithm is large deviations decay rate optimal in terms of the sum queue for the more general tree topology.

V. ANALYSIS

Due to space constraints, we omit most of the proofs in the following discussions. We request interested readers to refer to our technical report [26] for detailed proofs.

Any FSP $(\boldsymbol{a}(t), \boldsymbol{x}(t), \boldsymbol{e}(t))$ is differentiable almost everywhere in the interval $[-T, 0]$ [13]. Denote the set of time

[†]Due to this restriction, the P-TREE algorithm does not always minimize the drift of $\sum_{\boldsymbol{l}\in\mathcal{L}} X_{\boldsymbol{l}}(t)$ in the discrete time.

instances where the FSP is not differentiable as $\mathcal{T}$. Then $\mathcal{T}$ is of measure 0. In the rest of this paper, we will restrict our discussion to time $t \notin \mathcal{T}$, and we will call such time a regular time. Define the following related quantities $f_{\boldsymbol{l}}(t) = \frac{1}{F_{\boldsymbol{l}}} \frac{d}{dt} a_{\boldsymbol{l}}(t)$ and $\mu_{\boldsymbol{l}}(t) = \frac{1}{F_{\boldsymbol{l}}} \frac{d}{dt} e_{\boldsymbol{l}}(t)$.

We remind the readers that $F_{\boldsymbol{l}} f_{\boldsymbol{l}}(t)$ is different from the mean arrival rate $\hat{\lambda}_{\boldsymbol{l}}(t)$ since we are considering an FSP. From the queue evolution equations (1) and (2), we can derive the following for the FSP:

$$\begin{aligned} \frac{d}{dt} x_{\boldsymbol{l}}(t) &= F_{\boldsymbol{l}} f_{\boldsymbol{l}}(t) + \sum_{l=1}^{C(\boldsymbol{l})} F_{<\boldsymbol{l},l>} \mu_{<\boldsymbol{l},l>}(t) - F_{\boldsymbol{l}} \mu_{\boldsymbol{l}}(t), \\ &\quad \text{if } \boldsymbol{l} \text{ is not a leaf} \\ \frac{d}{dt} x_{\boldsymbol{l}}(t) &= F_{\boldsymbol{l}} f_{\boldsymbol{l}}(t) - F_{\boldsymbol{l}} \mu_{\boldsymbol{l}}(t), \text{ if } \boldsymbol{l} \text{ is a leaf} \qquad (7) \\ \sum_{\boldsymbol{l} \in \mathcal{L}} \frac{d}{dt} x_{\boldsymbol{l}}(t) &= \sum_{\boldsymbol{l} \in \mathcal{L}} F_{\boldsymbol{l}} f_{\boldsymbol{l}}(t) - \sum_{l=1}^{C} F_l \mu_l(t). \qquad (8) \end{aligned}$$

For example, below we briefly illustrate how we can derive (7) for the case when $\boldsymbol{l}$ is a leaf. From (1), we have

$$\frac{1}{B} X_{\boldsymbol{l}}(B(T+t)) = \frac{1}{B} \sum_{\tau=0}^{B(T+t)} (A_{\boldsymbol{l}}(\tau) - E_{\boldsymbol{l}}(\tau)) + O(\frac{1}{B}).$$

The term $O(\frac{1}{B})$ accounts for the terms $\frac{1}{B}(X_{\boldsymbol{l}}(0) - A_{\boldsymbol{l}}(B(T+t)) + E_{\boldsymbol{l}}(B(T+t)))$. Taking the limit as $B \to \infty$ along the subsequence that gives us the FSP, we have $x_{\boldsymbol{l}}(t) = a_{\boldsymbol{l}}(t) - e_{\boldsymbol{l}}(t)$. Differentiating, we obtain (7).

The following proposition captures fundamental constraints in a converge-cast for the FSPs under any algorithm.

Proposition 1: For any scheduling algorithm, any FSP $(\boldsymbol{a}(t), \boldsymbol{x}(t), \boldsymbol{e}(t))$ must satisfy the following constraints for all regular time t.

Interference constraint equations:

$$\begin{aligned} \sum_{l=1}^{C} \mu_l(t) &\leq 1 \qquad (9) \\ \sum_{l=1}^{C(\boldsymbol{l})} \mu_{<\boldsymbol{l},l>}(t) + \mu_{\boldsymbol{l}}(t) &\leq 1 \text{ for all interior nodes } \boldsymbol{l} \qquad (10) \\ \mu_{\boldsymbol{l}}(t) &\in [0,1] \text{ for all nodes } \boldsymbol{l}. \qquad (11) \end{aligned}$$

Flow constraint equations:

$$\begin{aligned} \mu_{\boldsymbol{l}}(t) &\leq f_{\boldsymbol{l}}(t) + \sum_{l=1}^{C(\boldsymbol{l})} \frac{F_{<\boldsymbol{l},l>}}{F_{\boldsymbol{l}}} \mu_{<\boldsymbol{l},l>}(t) \qquad (12) \\ &\quad \text{if } x_{\boldsymbol{l}}(t) = 0 \text{ \& } \boldsymbol{l} \text{ is an interior node} \\ \mu_{\boldsymbol{l}}(t) &\leq f_{\boldsymbol{l}}(t) \text{ if } x_{\boldsymbol{l}}(t) = 0 \text{ \& } \boldsymbol{l} \text{ is a leaf}. \qquad (13) \end{aligned}$$

Remark: If we think of $\mu_{\boldsymbol{l}}(t)$ as the fraction of time that link $\boldsymbol{l}$ is activated, Equations (9)-(11) state that for any set of interfering links, the sum of the fractions of time that each link in the set is activated must be no greater than 1. Equations (12) and (13) state that when the queue backlog $x_{\boldsymbol{l}}(t)$ at node $\boldsymbol{l}$ is 0, the net flow of data into the node must be no smaller than the flow out of the node.

Define the Lyapunov function $V(\boldsymbol{x}(t)) \triangleq \sum_{\boldsymbol{l} \in \mathcal{L}} x_{\boldsymbol{l}}(t)$. We will use the results of [13] to prove that the P-TREE algorithm is optimal in terms of maximizing the large deviations decay rate. Specifically, define I_π and J_π (correspondingly, I_{p-t} and J_{p-t}) to be the quantities (4) and (5) when the system is operating under scheduling algorithm π (correspondingly, under P-TREE). We will show that $J_{p-t} \geq I_\pi$ for all algorithms π. Hence, the fastest rate of decay of $\mathbf{P}[\sum_{\boldsymbol{l} \in \mathcal{L}} X_{\boldsymbol{l}}(+\infty) \geq B]$ is that obtained under the P-TREE algorithm. Recall for future reference that $\mathbf{P}[\sum_{\boldsymbol{l} \in \mathcal{L}} X_{\boldsymbol{l}}(+\infty) \geq B] = \mathbf{P}[V(\boldsymbol{x}^B(+\infty)) \geq 1]$. We state the main result of the paper formally as follows.

Proposition 2: The P-TREE algorithm attains the optimal decay rate, i.e., for any scheduling algorithm π, we have

$$\begin{aligned} &\limsup_{B \to \infty} \frac{1}{B} \log \left(\mathbf{P}^{p-t} \left[\sum_{l \in \mathcal{L}} X_l(+\infty) \geq B \right] \right) \\ &\qquad \leq \liminf_{B \to \infty} \frac{1}{B} \log \left(\mathbf{P}^{\pi} \left[\sum_{l \in \mathcal{L}} X_l(+\infty) \geq B \right] \right), \end{aligned}$$

i.e., $J_{p-t} \geq I_\pi$.

To prove Proposition 2, we use the result of Theorem 8 from [13] which we restate here for reference.

Theorem 3: Let π_0 be a scheduling policy that satisfies Assumptions 1, 2, 3, 4, 5 and 6 of [13]. Let π be any scheduling policy, then $\limsup_{B \to \infty} \frac{1}{B} \log(\mathbf{P}^{\pi_0}[V(\boldsymbol{x}^B(+\infty)) \geq 1]) \leq \liminf_{B \to \infty} \frac{1}{B} \log(\mathbf{P}^{\pi}[V(\boldsymbol{x}^B(+\infty)) \geq 1])$.

Assumptions 1, 2, 3, 5 and 6 are restated in our technical report [26], along with the proof of Lemma 4 (stated below), which verifies that the P-TREE algorithm in fact satisfies the stated assumptions. The details are omitted here due to space constraints. Compared to Assumption 4 (stated below), these other assumptions are simpler to verify and the techniques are also similar to those in our prior work [6], [27].

Lemma 4: The P-TREE algorithm and Lyapunov function $V(\cdot)$ satisfy Assumptions 1, 2, 3, 5 and 6 mentioned in [13].

Assumption 4 is stated below.

Assumption 4: For any FSP $(\boldsymbol{a}(t), \boldsymbol{x}(t), \boldsymbol{e}(t))$, the algorithm π_0 satisfies the following for all regular time t:

$$\begin{aligned} \frac{d}{dt} V(\boldsymbol{x}(t)) = \min_{\hat{\boldsymbol{\mu}} = [\hat{\mu}_l]} \quad & \sum_{\boldsymbol{l} \in \mathcal{L}} F_{\boldsymbol{l}} f_{\boldsymbol{l}}(t) - \sum_{l=1}^{C} F_l \hat{\mu}_l \\ \text{subject to} \quad & \boldsymbol{x}(t), \boldsymbol{f}(t), \hat{\boldsymbol{\mu}} \text{ satisfy FSP constraints} \\ & \text{in Proposition 1.} \qquad (14) \end{aligned}$$

Note that the drift of the Lyapunov function $V(\boldsymbol{x}(t))$ is given by $\frac{d}{dt} V(\boldsymbol{x}(t)) = \sum_{\boldsymbol{l} \in \mathcal{L}} \frac{d}{dt} x_{\boldsymbol{l}}(t)$, where $\sum_{\boldsymbol{l} \in \mathcal{L}} \frac{d}{dt} x_{\boldsymbol{l}}(t)$ is given in (8). Thus, the objective function in the optimization problem on the right-hand-side of (14) is the drift $\frac{d}{dt} V(\boldsymbol{x}(t))$ at time t when the service vector is $\hat{\boldsymbol{\mu}}$, conditioned on the given values of $\boldsymbol{x}(t)$ and $\boldsymbol{f}(t)$. Hence, Assumption 4 states that along every FSP, the scheduling algorithm π_0 minimizes the drift $\frac{d}{dt} V(\boldsymbol{x}(t))$ at each point in time over all possible scheduling algorithms. This assumption is more difficult to verify and is the key assumption that the P-TREE algorithm needs to satisfy for the result Proposition 2 to hold. The rest of this section will be dedicated to verifying that the P-TREE algorithm in fact satisfies this assumption.

First, the optimization problem on the right-hand-side of (14) can be expanded into the following optimization problem, which can be interpreted as a lower bound on the drift $\frac{d}{dt}V(\boldsymbol{x}(t))$ of any FSP at a regular time t under any scheduling policy, conditioned on the given values of $\boldsymbol{x}(t)$ and $\boldsymbol{f}(t)$ at time t.

$$\text{optA}(\boldsymbol{f}(t),\boldsymbol{x}(t)) = \qquad (15)$$
$$\begin{array}{ll}\min\limits_{\boldsymbol{\mu}(t)} & \sum\limits_{\boldsymbol{l}\in\mathcal{L}} F_{\boldsymbol{l}}f_{\boldsymbol{l}}(t) - \sum\limits_{l=1}^{C} F_l\mu_l(t) \\ \text{sub to} & (9)-(13).\end{array}$$

To see that (15) indeed provides a lower bound for the drift, note that the objective function of the optimization problem is $\frac{d}{dt}V(\boldsymbol{x}(t))$ and the constraints are the set of inequalities satisfied by any FSP as stated in Proposition 1. By minimizing over all possible values of $\boldsymbol{\mu}(t)$, we obtain a lower bound on the drift under any algorithm, conditioned on $\boldsymbol{x}(t)$ and $\boldsymbol{f}(t)$ at time t.

The following Lemma formally states that the P-TREE algorithm satisfies Assumption 4.

Lemma 5: For any FSP $(\boldsymbol{a}(t),\boldsymbol{x}(t),\boldsymbol{e}(t))$ of the P-TREE algorithm, the drift is given by $\frac{d}{dt}V(\boldsymbol{x}(t)) = \text{optA}(\boldsymbol{f}(t),\boldsymbol{x}(t))$.

Before we present the proofs, we briefly illustrate where lies the difficulty in verifying that the P-TREE algorithm satisfies Assumption 4. Note that the drift of the Lyapunov function $\sum_{\boldsymbol{l}\in\mathcal{L}} x_{\boldsymbol{l}}$ is equal to $\sum_{\boldsymbol{l}\in\mathcal{L}} F_{\boldsymbol{l}}f_{\boldsymbol{l}} - F_{\boldsymbol{l}}\mu_{\boldsymbol{l}}$. Suppose that link 1 is at depth 1, and it has the largest rate among all links at depth 1. At a given point t in an FSP, suppose that $x_1(t) > 0$, which means that in the original discrete time system the backlog of link 1 is very large. Then it is easy to see that P-TREE minimizes the drift because it will always activate link 1, and hence $\mu_1(t) = 1$. What is more complicated, however, is when $x_1(t) = 0$. In this case, the backlog of link 1 in the original discrete-time system is close to (but not always equal to) zero. Hence, under the P-TREE algorithm, link l will be served for some fraction of time. The exact fraction will depend on the services at its children links that feed packets into link 1. The situation will become even more complicated when these children links $\boldsymbol{l}$ in turn have $x_{\boldsymbol{l}}(t) = 0$ in the FSP. Hence, in order to prove that P-TREE algorithm minimizes the drift at time t in FSP, we must carefully account for all possible combinations of the values $x_{\boldsymbol{l}}(t)$ (being zero or strictly positive), which makes the analysis quite complicated. However, we will develop an important result (Proposition 7) that reveals an interesting structure of the dynamics of the P-TREE algorithm, which successfully addresses the above difficulty.

The rest of the section is dedicated to proving Lemma 5 and is divided into two subsections. In Subsection V-A, we derive properties of the FSP under the P-TREE algorithm. Then, in Subsection V-B, we show that the drift under the P-TREE algorithm achieves the value of $\text{optA}(\boldsymbol{f}(t),\boldsymbol{x}(t))$. Since $\text{optA}(\boldsymbol{f}(t),\boldsymbol{x}(t))$ is a lower bound on the drift of any scheduling algorithm, this implies that the P-TREE algorithm satisfies Assumption 4 and that Lemma 5 holds.

A. Properties of FSPs under P-TREE Algorithm

The following lemma proves that whenever the backlog for a link is positive or if the backlog is zero but is growing at a positive rate, then under the P-TREE algorithm this link must receive all remaining service possible after service has been assigned to the higher priority links.

Lemma 6: Any FSP $(\boldsymbol{a}(t),\boldsymbol{x}(t),\boldsymbol{e}(t))$ under the P-TREE algorithm satisfies the following for regular time t:
For $l = 1,\ldots,C$, if $x_l(t) > 0$ or if $x_l(t) = 0$ and $\frac{d}{dt}x_l(t) > 0$, then $\mu_l(t) = 1 - \sum_{i=1}^{l-1}\mu_i(t)$.

For any interior node $\boldsymbol{l}$ and $l = 1,\ldots,C(\boldsymbol{l})$, if $x_{<\boldsymbol{l},l>}(t) > 0$ or if $x_{<\boldsymbol{l},l>}(t) = 0$ and $\frac{d}{dt}x_{<\boldsymbol{l},l>}(t) > 0$, then $\mu_{<\boldsymbol{l},l>}(t) = 1 - \mu_{\boldsymbol{l}}(t) - \sum_{i=1}^{l-1}\mu_{<\boldsymbol{l},i>}(t)$.

We can now prove the following proposition which determines the service received by a link.

Proposition 7: Any FSP $(\boldsymbol{a}(t),\boldsymbol{x}(t),\boldsymbol{e}(t))$ of P-TREE algorithm satisfies the following for all regular time t:
For $l = 1,\ldots,C$:
If $x_l(t) > 0$, then the following holds: $\mu_l(t) = 1 - \sum_{i=1}^{l-1}\mu_i(t)$.

If $x_l(t) = 0$, then the following holds

$$\mu_l(t) = \begin{cases} \min\Big(1 - \sum_{i=1}^{l-1}\mu_i(t)\ , \\ \qquad f_l(t) + \sum_{j=1}^{C(l)}\mu_{<l,j>}(t)\frac{F_{<l,j>}}{F_l}\Big) \\ \qquad\qquad \text{if } l \text{ is not a leaf.} \\ \min\Big(1 - \sum_{i=1}^{l-1}\mu_i(t), f_l(t)\Big) \\ \qquad\qquad \text{if } l \text{ is a leaf.}\end{cases}$$

For any interior node $\boldsymbol{l}$, $l = 1,\ldots,C(\boldsymbol{l})$:
If $x_{<\boldsymbol{l},l>}(t) > 0$, then the following holds: $\mu_{<\boldsymbol{l},l>}(t) = 1 - \mu_{\boldsymbol{l}}(t) - \sum_{i=1}^{l-1}\mu_{<\boldsymbol{l},j>}(t)$.

If $x_{<\boldsymbol{l},l>}(t) = 0$, then the following holds

$$\mu_{<\boldsymbol{l},l>}(t) = \begin{cases} \min\Big(1 - \mu_{\boldsymbol{l}}(t) - \sum_{i=1}^{l-1}\mu_{<\boldsymbol{l},j>}(t), \\ \qquad f_{<\boldsymbol{l},l>}(t) + \sum_{j=1}^{C(<\boldsymbol{l},l>)}\mu_{<\boldsymbol{l},l,j>}(t)\frac{F_{<\boldsymbol{l},l,j>}}{F_{<\boldsymbol{l},l>}})\Big) \\ \qquad\qquad \text{if } <\boldsymbol{l},l> \text{ is not a leaf.} \\ \min\Big(1 - \mu_{\boldsymbol{l}}(t) - \sum_{i=1}^{l-1}\mu_{<\boldsymbol{l},j>}(t), f_{<\boldsymbol{l},l>}(t)\Big) \\ \qquad\qquad \text{if } <\boldsymbol{l},l> \text{ is a leaf.}\end{cases}$$

Remark: The idea expressed by the proposition is the following. Consider link l connected to the root (the first set of equations). If $x_l(t) > 0$, then under any algorithm, the link l can at most be assigned all the remaining service after service has been assigned to higher priority links. Hence, we will have the inequality $\mu_l(t) \le 1 - \sum_{i=1}^{l-1}\mu_i(t)$ (see also Proposition 1). What Proposition 7 states is that under the P-TREE algorithm, we must have strict equality. Loosely speaking, this means that the P-TREE algorithm uses up all the service. On the other hand, if $x_l(t) = 0$, then the amount of service given to link l will be constrained by the additional requirement that the out-flow at a node can not exceed the in-flow into the node (see Proposition 1). For example, if link l is also a leaf node, then under any algorithm, the service given to link l will be determined by which of the two $1-\sum_{i=1}^{l-1}\mu_i(t)$ and $f_l(t)$ is smaller. Hence we will have the inequality $\mu_l(t) \le \min\left(1-\sum_{i=1}^{l-1}\mu_i(t), f_l(t)\right)$. Again, what Proposition 7 states is that the P-TREE algorithm attains strict equality. A similar intuition applies to other parts of the proposition.

B. Proof of Lemma 5

So far, we have only shown that the FSP of the P-TREE algorithm satisfies certain properties under different combinations of the value $x_{\boldsymbol{l}}(t)$ (being zero or strictly positive). To prove Prop. 2, we need to verify that P-TREE minimizes the drift at each time in every FSPs. More precisely, we need to prove that any FSP $(\boldsymbol{a}(t), \boldsymbol{x}(t), \boldsymbol{e}(t))$ of the P-TREE algorithm has drift equal to optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$, i.e., $\boldsymbol{\mu}(t)$ is an optimizer for optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$ at every t.

Our strategy is to prove by contradiction. Assume that $\boldsymbol{\mu}(t)$ does not optimize optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$. Then, there must exist a change in service $\boldsymbol{\delta}$ such that $\boldsymbol{\mu}(t)+\boldsymbol{\delta}$ provides a better value for the objective function of optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$ while at the same time satisfying the constraints of optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$. Note that the difference between the values of the objective function for $\boldsymbol{\mu}(t)$ and for $\boldsymbol{\mu}(t)+\boldsymbol{\delta}$ is equal to $\sum_{l=1}^{C} F_l\delta_l$. Hence, it suffices to show that no $\boldsymbol{\delta}$ can produce $\sum_{l=1}^{C} F_l\delta_l > 0$ while still satisfying the constraints of optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$. This is proved in Proposition 8.

Proposition 8: For all $\boldsymbol{\delta}$ such that $\boldsymbol{\mu}(t)+\boldsymbol{\delta}$ satisfies the constraint equations for optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$, we must have $\sum_{l=1}^{C} F_l\delta_l \le 0$.

Before we can prove Proposition 8, we will need to derive certain properties of $\boldsymbol{\delta}$ based on the assumption that $\boldsymbol{\mu}(t)+\boldsymbol{\delta}$ satisfies the constraints of optA.

Lemma 9: If $\boldsymbol{\mu}(t)+\boldsymbol{\delta}$ satisfies the constraints of optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$, then $\boldsymbol{\delta}$ satisfies the following:
For $l = 1, \ldots, C$:

$$\delta_l \le \begin{cases} \max\left(-\sum_{i=1}^{l-1}\delta_i, \sum_{j=1}^{C(l)}\delta_{<l,j>}\frac{F_{<l,j>}}{F_l}\right) \\ \qquad \text{if } l \text{ is not a leaf.} \\ \max\left(-\sum_{i=1}^{l-1}\delta_i, 0\right) \\ \qquad \text{if } l \text{ is a leaf.} \end{cases}$$

For any interior node $\boldsymbol{l}$ and $l = 1, \ldots, C(\boldsymbol{l})$

$$\delta_{<\boldsymbol{l},l>} \le \begin{cases} \max\left(-\delta_{\boldsymbol{l}} - \sum_{i=1}^{l-1}\delta_{<\boldsymbol{l},j>},\right. \\ \qquad \left.\sum_{j=1}^{C(<\boldsymbol{l},l>)}\delta_{<\boldsymbol{l},l,j>}\frac{F_{<\boldsymbol{l},l,j>}}{F_{<\boldsymbol{l},l>}})\right) \\ \qquad \text{if } <\boldsymbol{l},l> \text{ is not a leaf.} \\ \max\left(-\delta_{\boldsymbol{l}} - \sum_{i=1}^{l-1}\delta_{<\boldsymbol{l},j>}, 0\right) \\ \qquad \text{if } <\boldsymbol{l},l> \text{ is a leaf.} \end{cases}$$

Remark: Note that this is a critical property for the overall proof because it holds regardless of the value of $x_{\boldsymbol{l}}(t)$ (which, as we discussed before, has been the main source of complexity). This Lemma expresses the following intuition. Take a link l at depth 1 for example. Recall from Proposition 7 that the P-TREE algorithm uses up all the service available. In such a situation, the increase in service δ_l for any link l is constrained by two factors. We must either sacrifice service (i.e., reduce δ_j) at higher priority links $j = 1, \ldots, l-1$ or increase service to the children of l. Hence, the change in service δ_l can at most be $\max\left(-\sum_{i=1}^{l-1}\delta_i, \sum_{j=1}^{C(l)}\delta_{<l,j>}\frac{F_{<l,j>}}{F_l}\right)$. Note that if a link is a leaf, then it does not have children and hence the second factor does not appear.

The following lemma, which uses Lemma 9, is essential to prove Proposition 8.

Lemma 10: If $\boldsymbol{\mu}(t)+\boldsymbol{\delta}$ satisfies the constraints for optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$, then $\boldsymbol{\delta}$ satisfies the following:
(a) Consider any node $\boldsymbol{l}$ that is not a leaf. Let l be any child of $\boldsymbol{l}$. If $<\boldsymbol{l},l>$ is not a leaf and all its children ($i = 1, \ldots, C(<\boldsymbol{l},l>)$) satisfy $\delta_{<\boldsymbol{l},l,i>} \le \max\left(-\delta_{<\boldsymbol{l},l>} - \sum_{j=1}^{i-1}\delta_{<\boldsymbol{l},l,j>}, 0\right)$, then $\delta_{<\boldsymbol{l},l>} \le \max(-\delta_{\boldsymbol{l}} - \sum_{j=1}^{l-1}\delta_{<\boldsymbol{l},j>}, 0)$.
(b) Consider any node l that is the child of the root. If l is not a leaf and all its children ($i = 1, \ldots, C(l)$) satisfy $\delta_{<l,i>} \le \max\left(-\delta_l - \sum_{j=1}^{i-1}\delta_{<l,j>}, 0\right)$, then $\delta_l \le \max(-\sum_{j=1}^{l-1}\delta_j, 0)$.
Remark: Part (a) of the lemma states that if all children of $<\boldsymbol{l},l>$ (i.e., $<\boldsymbol{l},l,i>$) satisfy the property that, to increase service to $<\boldsymbol{l},l,i>$, we must reduce service from higher priority nodes $<\boldsymbol{l},l>, <\boldsymbol{l},l,1>, \ldots, <\boldsymbol{l},l,i-1>$, then link $<\boldsymbol{l},l>$ also satisfies this property, i.e., to increase service to link $<\boldsymbol{l},l>$, we must reduce service to its higher priority links $\boldsymbol{l}, <\boldsymbol{l},1>, \ldots, <\boldsymbol{l},l-1>$. Part (b) is a special case for when the link is directly connected to the root node.

The significance of Lemma 10 is as follows. As we can see from Lemma 9, the leaf nodes satisfy the above-mentioned property that an increase in service to the link must come at a reduction in service to higher priority links. Lemma 10 states that if a link's children satisfy this property, then the link itself must also satisfy this property. Clearly, this idea leads to the propagation of this property up the tree from the leaf nodes and hence we expect that all links in the tree must satisfy this property. This is explicitly stated in Lemma 11.

Lemma 11: If $\boldsymbol{\mu}(t)+\boldsymbol{\delta}$ satisfies the constraint equations for optA$(\boldsymbol{f}(t), \boldsymbol{x}(t))$, then $\boldsymbol{\delta}$ satisfies the following:
(a) Consider any node $\boldsymbol{l}$ that is not a leaf. Let l be any child of $\boldsymbol{l}$. Then, $\delta_{<\boldsymbol{l},l>} \le \max(-\delta_{\boldsymbol{l}} - \sum_{j=1}^{l-1}\delta_{<\boldsymbol{l},j>}, 0)$.
(b) Consider any node l that is the child of the root. Then, $\delta_l \le \max(-\sum_{j=1}^{l-1}\delta_j, 0)$.

We are now ready to prove Proposition 8.

Proof: [of Proposition 8] By Lemma 11, we know that $\delta_l \le \max(-\sum_{j=1}^{l-1}\delta_j, 0)$ for $l = 1, \ldots, C$.

To prove the proposition, it is sufficient to show that for any mathematical quantities δ_l that satisfy the inequalities

$$\delta_l \le \max(-\sum_{j=1}^{l-1}\delta_j, 0), \tag{16}$$

for $l = 1, \ldots, C$, and any non-increasing, non-negative sequence $\{F_l\}$, the following is true: $\sum_{l=1}^{C} \delta_l F_l \leq 0$.

We emphasize that this is a purely mathematical result and that allowing $\{F_l\}$ to represent various sequences is simply a trick to shorten the proof. It does not mean that we consider various systems with different values for the link capacities.

We will prove this by induction. By (16), we know that $\delta_1 \leq 0$. Hence, $\sum_{l=1}^{1} \delta_l F_l \leq 0$ for any non-increasing, non-negative sequence of numbers $\{F_l\}$. Now, assume $\sum_{l=1}^{2} \delta_l F_l \leq 0, \ldots, \sum_{l=1}^{k-1} \delta_l F_l \leq 0, \sum_{l=1}^{k} \delta_l F_l \leq 0$ for any non-increasing, non-negative sequence of numbers $\{F_l\}$. We will show that this implies $\sum_{l=1}^{k+1} \delta_l F_l \leq 0$. There are two cases to consider. If $\delta_{k+1} \leq 0$, then the result immediately follows. On the other hand, if $\delta_{k+1} > 0$, by (16) we have $0 < \delta_{k+1} \leq -\sum_{l=1}^{k} \delta_l$. Hence, $\sum_{l=1}^{k+1} \delta_l F_l \leq \sum_{l=1}^{k} \delta_l F_l - \sum_{l=1}^{k} \delta_l F_{k+1} = \sum_{l=1}^{k} \delta_l \left(F_l - F_{k+1}\right)$. Since $\{F_l\}$ is a non-increasing sequence, the sequence $F_1 - F_{k+1}, \ldots, F_k - F_{k+1}$ is also a non-increasing non-negative sequence of length k. Hence, by the induction hypothesis, $\sum_{l=1}^{k} \delta_l \left(F_l - F_{k+1}\right) \leq 0$. This implies that $\sum_{l=1}^{k+1} \delta_l F_l \leq 0$ for any non-increasing, non-negative sequence $\{F_l\}$. ■

VI. Simulation

In this section, we present MATLAB simulation results for the topology shown in Figure 1. Note that the nodes/links are labelled according to the scheme in Section II. This topology consists of 12 nodes with two nodes at depth 1, 5 nodes at depth 2 and 4 nodes at depth 3. Six of the nodes are leaf nodes. There are 8 flows in the network, each with the root as the destination. In each time slot, one packet arrives at (or is generated by) each source node with a certain fixed probability, independent of other flows and other time slots. The average arrival rate for the flow originating at a node is labelled on the node. For example, the average arrival rate for the flow originating at node $(1,1,1)$ is 0.25. The numbers near the links denote the capacity of the link. For example, link $(1,1)$ has capacity 3 and link $(2,1)$ has capacity 2 (recall that, according to our convention, links $(1,1)$ and $(2,1)$ denote the links connecting node $(1,1)$ and node $(2,1)$, respectively, to their parents). We define $S(\boldsymbol{X}(t)) = X_1(t) + X_2(t) + X_{(1,1)}(t) + X_{(1,2)}(t) + X_{(1,3)}(t) + X_{(2,1)}(t) + X_{(2,2)}(t) + X_{(1,1,1)}(t) + X_{(1,1,2)}(t) + X_{(2,1,1)}(t) + X_{(2,2,1)}(t)$.

Our metric of interest is the overflow probability $\mathbf{P}[S(\boldsymbol{X}(t)) > B]$. We simulate the system under different scheduling policies: P-TREE scheduler, back-pressure & back-pressure-α schedulers and the multi-hop version of greedy maximal matching (GMM).

Let us briefly review the back-pressure [1] and greedy maximal matching [19] policies. Both policies have the following common features. The differential backlog across a link is the difference of the backlog at the source node of the link and that at the destination node of the link. For example, the differential backlog of the link $(1,1)$ is $X_{(1,1)} - X_1$. Each link is assigned a weight $W_{\boldsymbol{l}}$ that is the product of the differential backlog and the link capacity. For example, $W_{(1,1)} = (X_{(1,1)} - X_1)3$. The back-pressure scheduler will activate links (subject to interference constraints) in such a fashion as to maximize the

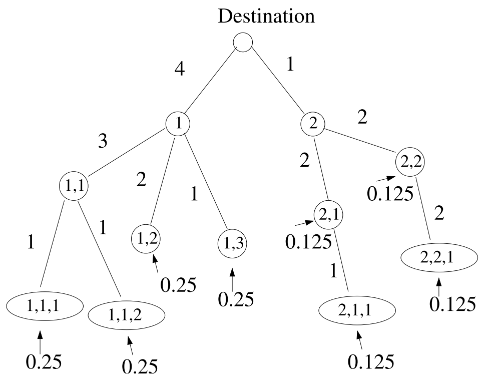

Fig. 1. System topology for simulation

sum of the weights of the activated links. The greedy maximal matching will instead do the following. It will first activate the link with the largest weight. Then, it will remove from consideration all links that interfere with this activated link. From the remaining links, it will activate the link with the largest weight and remove from consideration the links that interfere with this link. This procedure is repeated till there are no more links available.

The back-pressure-α algorithm is similar to the back-pressure algorithm except that, instead of taking the difference of the backlogs, the algorithm takes the difference of the backlogs raised to a power α. That is, the weight of link $(1,1)$ will be $W_{(1,1)} = (X_{(1,1)}^{\alpha} - X_1^{\alpha})3$. It can be shown that this algorithm minimizes the drift of the Lyapunov function $(\sum_{\boldsymbol{l} \in \mathcal{L}} X_{\boldsymbol{l}}^{\alpha+1})^{1/(\alpha+1)}$ and hence it is large deviations decay-rate optimal for the probability of overflow $\mathbf{P}((\sum_{\boldsymbol{l} \in \mathcal{L}} X_{\boldsymbol{l}}^{\alpha+1})^{1/(\alpha+1)} > B)$ [13]. As $\alpha \to 0$, we have $(\sum_{\boldsymbol{l} \in \mathcal{L}} X_{\boldsymbol{l}}^{\alpha+1})^{1/(\alpha+1)} \to \sum_{\boldsymbol{l} \in \mathcal{L}} X_{\boldsymbol{l}}$. Hence, as $\alpha \to 0$, one would expect this algorithm to have near-optimal performance in terms of the decay-rate for $\mathbf{P}(\sum_{\boldsymbol{l} \in \mathcal{L}} X_{\boldsymbol{l}} > B)$.

Back-pressure and back-pressure-α schedulers entail a high computational complexity due to the fact that the algorithms search for the best way to activate links in order to maximize the total weight. The greedy maximal matching algorithm overcomes this issue [19], [28]. For the node-exclusive interference model that we consider, the back-pressure and back-pressure-α schedulers reduce to a maximum-weighted matching problem that has complexity $O(|\mathcal{L}|^3)$ [21]. The greedy maximal matching algorithm has complexity $O(|\mathcal{L}| \log(|\mathcal{L}|))$ [19]. Our proposed P-TREE algorithm has an even lower complexity of $O(|\mathcal{L}|)$.

In Figure 2, we plot $\mathbf{P}[S(X) > B]$ vs B with the y-axis in log scale. We observe that the P-TREE scheduler has the best decay rate and indeed performs much better than the other schedulers. The back-pressure-α algorithm appears to perform very poorly as α is reduced. This is because of the large-deviations decay-rate kicking in at higher and higher values of the threshold B. This effect has been documented in detail in our other works [6], [27]. In contrast, our P-TREE algorithm not only maximizes the decay rate but also performs very well

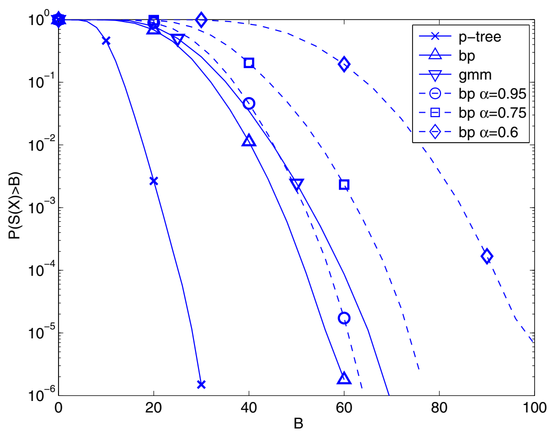

Fig. 2. The overflow probability of the sum-queue versus threshold B.

at small overflow thresholds.

VII. Conclusion

In this work, we consider the problem of scheduling links in a wireless multi-hop system performing convergecast. The goal of the scheduling algorithm is to minimize the sum-queue backlog over the network. We design a novel low complexity scheduling algorithm called P-TREE scheduler and prove that this scheduler maximizes the decay rate of the probability that the sum-queue exceeds a certain threshold. We use simulations to compare this algorithm with the well known back-pressure scheduler and the multi-hop version of greedy maximal matching scheduler. The P-TREE scheduler is seen to perform much better than these well known algorithms not only in terms of decay rate but also in terms of actual probabilities of overflow at small overflow thresholds.

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. 1936–1948, December 1992.
[2] M. J. Neely, E. Modiano, and C. E. Rohrs, “Dynamic Power Allocation and Routing for Time Varying Wireless Networks,” *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad-Hoc Networks*, vol. 23, no. 1, pp. 89–103, January 2005.
[3] S. Shakkottai, R. Srikant, and A. Stolyar, “Pathwise Optimality of the Exponential Scheduling Rule for Wireless Channels,” *Advances in Applied Probability*, pp. 1021–1045, December 2004.
[4] B. Sadiq, S. J. Baek, and G. de Veciana, “Delay-Optimal Opportunistic Scheduling and Approximations: The Log Rule.” in *Proceedings of IEEE INFOCOM*, April 2009.
[5] X. Lin and V. J. Venkataramanan, “On the Large-Deviations Optimality of Scheduling Policies Minimizing the Drift of a Lyapunov Function,” in *47th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, September 2009.
[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] L. Ying, S. Shakkottai, and A. Reddy, “On Combining Shortest-Path and Back-Pressure Routing Over Multihop Wireless Networks,” in *Proceedings of IEEE INFOCOM*, April 2009.
[8] 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.
[9] M. J. Neely, “Order Optimal Delay for Opportunistic Scheduling in Multi-User Wireless Uplinks and Downlinks,” *IEEE/ACM Transactions on Networking*, 2008.
[10] ——, “Delay Analysis for Maximal Scheduling in Wireless Networks with Bursty Traffic,” in *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008.
[11] A. L. Stolyar, “MaxWeight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic,” *Annals of Applied Probability*, vol. 14, no. 1, pp. 1–53, 2004.
[12] S. Shakkottai, “Effective Capacity and QoS for Wireless Scheduling,” *IEEE Transactions on Automatic Control*, vol. 53, no. 3, April 2008.
[13] V. J. Venkataramanan and X. Lin, “On the Queue-Overflow Probability of Wireless Systems: A New Approach Combining Large Deviations with Lyapunov Functions,” *submitted to IEEE Trans. on Information Theory*, 2009. [Online]. Available: http://min.ecn.purdue.edu/%7Elinx/publications.html
[14] A. L. Stolyar and K. Ramanan, “Largest Weighted Delay First Scheduling: Large Deviations and Optimality,” *Annals of Applied Probability*, vol. 11, no. 1, pp. 1–48, 2001.
[15] A. L. Stolyar, “Large Deviations of Queues Sharing a Randomly Time-varying Server,” *Queueing Systems*, vol. 59, pp. 1–35, 2008.
[16] G. R. Gupta and N. B. Shroff, “Delay Analysis for Multi-hop Wireless Networks,” in *Proceedings of IEEE INFOCOM*, April 2009.
[17] S. Jagabathula and D. Shah, “Optimal Delay Scheduling in Networks with Arbitrary Constraints,” in *ACM SIGMETRICS*, June 2008.
[18] 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, 1994.
[19] X. Lin and N. B. Shroff, “The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks,” *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 302–315, 2006.
[20] C. Joo, X. Lin, and N. B. Shroff, “Greedy Maximal Matching: Performance Limits for Arbitrary Network Graphs Under the Node-exclusive Interference Model,” *IEEE Trans. on Automatic Control*, vol. 54, no. 12, pp. 2734–2744, December 2009.
[21] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, New Jersey: Prentice-Hall, 1982.
[22] Y. Yi and S. Shakkottai, “Hop-by-hop Congestion Control over a Wireless Multi-hop Network,” in *Proceedings of IEEE INFOCOM*, Hong Kong, March 2004.
[23] S. Sarkar and L. Tassiulas, “End-to-end Bandwidth Guarantees Through Fair Local Spectrum Share in Wireless Ad-hoc Networks,” in *Proceedings of the IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003.
[24] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*, 2nd ed. New York: Springer-Verlag, 1998.
[25] S. P. Meyn, “Stability and Asymptotic Optimality of Generalized MaxWeight Policies,” *SIAM J. Control and Optimization*, vol. 47, no. 6, 2009.
[26] V. J. Venkataramanan and X. Lin, “Low-Complexity Scheduling Algorithm for Sum-Queue Minimization in Wireless Convergecast,” *Technical Report, Purdue University*, 2010. [Online]. Available: http://min.ecn.purdue.edu/%7Elinx/papers.html
[27] V. J. Venkataramanan, X. Lin, L. Ying, and S. Shakkottai, “On Scheduling for Minimizing End-to-End Buffer Usage over Multihop Wireless Networks,” in *Proceedings of IEEE INFOCOM*, March 2010.
[28] C. Joo, X. Lin, and N. B. Shroff, “Understanding the Capacity Region of the Greedy Maximal Scheduling Algorithm in Multi-hop Wireless Networks,” in *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008.