Delay Efficient Scheduling via Redundant Constraints in Multihop Networks

Longbo Huang, Michael J. Neely

***Abstract*—We consider the problem of delay-efficient scheduling in general multihop networks. While the class of max-weight type algorithms are known to be throughput optimal for this problem, they typically incur undesired delay performance. In this paper, we propose the *Delay-Efficient SCheduling algorithm* (DESC). DESC is built upon the idea of *accelerating queues* (AQ), which are virtual queues that quickly propagate the traffic arrival information along the routing paths. DESC is motivated by the use of redundant constraints to accelerate convergence in the classic optimization context. We show that DESC is throughput-optimal. The delay bound of DESC can be better than previous bounds of the max-weight type algorithms which did not use such traffic information. We also show that under DESC, the service rates allocated to the flows converge quickly to their target values and the average total "network service lag" is small. In particular, when there are $O(1)$ flows and the rate vector is of $\Theta(1)$ distance away from the boundary of the capacity region, the average total "service lag" only grows *linearly* in the network size.**

***Index Terms*—Queueing, Scheduling, Dynamic Power Allocation, Redundant Constraints, Lyapunov analysis**

I. Introduction

We consider the problem of routing a set of flows over a general multihop network, which operates in slotted time $t = 0, 1, 2, ...$ At every time slot, there are random packet arrivals entering the network. These packets eventually need to be routed to their destination. Therefore, at every time slot, the network operator needs to allocate power to the network links and transmit data over the links with the corresponding rates. Depending on the physical configuration of the network, there will be constraints on how the power and rates can be allocated. The goal of the operator is to find a joint power allocation, routing and scheduling policy that stabilizes the network and preferably, ensures a low delay of the data.

This is a classic routing problem that has been studied in previous work, e.g., [1], [2]. It is known that the class of max-weight type scheduling algorithms are able to stabilize the network whenever the arrival rate vector is within the network capacity region, hence are *throughput optimal*. For instance, the "Longest Connected Queue" (LCQ) algorithm [3] and the "Longest Connected Group" (LCG) algorithm [4] for downlink scheduling problems, and the "Dynamic Routing and Power Control" (DRPC) and Ehanced-DRPC (EDRPC) algorithms [1] [2] for general multihop networks are examples of the max-weight type algorithms. However, though being optimal in throughput performance, the max-weight type algorithms are not guaranteed to yield good delay performance. Indeed, the known delay-efficient results of the max-weight type algorithms are mainly for single-hop networks. In downlink systems with binary arrival and service variables, LCQ and LCG are shown to be *delay-order-optimal*, i.e., delay in the network does not grow with the number of users [4] [5]. In a single hop network with arrival rate in a reduced capacity region, the simpler maximal-weight scheduling algorithm is shown to be delay-order-optimal for both i.i.d. and bursty traffic [6]. For general multihop networks, a delay bound that is polynomial in the number of nodes is computed in [2] for the DRPC algorithm, although the bound is not order-optimal, and [7] derives a lower bound of the network delay.

In fact, it has been argued that the max-weight type algorithms perform poorly in delay for multihop networks mainly due to the following two disadvantages [8]: (1) They maintain a separate queue for each commodity flow at each node, and (2) They may route packets via a long route even if shorter routes are available. Many algorithms have thus been proposed trying to improve the delay performance of the max-weight type algorithms by alleviating the effect of these two disadvantages. [9] proposes a cluster-based max-weight algorithm to reduce the number of queues maintained at each node. [10] combines the shortest path routing with the max-weight scheduling algorithm. The scheme is able to reduce packet hop counts to the destination but can suffer from large queueing delay. [8] proposes using shadow queues to make scheduling decisions and using a single FIFO queue instead of one queue for each flow commodity. Simulation results show that the scheme is able to dramatically reduce delay. [11] combines max-weight and dynamic programming and proposes the scheme "Opportunistic Routing with Congestion Diversity" (ORCD) where packets are routed to the node with the "minimum-delay-to-go." In [12], the authors also prove that ORCD is throughput optimal. Outside the max-weight context, there has also been work designing and studying delay-efficient algorithms. [13] develops a delay-order-optimal scheduling algorithm for traffic in a reduced capacity region. [14] uses a dynamic programming approach to derive delay-efficient algorithms for the downlink scheduling problem. [15] and [16] instead use the large deviation approach to study and design algorithms that maximize the queue length decay exponent. However, though the proposed schemes are intuitively delay efficient, it is usually difficult to obtain explicit delay bounds for the non-max-weight type algorithms, and the analytical bounds for the max-weight type schemes are

Longbo Huang (web: http://www-scf.usc.edu/∼longbohu) and Michael J. Neely (web: http://www-rcf.usc.edu/∼mjneely) are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA.

This material is supported in part by one or more of the following: the DARPA IT-MANET program grant W911NF-07-0028, the NSF grant OCE 0520324, the NSF Career grant CCF-0747525.

usually not better than those of DRPC [2], which will serve as the benchmark algorithm for comparison in this paper.

In this paper, we focus on designing throughput-optimal delay-efficient algorithms using max-weight type approaches. However, we do not try to solve the two disadvantages mentioned above. We instead note that in all the above max-weight type algorithms, only the *actual* queue sizes are used to build the algorithms. We thus construct the *Delay-Efficient SCheduling algorithm* (DESC), a max-weight type algorithm, based on the actual network queues and a set of *virtual accelerating queues* (AQs), which quickly propagate the traffic arrival information down the routing path.

Our approach is motivated by the convergence accelerating techniques in the classic optimization context, where by adding redundant constraints to the original problem, one usually obtain a faster convergence of gradient-type algorithms [17]. As shown in [18], many of the max-weight type algorithms are closely related to solving a discrete version of some rate allocation optimization problem with a dual subgradient method. Thus one major role of the backlogs is to propagate the traffic information down the routing path. However, if only the actual queue sizes are used to do so, the queues have to build up to create "gradients" towards the destination, leading to large network delay. In fact, such undesired delay performance can happen even in the simple case of routing a single flow over a fixed route using a max-weight type algorithm [8]. DESC instead uses virtual queues to provide traffic information to intermediate nodes. Under DESC, the convergence of the allocated service rates to their target values is faster than that under the DRPC algorithm. The delay bound we obtain can also be potentially better than those of DRPC. Finally, we note that DESC can be combined with some of the above approaches to also deal with the two aforementioned disadvantages and to further reduce network delay. Interestingly, we note that the order-optimality proof of the max-weight algorithm for downlink systems in [5] can be viewed as considering all the redundant constraints in some associated optimization problem for the delay analysis.

This paper is organized as follows. In Section II we state our model. We then introduce the notion of an *accelerating queue* (AQ) and present the DESC algorithm in Section III. Section IV summarizes the performance results of DESC. The results are then proved in Section V. Section VI considers the case when there is delay in propagating the arrival information. Simulation results are in Section VII. For convenience, we summarize all the notations in this paper in Table I.

TABLE I
TABLE OF NOTATIONS

Notation	Definition
s_f, d_f	source and destination nodes of flow f
$A_f(t)$	number of arrivals for flow f at time t
P_f, K_f	the routing path of flow f and its length
$k^f(n)$	the order of node n in path P_f
$u^f(n)$, $l^f(n)$	the upstream and downstream nodes of n in P_f
$\boldsymbol{S}(t)$	the aggregate network channel state
$\boldsymbol{P}(t)$	the network power allocation vector
$\mu^f_{[m,n]}(t)$	the rate allocated to flow f over link $[m,n]$ at t
$Q^f_n(t)$	the queue size of flow f at node n at t
$H^f_n(t)$	the accelerating queue size of flow f at node n at t
$W^*_{[m,n]}(t)$	the weight of link $[m,n]$ at time t
$D^f_n(t)$	the messaging passing delay of $A_f(t)$ for node n at t
D	upper bound of $D^f_n(t)$ for all f, n, t
$\text{Lag}(t)$	the approximate network service lag

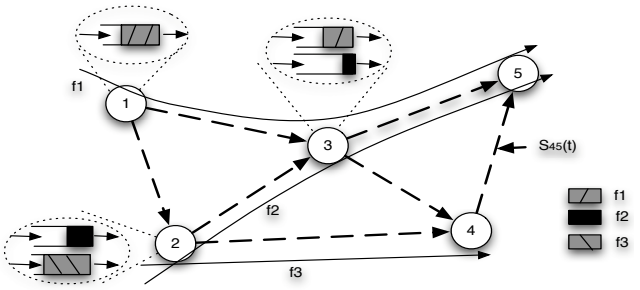

Fig. 1. Flows traversing a multihop network.

II. NETWORK MODEL

We consider a network operator that operates a general multihop network as shown in Fig. 1. The network is modeled as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \{1, 2, ..., N\}$ denotes the set of N nodes and $\mathcal{L}$ denotes the set of L directed links in the network. Here a link $[m, n]$ where $m, n \in \mathcal{N}$ is in $\mathcal{L}$ if there is a communication link from node m to node n. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, ...\}$. The goal of the operator is to support a set of flows going through the network and to achieve good delay performance.

A. The Flow and Routing Model

We assume there are a total of M flows going through the network. We denote the set of flows as $\mathcal{F} = \{1, 2, ..., M\}$. Each flow $f \in \mathcal{F}$ enters the network from its source node s_f and needs to be routed to its destination node d_f. Let $A_f(t)$ denote the number of packets of flow f arriving at its source node at time t. We assume $A_f(t)$ is i.i.d. every slot and let $\lambda_f = \mathbb{E}\{A_f(t)\}$. The operator can observe the value of $A_f(t)$ at every slot. However, the statistics of $A_f(t)$ may be unknown. We assume there exists some constant $A_{max} < \infty$ such that $A_f(t) \leq A_{max}$ for all f and t.

For each pair of nodes s, d, we define an acyclic path P between them to be a sequence of nodes $P = (n_1, n_2, ..., n_{K+1})$ such that $[n_i, n_{i+1}] \in \mathcal{L}$ for all $i = 1, ..., K$, $n_1 = s$, $n_{K+1} = d$ and $n_i \neq n_j$ if $i \neq j$. In a general multihop network, flows can usually be routed to their destination via multiple paths. In this paper, in order to highlight the scheduling component, we assume that every flow f is routed via a single fixed path P_f to its destination. The results in this paper can easily be extended to the case where every flow is routed over multiple acyclic fixed paths to their destinations. We define the *length* of the path $P_f = \{s_f = n_1, ..., n_{K_f+1} = d_f\}$ for flow f to be K_f. We use $k^f(n)$ to denote node n's order in the path P_f for all $n \in P_f$, e.g., $k^f(s_f) = 1$ and $k^f(d_f) = K_f + 1$. For any node n with $k^f(n) \geq 2$, we use $u^f(n)$ to denote its *upstream* node in the path P_f; for any node n with $k^f(n) \leq K_f$, we use $l^f(n)$ to denote its *downstream* node in the path P_f.

Note that this routing model is different from the multihop

network considered in [2], [11], where no route information is needed. However, this assumption is not very restrictive. In many cases, e.g., the internet, such route information can easily be obtained.

B. The Transmission Model

We assume the channel states of the links are potentially time varying. We use $\boldsymbol{S}(t) = (S_{mn}(t), m, n \in \mathcal{N})$, where $S_{mn}(t)$ is the channel condition between nodes m and n, to denote the aggregate channel state vector of the network. Note that $\boldsymbol{S}(t)$ contains information of channels between all pairs of nodes in the network, even pairs that do not have a communication link in between. This is to capture the fact that in some cases, though one node can not communicate with another node, their transmissions can still interfere with each other. We assume $\boldsymbol{S}(t)$ takes values in a finite state space $\boldsymbol{\mathcal{S}}$. For simplicity, we also assume that $\boldsymbol{S}(t)$ is i.i.d. over slots, but the components of $\boldsymbol{S}(t)$ are allowed to be correlated. The network operator can observe the aggregate channel state information $\boldsymbol{S}(t)$ at every time slot, but the distribution of $\boldsymbol{S}(t)$ is not necessarily known.

At every time t, after observing the channel state vector $\boldsymbol{S}(t)$, the network operator allocates power to each link for data transmission. It does so by choosing a power allocation vector $\boldsymbol{P}(t) = (P_{[m,n]}(t), [m,n] \in \mathcal{L})$, where $P_{[m,n]}(t)$ denotes the power allocated to link $[m,n]$. We assume that if $\boldsymbol{S}(t) = \boldsymbol{S} \in \boldsymbol{\mathcal{S}}$, then $\boldsymbol{P}(t)$ is chosen from a feasible power allocation set associated with $\boldsymbol{S}$, denoted by $\mathcal{P}^{(\boldsymbol{S})}$. We assume for any $\boldsymbol{S} \in \boldsymbol{\mathcal{S}}$, $\mathcal{P}^{(\boldsymbol{S})}$ is compact and time invariant. Given the channel state $\boldsymbol{S}(t)$ and the power vector $\boldsymbol{P}(t)$, the rate over link $[m,n]$ at time t is given by $\mu_{[m,n]}(t) = \Phi_{[m,n]}(\boldsymbol{S}(t), \boldsymbol{P}(t))$, for some general rate-power function $\Phi_{[m,n]}(\cdot,\cdot)$. Now let $\mu^f_{[m,n]}(t)$ be the rate allocated to flow f over link $[m,n]$ at time t, chosen subject to the following constraint: $\sum_f \mu^f_{[m,n]}(t) \leq \mu_{[m,n]}(t)$. It is evident that if $m \neq u^f(n)$, i.e., m is not the upstream node of n in the path P_f, then $\mu^f_{[m,n]}(t) = 0 \ \forall t$. In the following, we assume that there exists some $\mu_{max} < \infty$ such that $\mu_{[m,n]}(t) \leq \mu_{max}$ for all $[m,n] \in \mathcal{L}$, $\boldsymbol{S} \in \boldsymbol{\mathcal{S}}$ and $\boldsymbol{P} \in \mathcal{P}^{(\boldsymbol{S})}$.

C. Queueing Dynamics and Network Capacity Region

Let $\boldsymbol{Q}(t) = (Q^f_n(t))$, $f \in \mathcal{F}, n \in P_f$ and $t = 0, 1, 2, ...$ be the queue backlog vector of the network, in units of packets. [1] We assume the following queueing dynamics for all node $n \in P_f$ with $k^f(n) \leq K_f$:

$$Q^f_n(t+1) \leq \max[Q^f_n(t) - \mu^f_{[n,l^f(n)]}(t), 0] + \mu^f_{[u^f(n),n]}(t) \quad (1)$$

In the above equation we assume that when $k^f(n) = 1$, i.e., when node n is the source node of flow f, $\mu^f_{[u^f(n),n]}(t) = A_f(t)$ for all t. The inequality is due to the fact that the upstream node may not have enough packets to send. When $k^f(n) = K_f + 1$, i.e., $n = d_f$, we always assume that: $Q^f_n(t) = \mu^f_{[n,l^f(n)]}(t) = 0$ for all t. Throughout this paper, we assume the following notion of queue stability:

$$\overline{\boldsymbol{Q}} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{f,n} \mathbb{E}\{Q^f_n(\tau)\} < \infty. \quad (2)$$

Define $\Lambda \subset \mathbb{R}^M$ to be the *network capacity region*, which is the closure of all arrival rate vectors $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_M)^T$ for which under the routing and transmission configurations there exists a stabilizing control algorithm that ensures (2). The following theorem from [19] gives a useful characterization of the capacity region in our setting and will be useful in the following analysis.

***Theorem** 1:* The capacity region Λ is the set of arrival vectors $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_M)^T \in \mathbb{R}^M$ such that there exists a stationary randomized power allocation and scheduling algorithm that allocates power and flow rates purely as a function of $\boldsymbol{S}(t)$ and achieves:

$$\mathbb{E}\{\mu^f_{[n,l^f(n)]}(t)\} = \lambda_f, \ \forall \ f \in \mathcal{F}, n \in P_f : k^f(n) \leq K_f, \quad (3)$$

where the expectation is taken with respect to the random channel states and the (possibly) random power allocation and scheduling actions. □

In the following, we use $\boldsymbol{S}$-only policies to refer to stationary randomized policies that allocate power and flow rates purely as functions of the aggregate channel state $\boldsymbol{S}(t)$.

D. The Delay Efficient Scheduling Problem

Our goal is to find a joint power allocation and scheduling algorithm that, at every time slot, chooses the right power allocation vector and transmits the right amount of packets for each flow, so as to maintain queue stability of the network and to yield good delay performance. We will refer to this problem as the *Delay Efficient Scheduling Problem* (DESP).

This framework has been studied by many previous works, e.g., [1], [2]. It is also well known that the max-weight type algorithms in e.g., [2], [19], can be used to stabilize the network whenever the arrival rate vector is in the network capacity region. However, the best known delay bounds of the max-weight algorithms are not *order optimal*. Indeed, the delay performance of the max-weight type algorithms can be poor except for scheduling problems in downlink systems and some single hop networks, [5], [6].

III. Towards Better Delay Performance

In this section, we present the Delay-Efficient SCheduling algorithm (DESC). In the following, we first introduce the idea of an "Accelerating Queue" (AQ). We then use these AQs to develop the DESC algorithm.

A. Accelerating Queues and Redundant Constraints

In this subsection, we first define the notion of an *accelerating queue* (AQ). We then provide an example relating the AQs to redundant constraints in the optimization context.

For each flow f traversing a path $P_f = (n^f_1, n^f_2, ..., n^f_{K_f+1})$, we create K_f *accelerating queues* (AQ) $H^f_n(t)$ for $n \in \{n^f_1, n^f_2, ..., n^f_{K_f}\}$ that evolve as follows:

$$H^f_n(t+1) = \max\left[H^f_n(t) - \mu^f_{[n,l^f(n)]}(t), 0\right] + \theta A_f(t) + (1-\theta)\mu^f_{[u^f(n),n]}(t), \quad (4)$$

[1] Nodes that are not used by any flow are assumed to always have zero backlog for all flows.

where $\theta \in (0,1]$ is a parameter that is chosen independent of the network size and routing configurations. The choice of θ will be discussed in Section IV-B. Here we similarly define $\mu^f_{[u^f(n),n]}(t) = A_f(t)$ if $k^f(n) = 1$, i.e., $n = s_f$, and $H^f_n(t) = \mu^f_{[n,l^f(n)]}(t) = 0$ if $k^f(n) = K_f + 1$, i.e., $n = d_f$, for all f and t. These AQs are used to ensure that the instantaneous traffic arrival information is quickly sent to all the downstream nodes. We emphasize that these AQs are *virtual* queues and can easily be implemented in software. The actual queues in the system will still obey the queueing dynamic in (1). Note that (4) requires that the $A_f(t)$ values be sent to all intermediate nodes instantly, which can easily be done when centralized control is available. We will consider the case when this information propagation requires nonzero time in Section VI.

We now relate the AQs to redundant constraints in the optimization context. Consider the example shown in Fig. 1, where we try to support the three flows f_1, f_2, f_3. For simplicity assume all channels are static, i.e., no channel variation, and the arrivals are constant. At each time, rate needs to be allocated to each link $[m,n] \in \mathcal{L}$ for data transmission via power allocation. Assume that due to physical constraints, e.g., interference, the obtained rates are restricted to be within some feasible rate allocation set $\mathcal{X}$. Our goal is to find an algorithm to stabilize the network whenever possible.

Now let $\mu^{f_i}_{[m,n]}$ be the rate allocated to flow f_i over link $[m,n]$. It has been shown in [18] that a max-weight type algorithm applied to the above routing problem is closely related to solving the following optimization problem by a dual subgradient method.

$$\begin{aligned}(\mathbf{P1}) \quad \min \quad & 1 \\ s.t. \quad & \lambda_1 \le \mu^{f_1}_{[1,3]}, \quad \mu^{f_1}_{[1,3]} \le \mu^{f_1}_{[3,5]}, \\ & \lambda_2 \le \mu^{f_2}_{[2,3]}, \quad \mu^{f_2}_{[2,3]} \le \mu^{f_2}_{[3,5]}, \\ & \lambda_3 \le \mu^{f_3}_{[2,4]}, \\ & (\mu^{f_i}_{[m,n]},\, i=1,2,3,\, \forall [m,n])^T \in \mathcal{X}.\end{aligned}$$

Now consider modifying $(\mathbf{P1})$ by adding two *redundant* constraints as follows:

$$\begin{aligned}(\mathbf{P2}) \quad \min \quad & 1 \\ s.t. \quad & \lambda_1 \le \mu^{f_1}_{[1,3]}, \quad \mu^{f_1}_{[1,3]} \le \mu^{f_1}_{[3,5]}, \\ & \theta\lambda_1 + (1-\theta)\mu^{f_1}_{[1,3]} \le \mu^{f_1}_{[3,5]}, \qquad (5) \\ & \lambda_2 \le \mu^{f_2}_{[2,3]}, \quad \mu^{f_2}_{[2,3]} \le \mu^{f_2}_{[3,5]}, \\ & \theta\lambda_2 + (1-\theta)\mu^{f_2}_{[2,3]} \le \mu^{f_2}_{[3,5]}, \qquad (6) \\ & \lambda_3 \le \mu^{f_3}_{[2,4]}, \\ & (\mu^{f_i}_{[m,n]},\, i=1,2,3,\, \forall [m,n])^T \in \mathcal{X}.\end{aligned}$$

It is easy to see that the set of optimal solutions (possibly empty) of $(\mathbf{P1})$ and $(\mathbf{P2})$ are always the same. Also, it has been observed in the optimization context, e.g., Page 585 in [17], that adding these redundant constraints usually leads to faster convergence of the variables $\mu^{f_i}_{[m,n]}$ to their target values under gradient type methods, due to the fact that these additional constraints effectively "reduce" the search space of the optimal solution for the optimization methods.

Now suppose we try to solve $(\mathbf{P2})$ with a dual subgradient method and we assign to the redundant constraints (5) and (6) Lagrange multipliers $H_3^{f_1}$ and $H_3^{f_2}$. Then $H_3^{f_1}$ and $H_3^{f_2}$ will be updated according to:

$$H_3^{f_1}(t+1) = \left[H_3^{f_1}(t) - \mu^{f_1}_{[3,5]}(t)\right]^+ + \theta\lambda_1 + (1-\theta)\mu^{f_1}_{[1,3]}(t),$$
$$H_3^{f_2}(t+1) = \left[H_3^{f_2}(t) - \mu^{f_2}_{[3,5]}(t)\right]^+ + \theta\lambda_2 + (1-\theta)\mu^{f_2}_{[2,3]}(t),$$

where $[x]^+ = \max[x, 0]$ and $\mu^f_{[m,n]}(t)$ is obtained by solving some optimization problem. Compare these update rules with (4), we see that *the Accelerating Queues correspond exactly to the Lagrange multipliers of the redundant constraints*, and hence can be viewed as mimicking the functionality of the redundant constraints. Therefore, we expect that with the aid of the AQs, the $\mu^f_{[n,l^f(n)]}$ values will converge quickly to their target values under DESC, and hence the network delay will likely be small.

B. The DESC algorithm

In this section, we develop the Delay Efficient SCheduling algorithm (DESC) that will be applied to the DESP problem.

Delay Efficient SCheduling Algorithm (DESC): Choose a parameter $\theta \in (0,1]$ independent of the network size and routing configurations. At every time slot t, observe all queue values $Q^f_n(t)$ and $H^f_n(t)$, and do:

(1) Link Weight Computing: For all $[m,n] \in \mathcal{L}$ such that there exists a flow f with $m = u^f(n)$, find the flow $f^*_{[m,n]}$ such that (ties broken arbitrarily):

$$f^*_{[m,n]} = \arg\max_{f\in\mathcal{F}: m=u^f(n)} \Big\{ Q^f_m(t) - Q^f_n(t) + H^f_m(t) - (1-\theta)H^f_n(t) \Big\}. \qquad (7)$$

Then define the *weight* of the link $[m,n]$ to be:

$$W^*_{[m,n]}(t) = \max\Big[Q^{f^*_{[m,n]}}_m(t) - Q^{f^*_{[m,n]}}_n(t) + H^{f^*_{[m,n]}}_m(t) - (1-\theta)H^{f^*_{[m,n]}}_n(t), 0 \Big]. \qquad (8)$$

If no such f exists, i.e., $[m,n]$ is not used by any flow, define $W^*_{[m,n]}(t) = 0\ \forall\, t$.

(2) Power Allocation: Observe the aggregate channel state $\boldsymbol{S}(t)$, if $\boldsymbol{S}(t) = \boldsymbol{S}$, choose $\boldsymbol{P}(t) \in \mathcal{P}^{(\boldsymbol{S})}$ such that:

$$\boldsymbol{P}(t) = \arg\max_{\boldsymbol{P}\in\mathcal{P}^{(\boldsymbol{S})}} \sum_{[m,n]\in\mathcal{L}} \mu_{[m,n]}(t) W^*_{[m,n]}(t), \qquad (9)$$

where $\mu_{[m,n]}(t) = \Phi_{[m,n]}(\boldsymbol{S}, \boldsymbol{P}(t))$.

(3) Scheduling: For each link $[m,n] \in \mathcal{L}$, allocate the transmission rate as follows:

$$\mu^f_{[m,n]}(t) = \begin{cases} \mu_{[m,n]}(t) & \text{if } f = f^*_{[m,n]} \text{ and } W^*_{[m,n]}(t) > 0 \\ 0 & \text{else.} \end{cases}$$

That is, the full transmission rate over each link $[m,n]$ at time t is allocated to the flow $f^*_{[m,n]}$ that achieves the maximum positive weight $W^*_{[m,n]}(t)$ of the link. If $\mu^{f^*_{[m,n]}}_{[m,n]}(t) > Q^{f^*_{[m,n]}}_m(t)$, *null bits* are transmitted if needed.

(4) Queue Update: Update $Q_n^f(t)$ and $H_n^f(t)$, $\forall\, f, n \in P_f$, according to (1) and (4), respectively.

We note that DESC inherits almost all properties of the previous max-weight type algorithms: it does not require statistical knowledge of the arrival or channels, and it guarantees network stability whenever the arrival rate is inside the network capacity region. DESC is also not very difficult to implement. The link weights can easily be computed locally. The scheduling part can be done node-by-node. The AQs are virtual queues implemented in software and can easily be updated by message passing the information of $A_f(t)$ to all the nodes in its path, similar to the assumptions of the internet flow models in [20], [21]. Although DESC requires routing information, such information is usually not difficult to obtain in many contexts, e.g., the internet. The most complex part is the power allocation computation, which in general can be NP-hard. However, it can easily be solved in some special cases, e.g., when transmissions over different links do not interfere with each other, e.g., internet, or it can be easily approximated at the cost of some network capacity loss [19].

We finally note that DESC is similar to the DRPC algorithm in [2], in terms of power allocation and scheduling. Indeed, the major difference is the use of AQs in DESC. However, we will see that this simple distinction can lead to significant difference in algorithm performance.

IV. DESC: Stability and Delay Performance

Now we present the performance results of DESC and discuss their implications, using the DRPC algorithm in [2] as the benchmark algorithm for comparison. The proofs will be presented in Section V. Our first result characterizes the performance of DESC in terms of queue stability. In particular, we show that whenever the arrival rate vector is within the capacity region, both the actual queues and the AQs are stable. This result shows that the DESC algorithm is *throughput optimal*. Our second result concerns the difference between the aggregate packet arrival rates and the aggregate service rates allocated to the flows. This result, as we will see, states that under DESC, the service rates allocated to each flow over its path converge quickly to their desired rate values. We now have our first performance result of DESC.

***Theorem** 2:* Suppose there exists an $\epsilon > 0$ such that $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \Lambda$, then under the DESC algorithm, we have:

$$\overline{\sum_f \sum_{n\in P_f} \frac{Q_n^f(t)}{K_f}} \leq \frac{2\sum_f K_f B^2}{\epsilon} - \overline{\sum_f \sum_{n\in P_f} \frac{H_n^f(t)[\theta k^f(n) + (1-\theta)]}{K_f}}, \tag{10}$$

$$\overline{\sum_f \sum_{n\in P_f} H_n^f(t)} \leq \frac{2\sum_f K_f B^2}{\theta\epsilon} - \overline{\sum_f \frac{Q_{s_f}^f(t)}{\theta}}, \tag{11}$$

where $\mathbf{1}$ is the M-dimensional column vector with every entry equal to 1, $B = \max[A_{max}, \mu_{max}]$ and the notation $\overline{x(t)}$ is defined:

$$\overline{x(t)} = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{x(t)\}, \tag{12}$$

which denotes the expected time average of a sequence $\{x(t), t = 0, 1, 2, ...\}$. Here the expectation is taken over the randomness of the arrivals and link states.

Note here $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \Lambda$ means that $(\lambda_1 + \epsilon, \lambda_2 + \epsilon, ..., \lambda_M + \epsilon)^T \in \Lambda$. In other words, we can increase the arrival rates of *all* flows by the same amount ϵ and the rate vector is still supportable. Thus ϵ can be viewed as measuring how far is the current rate vector away from the boundary of the capacity region Λ. Also note that the bound (10) contains the parameters K_f on the left-hand side, which are the path lengths of the flows. This is due to the following: When analyzing the average total backlog size, one has to compare the drift under DESC with the drift under an $\boldsymbol{S}$-only strategy. To obtain bounds on the total network backlog, the $\boldsymbol{S}$-only strategy has to *simultaneously* ensure that for each flow f, its average output rate at every node $n \in P_f (n \neq d_f)$ is larger than its average input rate into that node by some $\delta_n^f > 0$. To guarantee such a stationary randomized policy exists, we need $\sum_{n\in P_f} \delta_n^f \leq \epsilon$. In our case, all δ_n^f are equal, hence $\delta_n^f = \epsilon/K_f$ and K_f appears on the left-hand side.

We note that the bound (10) has the potential to be strictly better than the bounds of the DRPC and EDRPC algorithms in [2]. Indeed, the corresponding congestion bound under DRPC can be shown to be:

$$\overline{\sum_f \sum_{n\in P_f} \frac{Q_n^f(t)}{K_f}} \leq \frac{\sum_f K_f B^2}{\epsilon}. \tag{13}$$

Hence if the second term in (10) is large, then (10) can be strictly better than (13). As we will see in the simulation section, the second term is indeed large and thus the bound in (10) can actually be better than (13) for DRPC.

We also note that the bound for the AQs in (11) is smaller than the bound (13) for the actual queues under DRPC roughly by a factor $\theta K_f/2$. Since the *AQ* sizes measure the convergence speed of the allocated rates under DESC and the *actual queue* sizes measure the convergence speed of the allocated rates under DRPC, we see from (11) and (13) that the allocated rates under DESC converge faster to their target values than under DRPC. To see this better, we define the following total *approximate network service lag* function $\text{Lag}(t)$ at time t as:

$$\begin{aligned}\text{Lag}(t) &= \sum_f \left[K_f A_f[0, t-1] - \sum_{n\in P_f} \mu^f_{[n,l^f(n)]}[0, t-1]\right] \\ &= \sum_f \left[K_f \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{n\in P_f} \sum_{\tau=0}^{t-1} \mu^f_{[n,l^f(n)]}(\tau)\right], \end{aligned} \tag{14}$$

where $A_f[0, t-1] = \sum_{\tau=0}^{t-1} A_f(\tau)$ and $\mu^f_{[n,l^f(n)]}[0, t-1] = \sum_{\tau=0}^{t-1} \mu^f_{[n,l^f(n)]}(\tau)$ are the total number of arrivals of flow f and the total amount of rate allocated to flow f over link $[n, l^f(n)]$ in the time period $[0, t-1]$. Since every arrival from $A_f(t)$ needs to be transmitted K_f times before reaching the destination, the quantity $\text{Lag}(t)$ can be viewed as approximating the total amount of "effective unserved data" that is currently in the network. Hence any delay efficient algorithm is expected to have a small time average value of

$\text{Lag}(t)$. The following theorem characterizes the performance of DESC with respect to the $\text{Lag}(t)$ metric.

Theorem 3: Suppose there exists an $\epsilon > 0$ such that $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \Lambda$, then under the DESC algorithm, we have:

$$\overline{\text{Lag}(t)} \quad \leq \quad \frac{2\sum_f K_f B^2}{\theta^2 \epsilon} - \overline{\sum_f \frac{Q^f_{s_f}(t)}{\theta^2}}, \tag{15}$$

where the notion $\overline{x(t)}$ is defined in (12).

Note that if $\sum_f K_f = O(N)$, e.g., $M = O(1)$, and $\epsilon = \Theta(1)$, then (15) implies that the average total service lag in the network grows only *linearly* in the network size.

A. Demonstrating Example

As a concrete demonstration of Theorems 2 and 3, we look at a simple example shown in Fig. 2, where a flow is going through an $N+1$ node tandem wireless network, with the source being node 1 and the destination being node $N+1$.

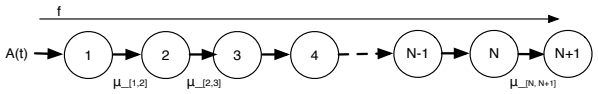

Fig. 2. A flow traversing a tandem.

In this case we see that $K_f = N$. Suppose we choose $\theta = 1$ in the DESC algorithm, then Theorems 2 and 3 state that under DESC, we have:

$$\overline{\sum_{i=1}^{N} Q_n(t)} \leq \frac{2N^2B^2}{\epsilon} - \overline{\sum_{n=1}^{N} nH_n(t)}, \tag{16}$$

$$\overline{\sum_{n=1}^{N} H_n(t)} \leq \frac{2NB^2}{\epsilon} - \overline{Q_1(t)}, \tag{17}$$

$$\overline{\text{Lag}(t)} = \overline{N\sum_{\tau=0}^{t-1} A(\tau) - \sum_{n=1}^{N}\sum_{\tau=0}^{t-1} \mu_{[n,n+1]}(\tau)} \leq \frac{2NB^2}{\epsilon}. \tag{18}$$

Suppose the network parameters are such that $\lambda = \Theta(1)$ and $\epsilon = \Theta(1)$. Then it is easy to see that the term $\overline{\sum_{n=1}^{N} nH_n(t)} = \Omega(N^2)$. By subtracting out this term, the bound in (16) can likely be small. (17) shows that the time average value of the AQs is $O(N)$, which implies that the rates allocated to the links converge in $O(N)$ time to their target values. (18) shows that the average total network service lag at all the nodes is no more than some $\Theta(N)$ constant, whereas the average service lag under DRPC will be $\Omega(N^2)$ in this case.

We will see in the simulation section that under the DESC algorithm, the time average actual queue backlog can indeed be $\Theta(N)$ when $\lambda = \Theta(1)$ and $\epsilon = \Theta(1)$.

B. Discussion on the Choice of θ

We note that the results in Theorems 2 and 3 hold for any $\theta \in (0, 1]$. Hence the bounds may be optimized by choosing the best θ. Intuitively, using a larger θ, e.g., $\theta = 1$ will lead to a faster convergence of the allocated rates. However, using a $\theta \neq 1$ may also be beneficial in some cases when we want to reduce the impact of the arrival information, for example, when the propagated traffic information may become noisy.

V. Performance Analysis

In this section we analyze the DESC algorithm by using the Lyapunov techniques developed in works [1] [2] [19]. To start, we first have the following lemma, which shows that if the current rate vector is strictly inside Λ, then we can find an $\boldsymbol{S}$-only policy that offers "perturbed" rates to all the nodes along the paths for all flows.

Lemma 1: Suppose there exists an $\epsilon > 0$ such that $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \Lambda$, then there exists an $\boldsymbol{S}$-only policy under which:

$$\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t)\big\} = \lambda_f + \delta^f_n, \quad \forall\, n \in P_f : k^f(n) \leq K_f \tag{19}$$

for any $-\lambda_n \leq \delta^f_n \leq \epsilon$ and for all $f \in \mathcal{F}$.

Proof: By Theorem 1, we see that if $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \Lambda$, then there exists an $\boldsymbol{S}$-only policy Π that achieves:

$$\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t)\big\} = \lambda_f + \epsilon, \quad \forall\, f, n \in P_f : k^f(n) \leq K_f.$$

Now we create a new $\boldsymbol{S}$-only policy Π' by modifying Π as follows. In every time slot, allocate rates to nodes using the policy Π. However, for each $n \in P_f$, in every time slot, transmit packets for flow f with the corresponding rate with probability $(\lambda_f + \delta^f_n)/(\lambda_f + \epsilon)$ (this is a valid probability since $-\lambda_n \leq \delta^f_n \leq \epsilon$). It is easy to see that Π' is an $\boldsymbol{S}$-only policy, and that (19) is satisfied under Π'. ■

We now prove the performance of DESC. To start, we first define the following Lyapunov function:

$$L(\boldsymbol{Q}(t), \boldsymbol{H}(t)) = \frac{1}{2}\sum_f \sum_{n\in P_f} \Big([Q^f_n(t)]^2 + [H^f_n(t)]^2\Big). \tag{20}$$

Denote $\boldsymbol{Z}(t) = (\boldsymbol{Q}(t), \boldsymbol{H}(t))$, and define the one-slot conditional *Lyapunov drift* to be:

$$\Delta(t) \quad = \quad \mathbb{E}\big\{L(t+1) - L(t) \mid \boldsymbol{Z}(t)\big\}, \tag{21}$$

where we use $L(t)$ as a short-hand notation for $L(\boldsymbol{Q}(t), \boldsymbol{H}(t))$. We have the following lemma:

Lemma 2: The drift $\Delta(t)$ defined in (21) satisfies:

$$\begin{aligned}\Delta(t) \leq C - \sum_f \sum_{n\in P_f} H^f_n(t)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) \\ -(1-\theta)\mu^f_{[u^f(n),n]}(t) - \theta A_f(t) \mid \boldsymbol{Z}(t)\big\} \\ -\sum_f \sum_{n\in P_f} Q^f_n(t)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t) \mid \boldsymbol{Z}(t)\big\}.\end{aligned} \tag{22}$$

where $C = 2\sum_f K_f B^2$.

Proof: See Appendix A. ■

We are now ready to prove Theorem 2. We will use the following theorem in [19].

Theorem 4: Let $\boldsymbol{Q}(t)$ be a vector process of queue backlogs that evolves according to some probability law, and let $L(\boldsymbol{Q}(t))$ be a non-negative function of $\boldsymbol{Q}(t)$. If there exists processes $f(t)$ and $g(t)$ and positive constants $a, b > 0$ such that at all time t, we have:

$$\Delta(t) \leq ag(t) - bf(t),$$

then:

$$b\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\big\{f(\tau)\big\} \leq a\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\big\{g(\tau)\big\}. \ \square$$

Proof: (Theorem 2) Rearranging the terms in (22), we obtain:

$$\Delta(t) \leq C + \sum_f \sum_{n\in P_f} H_n^f(t)\mathbb{E}\{\theta A_f(t) \mid \boldsymbol{Z}(t)\} + \sum_f \big[Q_{s_f}^f(t) + (1-\theta)H_{s_f}^f(t)\big]\mathbb{E}\{A_f(t) \mid \boldsymbol{Z}(t)\}, \quad (23)$$

$$-\sum_f \sum_{n\in P_f: k^f(n)\leq K_f} \mathbb{E}\{\mu_{[n,l^f(n)]}^f(t)\big[Q_n^f(t) - Q_{l^f(n)}^f(t) + H_n^f(t) - (1-\theta)H_{l^f(n)}^f(t)\big] \mid \boldsymbol{Z}(t)\}.$$

Now compare (23) and the DESC algorithm and recall that $\mu_{[m,n]}(t) = \Phi_{[m,n]}(\boldsymbol{S}(t), \boldsymbol{P}(t))$, we see that *the DESC algorithm chooses the power allocation vector and allocates transmission rates to flows every time slot to minimize the right-hand side (RHS) of the drift expression (23).* Because the RHS of (22) and (23) are equivalent, the drift value satisfies the following:

$$\Delta(t) \leq C - \sum_f \sum_{n\in P_f} H_n^f(t)\mathbb{E}\{\mu_{[n,l^f(n)]}^{*f}(t) - (1-\theta)\mu_{[u^f(n),n]}^{*f}(t) - \theta A_f(t) \mid \boldsymbol{Z}(t)\} - \sum_f \sum_{n\in P_f} Q_n^f(t)\mathbb{E}\{\mu_{[n,l^f(n)]}^{*f}(t) - \mu_{[u^f(n),n]}^{*f}(t) \mid \boldsymbol{Z}(t)\}, \quad (24)$$

where $\mu_{[n,l^f(n)]}^{*f}(t)$ corresponds to the rate allocated to Flow f on link $[n, l^f(n)]$ at time t by any other alternative algorithms.

Now since $\boldsymbol{\lambda}+\boldsymbol{\epsilon} \in \Lambda$, by Lemma 1, we see that there exists an $\boldsymbol{S}$-only policy that chooses the power allocation vector $\boldsymbol{P}(t)$ and allocates transmission rates $\mu_{[n,l^f(n)]}^f(t)$ purely as a function of the aggregate channel state $\boldsymbol{S}(t)$, and yields:

$$\mathbb{E}\{\mu_{[n,l^f(n)]}^{*f}(t) \mid \boldsymbol{Z}(t)\} = \lambda_f + \frac{k^f(n)\epsilon}{K_f}, \quad (25)$$

for all $f \in \mathcal{F}$ and $n \in P_f : k^f(n) \leq K_f$. Thus:

$$\mathbb{E}\{\mu_{[n,l^f(n)]}^{*f}(t)\} = \lambda_f + \frac{k^f(n)\epsilon}{K_f}. \quad (26)$$

Now plug this alternative algorithm into (24), we have:

$$\Delta(t) \leq C - \sum_f \sum_{n\in P_f} Q_n^f(t)\frac{\epsilon}{K_f} - \sum_f \sum_{n\in P_f} H_n^f(t)\frac{[\theta k^f(n) + (1-\theta)]\epsilon}{K_f}, \quad (27)$$

which by Theorem 4 implies:

$$\overline{\sum_f \sum_{n\in P_f}\left[\frac{Q_n^f(t)}{K_f} + \frac{H_n^f(t)[\theta k^f(n)+(1-\theta)]}{K_f}\right]} \leq \frac{C}{\epsilon} = \frac{2\sum_f K_f B^2}{\epsilon}. \quad (28)$$

Rearranging terms, we have:

$$\overline{\sum_f \sum_{n\in P_f} \frac{Q_n^f(t)}{K_f}} \leq \frac{2\sum_f K_f B^2}{\epsilon} - \overline{\sum_f \sum_{n\in P_f} \frac{H_n^f(t)[\theta k^f(n)+(1-\theta)]}{K_f}}. \quad (29)$$

This proves (10). Now similar as above, but plug in (24) another alternative $\boldsymbol{S}$-only policy that yields for all $f \in \mathcal{F}$:

$$\mathbb{E}\{\mu_{[n,l^f(n)]}^{*f}(t) \mid \boldsymbol{Z}(t)\} = \lambda_f + \epsilon,\ n \in P_f : k^f(n) \leq K_f. \quad (30)$$

Such an algorithm exists by Lemma 1. We then obtain:

$$\Delta(t) \leq C - \sum_f Q_{s_f}^f(t)\epsilon - \sum_f \sum_{n\in P_f} H_n^f(t)\theta\epsilon - \sum_f H_{s_f}^f(t)(1-\theta)\epsilon. \quad (31)$$

Using the fact that $H_{s_f}^f(t)(1-\theta)\epsilon \geq 0$, this implies that:

$$\overline{\sum_f \frac{Q_{s_f}^f(t)}{\theta}} + \overline{\sum_f \sum_{n\in P_f} H_n^f(t)} \leq \frac{2\sum_f K_f B^2}{\theta\epsilon} \quad (32)$$

Proving the theorem. ■

Now we prove Theorem 3:

Proof: (Theorem 3) For a flow $f \in \mathcal{F}$, let its path be $P_f = \{n_1, n_2, ..., n_{K_f+1}\}$. From (4), it is easy to show that for all t, we have [19]:

$$H_{n_1}^f(t) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1} \mu_{[n_1,n_2]}^f(\tau),$$

which implies:

$$\sum_{\tau=0}^{t-1} \mu_{[n_1,n_2]}^f(\tau) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - H_{n_1}^f(t). \quad (33)$$

Repeating the above for n_2, we have:

$$\begin{aligned} & H_{n_2}^f(t) \\ \geq\ & \sum_{\tau=0}^{t-1}[\theta A_f(\tau) + (1-\theta)\mu_{[n_1,n_2]}^f(\tau)] - \sum_{\tau=0}^{t-1}\mu_{[n_2,n_3]}^f(\tau) \\ \geq\ & \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu_{[n_2,n_3]}^f(\tau) - (1-\theta)H_{n_1}^f(t), \end{aligned}$$

where the second inequality follows from (33). Hence:

$$H_{n_2}^f(t) + (1-\theta)H_{n_1}^f(t) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu_{[n_2,n_3]}^f(\tau).$$

More generally, we have for all $i = 1, ..., K_f$ that:

$$\sum_{j=1}^{i}(1-\theta)^{i-j}H_{n_j}^f(t) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu_{[n_i,n_{i+1}]}^f(\tau).$$

Summing up all $i = 1, 2, ..., K_f$, we have:

$$\sum_{i=1}^{K_f}\sum_{j=1}^{i}(1-\theta)^{i-j}H_{n_j}^f(t) \geq \sum_{i=1}^{K_f}\left[\sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu_{[n_i,n_{i+1}]}^f(\tau)\right].$$

However:

$$\begin{aligned} \sum_{i=1}^{K_f}\sum_{j=1}^{i}(1-\theta)^{i-j}H_{n_j}^f(t) &= \sum_{i=1}^{K_f} H_{n_i}^f(t)\cdot\Big[\sum_{j=0}^{K_f-i}(1-\theta)^j\Big] \\ &\leq \frac{1}{\theta}\sum_{i=1}^{K_f} H_{n_i}^f(t), \end{aligned}$$

which implies:

$$\sum_{i=1}^{K_f}\left[\sum_{\tau=0}^{t-1}A_f(\tau)-\sum_{\tau=0}^{t-1}\mu^f_{[n_i,n_{i+1}]}(\tau)\right] \leq \frac{1}{\theta}\sum_{i=1}^{K_f}H^f_{n_i}(t).$$

Summing this over all $f\in\mathcal{F}$ and using (11) in Theorem 2 proves Theorem 3. ∎

VI. DESC UNDER DELAYED ARRIVAL INFORMATION

Here we consider the case when the time required to propagate the arrival information $A_f(t)$ is nonzero. Such a case can happen, for instance, when there is no central controller and thus message passing is required to propagate the $A_f(t)$ values. Let $D^f_n(t)$ be the delay (number of slots) to propagate the $A_f(t)$ information from s_f to node $n\in P_f$ at time t. We assume there exists a constant $D<\infty$ such that $D^f_n(t)\leq D$ for all f,n,t. Note that in this case, we can no longer use (4) to update the AQs due to the message passing delay. Instead, we modify the DESC algorithm to use the "delayed" traffic information. Specifically, we create a set of AQs using the "delayed" traffic information $A_f(t-D)$ as follows: For all $0\leq t<D$, let $H^f_n(t)=0$ and for all $t\geq D$, we update the AQs according to:

$$H^f_n(t+1)=\max\left[H^f_n(t)-\mu^f_{[n,l^f(n)]}(t),0\right] +\theta A_f(t-D)+(1-\theta)\mu^f_{[u^f(n),n]}(t). \quad (34)$$

We then define the following Delayed-DESC algorithm to perform power allocation and scheduling.

Delayed-DESC: Choose a parameter $\theta\in(0,1]$ independent of the network size and routing configurations. At every time slot, observe all queue values $Q^f_n(t-D)$ and $H^f_n(t)$, and do:

1) *Link Weight Computing:* For all $[m,n]\in\mathcal{L}$ such that there exists a flow f with $m=u^f(n)$, find the flow $f^*_{[m,n]}$ such that (ties broken arbitrarily):

$$f^*_{[m,n]}=\arg\max_{f\in\mathcal{F}:m=u^f(n)}\Big\{Q^f_m(t-D)-Q^f_n(t-D) +H^f_m(t)-(1-\theta)H^f_n(t)\Big\}. \quad (35)$$

   Then define the *weight* of the link $[m,n]$ to be:

$$W^*_{[m,n]}(t)=\max\Big[Q^{f^*_{[m,n]}}_m(t-D)-Q^{f^*_{[m,n]}}_n(t-D) +H^{f^*_{[m,n]}}_m(t)-(1-\theta)H^{f^*_{[m,n]}}_n(t),0\Big]. \quad (36)$$

   If no such f exists, i.e., $[m,n]$ is not used by any flow, define $W^*_{[m,n]}(t)=0\ \forall\, t$.
2) *Power Allocation* and *Scheduling* are the same as DESC except that the weights are now given by (36).
3) *Queue Update:* Update $Q^f_n(t)$ and $H^f_n(t)$ for all n,f according to (1) and (34), respectively.

Specifically, Delayed-DESC is the same as DESC except that at every time slot t, it uses $(\boldsymbol{Q}(t-D),\boldsymbol{H}(t))$ as the queue backlog vector to perform power and rate allocation, and the AQ values are updated according (34) instead of (4). The performance of Delayed-DESC is summarized in the following theorem.

***Theorem** 5:* Suppose there exists an $\epsilon>0$ such that $\boldsymbol{\lambda}+\mathbf{1}\epsilon\in\Lambda$, then under the Delayed-DESC algorithm with the parameter D, we have:

$$\overline{\sum_f\sum_{n\in P_f}\frac{Q^f_n(t)}{K_f}}\leq\frac{2\sum_f K_f(1+D)B^2}{\epsilon} -\overline{\sum_f\sum_{n\in P_f}\frac{H^f_n(t)[\theta k^f(n)+(1-\theta)\epsilon]}{K_f}},$$

$$\overline{\sum_f\sum_{n\in P_f}H^f_n(t)}\leq\frac{2\sum_f K_f(1+D)B^2}{\theta\epsilon}-\sum_f\overline{\frac{Q^f_{s_f}(t)}{\theta}},$$

$$\overline{\mathrm{Lag}(t)}\leq\frac{2\sum_f K_f(1+D)B^2}{\theta^2\epsilon}-\sum_f\overline{\frac{Q^f_{s_f}(t)}{\theta^2}},$$

where $\overline{x(t)}$ represents the expected time average of the sequence $\{x(t)\}_{t=0}^{\infty}$ and is defined in (12).

Note that since we typically only need a few bits to represent the $A_f(t)$ values, and we can pass these AQ information at a much faster time scale, i.e., not necessarily once per slot, the D value will typically be very small. In this case, Theorem 5 states that Delayed-DESC performs nearly as well as DESC.

Proof: (Theorem 5) Let $\hat{\boldsymbol{Z}}(t)=(\boldsymbol{Q}(t-D),\boldsymbol{H}(t))$ be the delayed queue state and define the drift to be:

$$\Delta(t)\triangleq\mathbb{E}\big\{L(t+1)-L(t)\mid\hat{\boldsymbol{Z}}(t)\big\},$$

where if $t-D<0$ we define $Q^f_n(t-D)=0$. Now using Lemma 2, we see that:

$$\Delta(t)\leq C-\sum_f\sum_{n\in P_f}H^f_n(t)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) -(1-\theta)\mu^f_{[u^f(n),n]}(t)-\theta A^f(t-D)\mid\hat{\boldsymbol{Z}}(t)\big\} -\sum_f\sum_{n\in P_f}\mathbb{E}\big\{Q^f_n(t)\big[\mu^f_{[n,l^f(n)]}(t)-\mu^f_{[u^f(n),n]}(t)\big]\mid\hat{\boldsymbol{Z}}(t)\big\}, \quad (37)$$

where $C=2\sum_f K_fB^2$. Denote the RHS of (37) as $\Delta_R(t)$. Using the fact that for all f and $n\in P_f$:

$$Q^f_n(t-D)-DB\leq Q^f_n(t)\leq Q^f_n(t-D)+DB,$$

we have:

$$\sum_f\sum_{n\in P_f}Q^f_n(t)\big[\mu^f_{[n,l^f(n)]}(t)-\mu^f_{[u^f(n),n]}(t)\big] \geq-\sum_f\sum_{n\in P_f}2DB^2 +\sum_f\sum_{n\in P_f}Q^f_n(t-D)\big[\mu^f_{[n,l^f(n)]}(t)-\mu^f_{[u^f(n),n]}(t)\big],$$

where $B=\max[A_{max},\mu_{max}]$. Plug this into (37), we get:

$$\Delta_R(t)\leq C+2\sum_f K_fDB^2 -\sum_f\sum_{n\in P_f}Q^f_n(t-D)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t)-\mu^f_{[u^f(n),n]}(t)\mid\hat{\boldsymbol{Z}}(t)\big\} -\sum_f\sum_{n\in P_f}H^f_n(t)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) -(1-\theta)\mu^f_{[u^f(n),n]}(t)-\theta A^f(t-D)\mid\hat{\boldsymbol{Z}}(t)\big\}. \quad (38)$$

We can similarly see that *the power and rate allocation of Delayed-DESC minimizes the RHS of (38)*. Hence the above inequality holds if we plug in any alternative power and rate allocation policy. Thus under Delayed-DESC, we have:

$$\Delta(t) \le C + 2\sum_f K_f D B^2 \tag{39}$$
$$-\sum_f \sum_{n\in P_f} Q_n^f(t-D)\mathbb{E}\{\mu^{*f}_{[n,l^f(n)]}(t) - \mu^{*f}_{[u^f(n),n]}(t) \mid \hat{\boldsymbol{Z}}(t)\}$$
$$-\sum_f \sum_{n\in P_f} H_n^f(t)\mathbb{E}\{\mu^{*f}_{[n,l^f(n)]}(t)$$
$$-(1-\theta)\mu^{*f}_{[u^f(n),n]}(t) - \theta A^f(t-D) \mid \hat{\boldsymbol{Z}}(t)\},$$

where $\mu^{*f}_{[n,l^f(n)]}(t)$ is the rate allocated to flow f over link $[n, l^f(n)]$ by any alternative policy. We can now carry out a similar argument as in the proof of Theorem 2 and prove the theorem. The details are omitted for brevity. ■

VII. Simulation

Here we provide simulation results of DESC. The network topology and flow configuration are shown in Fig. 3. We assume the channel conditions are independent and each link $[m, n]$ is i.i.d. every slot being ON with probability 0.8 and OFF with probability 0.2. When the channel is "ON," we can allocate one unit of power and transmit two packets; otherwise we can send zero packets. We further assume that all links can be activated without affecting others. However, a node can only transmit over one link at a time, though it can simultaneously receive packets from multiple nodes. Each flow f_i is an independent Bernoulli process with $A_{f_i}(t) = 2$ with probability $\lambda_i/2$ and $A_{f_i}(t) = 0$ else. The rate vector is given by $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T = (0.8, 0.4, 0.2, 0.6)^T$. We simulate the system with $(h = \frac{\eta}{2}, \upsilon = \frac{\eta}{2})$, where η, h and υ are parameters in Fig. 3. Note that in this case, $N = \frac{3\eta}{2} + 7$. The η value is chosen to be $\{10, 50, 100, 200, 500\}$. We use $\theta = 0.5$.

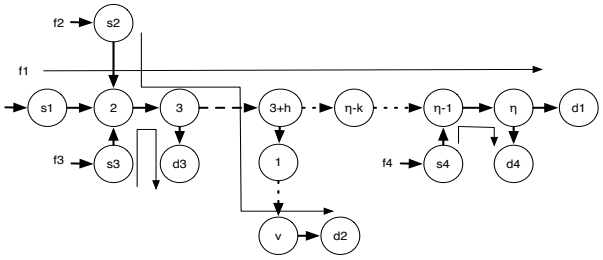

Fig. 3. A Network with 4 Flows. η is f_1's path length, h measures the path overlap length of f_1 and f_2, and υ is the vertical path length of f_2.

Fig. 4 and Fig. 5 show the simulation results. Fig. 4 shows that under DESC, the average total backlogs of Flow 1 and 2 scale only *linearly* in N. This is in contrast to the example provided in [8], which shows that the average backlog grows quadratically with N under the usual max-weight scheduling policy. We also see that the average total backlog of Flow 3 and 4 remains roughly the same. This is intuitive, as their path lengths do not grow with N. Interestingly, we observe that $\overline{\sum_{n\in P_f} H_n^f(t)} \ge \overline{\sum_{n\in P_f} Q_n^f(t)}$, $\forall\, f$. By equation (11) of Theorem 2, this implies that :

$$\overline{\sum_f \sum_{n\in P_f} Q_n^f(t)} \le \overline{\sum_f \sum_{n\in P_f} H_n^f(t)} \le \frac{2\sum_f K_f B^2}{\theta\epsilon}.$$

Since we also have $\sum_f K_f = O(N)$ and $\epsilon = \Theta(1)$ in this example, we see that indeed $\overline{\sum_f \sum_{n\in P_f} Q_n^f(t)} = O(N)$ in this case. This suggests that DESC can potentially be a way to achieve *delay-order-optimal* scheduling in general multihop networks. Fig. 5 shows that the total average rates allocated to Flow 1 and 2 over their paths, i.e., $\frac{1}{tK_{f_1}}\sum_{n_i\in P_{f_1}} \mu^{f_1}_{[n_i,l^{f_1}(n_i)]}[0, t-1]$ and $\frac{1}{tK_{f_2}}\sum_{n_j\in P_{f_2}} \mu^{f_2}_{[n_j,l^{f_2}(n_j)]}[0, t-1]$ with $\mu^{f_k}_{[n_i,l^{f_k}(n_i)]}[0, t-1] = \sum_{\tau=0}^{t-1} \mu^{f_k}_{[n_i,l^{f_k}(n_i)]}(\tau)$, converge quickly to above the actual average arrival rates i.e., $\frac{1}{t}A_f[0, t-1]$. Whereas the corresponding rates under DRPC converge very slowly from below to the actual average arrival rate. These plots suggest that the poor delay performance of many max-weight type algorithms in multihop networks can be due to slow convergence of the corresponding service rates.

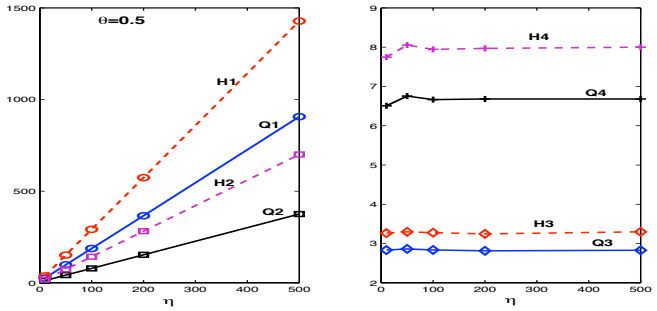

Fig. 4. Q_i and H_i, $i = 1, 2, 3, 4$, are the average total *actual* and *AQ* backlog sizes of flow i, respectively.

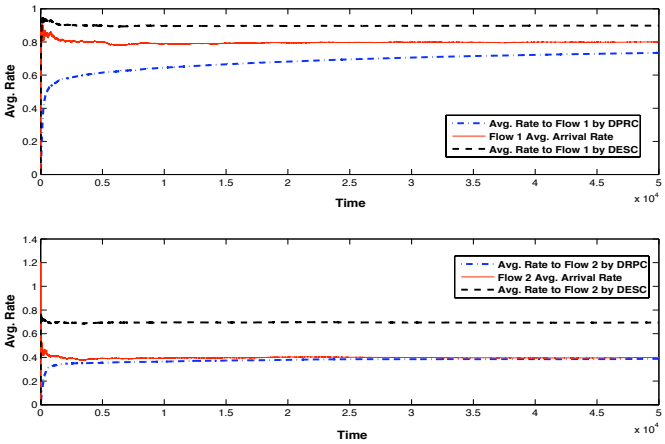

Fig. 5. UP: the average rate allocated to Flow 1 ($\eta = 100$); DOWN: the average rate allocated to Flow 2 ($\eta = 100$).

VIII. Conclusion

In this paper, we consider the problem of delay-efficient scheduling in general multihop networks. We develop a max-weight type Delay Efficient SCheduling Algorithm (DESC).

We show that DESC is throughput optimal and derive a queueing bound which can potentially be better than previous congestion bounds on max-weight type algorithms. We also show that under DESC, the time required for the allocated rates to converge to their target values scales only linearly in the network size. This contrasts with the usual max-weight algorithms, which typically require a time that is at least quadratic in the network size.

APPENDIX A – PROOF OF LEMMA 2

Proof: Define $B = \max[A_{max}, \mu_{max}]$ and denote $[x]^+ = \max[x, 0]$. From the queueing dynamic (1) we have for all node $n \in P_f$ with $k^f(n) \leq K_f$ that:

$$\begin{aligned}
&[Q_n^f(t+1)]^2 \\
&= \big([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+ + \mu^f_{[u^f(n),n]}(t)\big)^2 \\
&= \big([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+\big)^2 + [\mu^f_{[u^f(n),n]}(t)]^2 \\
&\qquad +2([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+)\mu^f_{[u^f(n),n]}(t) \\
&\leq [Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^2 + B^2 + 2Q_n^f(t)\mu^f_{[u^f(n),n]}(t).
\end{aligned}$$

The inequality holds since for any $x \in \mathbb{R}$, we have $([x]^+)^2 \leq x^2$, and $([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+)\mu^f_{[u^f(n),n]}(t) \leq Q_n^f(t)\mu^f_{[u^f(n),n]}(t)$. We also use in the above equation that if $k^f(n) = 1$, i.e., node n is the source node of flow f, then $\mu^f_{[u^f(n),n]}(t) = A_f(t)$ for all t. Thus by expanding the term $[Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^2$, we have for all $n \in P_f$ with $k^f(n) \leq K_f$ that:

$$\begin{aligned}
[Q_n^f(t+1)]^2 \leq\ & [Q_n^f(t)]^2 + 2B^2 \\
& -2Q_n^f(t)\big[\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t)\big].
\end{aligned}$$

Note that if $k^f(n) = K_f + 1$, i.e., $n = d_f$, we have $Q_n^f(t) = 0$ for all t, and so $[Q_n^f(t+1)]^2 - [Q_n^f(t)]^2 = 0$, $\forall t$. Summing the above over all n, f, and multiply by $\frac{1}{2}$, we have:

$$\begin{aligned}
&\frac{1}{2}\sum_f \sum_{n\in P_f} [Q_n^f(t+1)]^2 - \frac{1}{2}\sum_f \sum_{n\in P_f} [Q_n^f(t)]^2 \qquad (40)\\
&\quad \leq Y_1 B^2 - \sum_f \sum_{n\in P_f} Q_n^f(t)\big[\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t)\big],
\end{aligned}$$

where $Y_1 = \sum_f K_f$. Repeat the above argument on the term $\sum_f \sum_{n\in P_f} [H_n^f(t+1)]^2 - \sum_f \sum_{n\in P_f} [H_n^f(t)]^2$, we obtain:

$$\begin{aligned}
&\frac{1}{2}\sum_f \sum_{n\in P_f} [H_n^f(t+1)]^2 - \frac{1}{2}\sum_f \sum_{n\in P_f} [H_n^f(t)]^2 \qquad (41)\\
&\leq Y_1 B^2 - \sum_f \sum_{n\in P_f} H_n^f(t)[\mu^f_{[n,l^f(n)]}(t) - (1-\theta)\mu^f_{[u^f(n),n]}(t) \\
&\qquad\qquad -\theta A_f(t)].
\end{aligned}$$

Now adding (40) to (41), taking expectations conditioning on $\boldsymbol{Z}(t) = (\boldsymbol{Q}(t), \boldsymbol{H}(t))$, and letting $C = 2Y_1B^2 = 2\sum_f K_f B^2$ proves the lemma. ∎

REFERENCES

[1] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control, vol. 37, no. 12, pp. 1936-1949*, Dec. 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, Vol 23, NO.1*, January 2005.

[3] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory, Vol. 39, No. 2, pp. 466-478*, March1993.

[4] M. J. Neely. Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks. *IEEE/ACM Transactions on Networking, vol. 16, no. 5, pp. 1188-1199,*, October 2008.

[5] 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.

[6] M. J. Neely. Delay analysis for maximal scheduling with flow control in wireless networks with bursty traffic. *IEEE Transactions on Networking*, August 2009.

[7] G. R. Gupta and N. B. Shroff. Delay analysis for multi-hop wireless networks. *Proceedings of IEEE INFOCOM*, April 2009.

[8] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. *Proceedings of IEEE INFOCOM 2009 Mini-Conference*, April 2009.

[9] L. Ying, R. Srikant, and D. Towsley. Cluster-based back-pressure routing algorithm. *Proc. of IEEE INFOCOM*, April 2008.

[10] L. Ying, S. Shakkottai, and A. Reddy. On combining shortest-path and back-pressure routing over multihop wireless networks. *Proc. of IEEE INFOCOM*, April 2009.

[11] P. Gupta and T. Javidi. Towards throughput and delay-optimal routing for wireless ad-hoc networks. *Asilomar Conference on Signals, Systems and Computers*, Nov 2007.

[12] M. Naghshvar, H. Zhuang, and T. Javidi. A general class of throughput optimal routing policies in multi-hop wireless networks. *arXiv:0908.1273v1*.

[13] S. Jagabathula and D. Shah. Optimal delay scheduling in networks with arbitrary constraints. *In Proceedings of ACM SIGMETRIC/Performance,*, 2008.

[14] B. Sadiq and S. Baekand G. de Veciana. Delay-optimal opportunistic scheduling and approximations: The log rule. *Proceedings of IEEE INFOCOM*, April 2009.

[15] S. Shakkottai, R. Srikant, and A. Stolyar. Pathwise optimality of the exponential scheduling rule for wireless channels. *Advances in Applied Probability,*, December 2004.

[16] V. J. Venkataramanan and X. Lin. Structural properties of ldp for queue-length based wireless scheduling algorithms. *45th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois,*, September 2007.

[17] D. P. Bertsekas. *Nonlinear Programming, 2nd Edition*. Athena Scientific, 2003.

[18] L. Huang and M. J. Neely. Delay reduction via lagrange multipliers in stochastic network optimization. *Proc. of WiOpt, Seoul*, June 2009.

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

[20] Y. Li, A. Papachristodoulou, and M. Chiang. Stability of congestion control schemes with delay sensitive traffic. *Proc. IEEE ACC, Seattle, WA*, June 2008.

[21] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, vol. 7(6): 861-75, Dec. 1999.