# Super-Fast Delay Tradeoffs for Utility Optimal Fair Scheduling in Wireless Networks

Michael J. Neely

***Abstract*— We consider the fundamental delay tradeoffs for utility optimal scheduling in a general network with time varying channels. A network controller acts on randomly arriving data and makes flow control, routing, and resource allocation decisions to maximize a fairness metric based on a concave utility function of network throughput. A simple set of algorithms are constructed that yield total utility within $O(1/V)$ of the utility-optimal operating point, for any control parameter $V > 0$, with a corresponding end-to-end network delay that grows only logarithmically in $V$. This is the first algorithm to achieve such "super-fast" performance. Furthermore, we show that this is the best utility-delay tradeoff possible. This work demonstrates that the problem of maximizing throughput utility in a data network is fundamentally different than related problems of minimizing average power expenditure, as these latter problems cannot achieve such performance tradeoffs.**

***Index Terms*— Fairness, flow control, wireless networks, queueing analysis, optimization, delay, network capacity**

## I. Introduction

We consider the fundamental tradeoff between network utility and network delay for a wireless network with time varying channels. Traffic arrives to the network randomly, and we assume input rates exceed network capacity. Such a situation is typical for modern networks where growing user demands can quickly overload physical system resources. It is essential to establish simple solutions that maintain low network congestion and delay while providing fair access to all users. In this paper, we evaluate fairness according to a general concave utility function of the long term throughput of each user. The goal is to design a controller that drives total utility towards its maximum value, considering all possible methods of flow control, routing, and resource allocation, while ensuring an optimal tradeoff in network delay.

In our previous work on the network fairness problem, we constructed a set of algorithms indexed by a control parameter $V > 0$ that yield total network utility within $O(1/V)$ of the utility-optimal operating point, with a corresponding end-to-end delay tradeoff that is linear in $V$ [2] [3]. This result suggests that delay necessarily increases when utility is pushed toward optimality, although the existence of such a tradeoff and the form of the optimal utility-delay curve were left as open questions. In this paper we explore these questions and characterize the fundamental tradeoff curve. Specifically, we consider a particular class of *overloaded systems*, where user data rates strictly dominate the optimally fair operating point. We then develop a novel algorithm that deviates from the optimal utility by no more than $O(1/V)$ while ensuring that average network delay is less than or equal to $O(\log(V))$. Further, for the special case of one-hop networks, we prove that no algorithm can achieve a better asymptotic tradeoff. This establishes a fundamental relationship between utility and delay, and demonstrates the unexpected result that logarithmic delay is possible for systems with any concave utility metric.

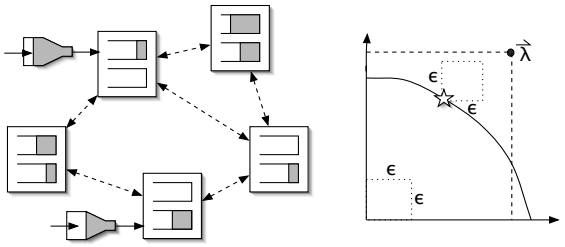


Fig. 1. An example network of 5 nodes, and an illustration of a capacity region (shown in two dimensions) with an input rate vector that strictly dominates the optimal operating point.

Related work in [5] considers the tradeoff between energy and delay for a single queue that stores data for transmission over a single fading channel. There, it is shown that any scheduling policy yielding average energy expenditure within $O(1/V)$ of the minimum energy required for stability must also have average queueing delay greater than or equal to $\Omega(\sqrt{V})$. Strategies for achieving this tradeoff are proposed in [5] using the concept of *buffer partitioning*, and a recent result in [33] shows that the same square-root tradeoff applies to the minimum energy problem for multi-user networks.

In this paper, we combine the technique of buffer partitioning with the recently developed technique of *performance optimal Lyapunov scheduling* [2] [3] [4] [34]. Specifically, in [2] [3] [4] [34] Lyapunov drift theorems are developed that treat stability and performance optimization simultaneously, leading to simple and robust control strategies. Here, we extend the theory to treat optimal utility-delay tradeoffs. The result is a novel set of Lyapunov scheduling algorithms that can be used for general networks, without requiring a-priori knowledge of traffic rates or channel statistics. The algorithms use weights that aggressively switch ON and OFF in order to achieve optimal delay tradeoffs. We find that the special structure of the long-term fairness objective allows for a "super-fast" logarithmic delay tradeoff that cannot be achieved in related problems of minimizing energy expenditure.

It is important to distinguish the tradeoffs we explore here to the capacity-delay tradeoffs recently explored for ad-hoc mobile networks in [2], [6]-[10]. These tradeoffs are quite

Michael J. Neely is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (email: mjneely AT usc.edu, web: http://www-rcf.usc.edu/∼mjneely).

This work was presented in part at the IEEE INFOCOM conference, Barcelona, Spain, April 2006.

This material is based upon work supported in part by the National Science Foundation under grant OCE 0520324.

different, in that they consider the "opposite end" of the performance curve. Indeed, in this paper we consider the impact of pushing network utility arbitrarily close to optimal, whereas the work in [2], [6]-[10] considers the opposite scenario where network utility (which is measured by throughput) is dramatically reduced with the goal of also reducing network delay. These antipodal ends of the tradeoff curve are conceptually different and involve completely different analytical methods.

Previous work in the area of utility-optimal scheduling is closely tied to the theory of convex optimization and Lagrangian duality. In [11], a network flow problem is formulated in terms of a network utility function, and a technique of introducing Lagrange multipliers as "shadow prices" was shown to offer distributed solution strategies. The relationship between convex duality theory and TCP-like congestion control algorithms is explored in [12]. Convex optimization approaches to static wireless network problems are considered in [13]-[21]. Stability problems for stochastic networks are treated in [22]-[28] using Lyapunov stability theory, and recent approaches to stability and performance optimization are considered in [29]-[32] using fluid models and/or stochastic gradient algorithms to transform a stochastic problem into one that is similar to a static problem. A detailed comparison between static gradient algorithms and stable Lyapunov scheduling is presented in [27] [2], and a Lyapunov method for performance optimization is developed in [2] [3] [4] [34] that yields results similar to those of stochastic gradient algorithms and also provides explicit performance and delay bounds.

The stochastic scheduling techniques we use in this paper are quite new, and go beyond the gradient algorithms suggested by classical optimization theory. The resulting utility-delay tradeoff that we achieve extends the field of stochastic optimal networking and demonstrates that significant performance gains are possible through simple scheduling policies.

An outline of this paper is as follows: In the next section we introduce the system model, emphasizing one-hop networks for simplicity. In Sections III and IV we specify the control objective and design the network control algorithm. Optimality of our $O(\log(V))$ delay result for one-hop networks is proven in Section V. Extensions to multi-hop networks are considered in Section VI.

## II. Problem Formulation

Consider a one-hop data network with $M$ links. The network operates in slotted time with timeslots $t \in \{0, 1, 2, \ldots\}$, and every timeslot data randomly enters the network. We define $A_m(t)$ as the amount of new data (in units of bits) that arrive for transmission over link $m$ during slot $t$ (for $m \in \{1, \ldots, M\}$). For simplicity of exposition, we assume arrivals are i.i.d. over timeslots, with arrival rates $\lambda_m = \mathbb{E}\{A_m(t)\}$. These arrival rates are not necessarily known to the network controller. Let $\vec{\lambda} = (\lambda_1, \ldots, \lambda_M)$ represent the vector of arrival rates for each link. This one-hop system model can be used, for example, to represent a single transmission node with $M$ downlink channels, a single access point with $M$ uplink channels, or a collection of distributed links in a multi-node ad-hoc network (as in Fig. 1). We assume throughout that $\lambda_m > 0$ for all sessions $m \in \{1, \ldots, M\}$.

### A. Flow Control

Data is not immediately admitted into the network. Rather, we assume that data from stream $A_m(t)$ is first placed into a distinct *transport layer storage reservoir* at its source node (see Fig. 1). Define $L_m(t)$ as the amount of data currently stored in the type-$m$ reservoir (for $m \in \{1, \ldots, M\}$). Every timeslot, a network controller for stream $m$ makes *flow control decisions* by choosing $R_m(t)$, the amount of type $m$ data allowed to enter the network on slot $t$. Note that $R_m(t) \leq A_m(t) + L_m(t)$. Any newly arriving data that is not admitted to the network is placed into the storage reservoir, or dropped if there is no room for storage. These transport layer storage buffers can either be infinite (so that no data is ever dropped), or finite (possibly zero). Of particular interest is the case of a "size-zero" reservoir, in which case any data that is not immediately admitted into the network is necessarily dropped.

### B. Resource and Rate Allocation

Admitted data from stream $m$ is stored in a network layer queue to await transmission over link $m$. Let $U_m(t)$ represent the amount of type-$m$ data (or *unfinished work*) in network queue $m$ at the beginning of slot $t$ (in units of bits), and let $\mu_m(t)$ represent the *transmission rate* over link $m$ during slot $t$ (in units of bits/slot).[1] The process $U_m(t)$ thus evolves according to the following queueing law:

$$U_m(t+1) = \max[U_m(t) - \mu_m(t), 0] + R_m(t) \qquad (1)$$

In wireless systems, the transmission rate might depend on resource allocation decisions (such as bandwidth or power allocation), and on time varying and uncontrollable channel conditions (due to environmental effects, fading, or user mobility). Hence, for a general system we define $\vec{S}(t) = (S_1(t), \ldots, S_M(t))$ as the *channel state vector* at time $t$, representing the uncontrollable properties of the channel that effect transmission. We assume that the number of channel states is finite, and that channel state vectors are i.i.d. over timeslots.[2] For each state vector $\vec{S}$, define $\Omega_{\vec{S}}$ as the compact set of all feasible transmission rate vectors $\vec{\mu} = (\mu_1, \ldots, \mu_M)$ available for resource allocation decisions when $\vec{S}(t) = \vec{S}$. Every timeslot, the network controller observes the current channel state vector $\vec{S}(t)$ and chooses a transmission rate vector $\vec{\mu}(t) = (\mu_1(t), \ldots, \mu_M(t))$ such that $\vec{\mu}(t) \in \Omega_{\vec{S}(t)}$. We assume each set $\Omega_{\vec{S}}$ has the property that if $\vec{\mu} \in \Omega_{\vec{S}}$ then setting any entry of $\vec{\mu}$ to 0 yields a vector that is also in $\Omega_{\vec{S}}$.

In networks with general inter-channel interference properties, this control decision may require full coordination of all transmission links. However, in cases where channels can be decoupled into a collection of $K$ independent sets, we have:

$$\Omega_{\vec{S}} = \Omega^1_{\vec{S}^1} \times \ldots \times \Omega^K_{\vec{S}^K} \qquad (2)$$

where $\vec{S}^k$ represents the channel states of data links within the $k$th independent set. In such a network, resource allocation decisions can be made separately within each set.

[1] It is often convenient to use units of *packets* and *packets/slot* in cases when data arrives according to fixed length packets and all transmission rates are integral multiples of the packet size.

[2] Extensions to non-i.i.d. systems can be treated via the methods in [2] [27].

Alternatively, subsets $\tilde{\Omega}_{\vec{S}}$ can be defined as the effective rate options available under a particular distributed multiple access structure specified in advance. Note that for networks without channel variations, $\Omega$ can represent the set of all *link activation vectors* available for scheduling decisions, as in [22].

## III. Network Capacity and the Control Objective

The *network capacity region* $\Lambda$ is defined as the closure of all arrival rate vectors $\vec{\lambda}$ that the network can stably support. Specifically, if we assume that the flow controllers allow all arriving data directly into the network, then $\vec{\lambda} \notin \Lambda$ implies that no resource allocation algorithm that meets the system constraints specified in the previous section can stabilize all queues of the network. However, if $\vec{\lambda}$ is strictly interior to $\Lambda$, then there must exist an algorithm that stabilizes the network and thereby achieves a throughput vector equal to the input rate vector $\vec{\lambda}$. The capacity region $\Lambda$ is described for a general multi-hop network in [2] [27] [34], from which the following special case result for one-hop networks immediately follows.

*Theorem 1:* (Capacity of One-Hop Networks) An input rate vector $\vec{\lambda}$ is in the capacity region $\Lambda$ if and only if there exists a stationary randomized resource allocation algorithm that chooses transmission rates $\vec{\mu}(t) \in \Omega_{\vec{S}(t)}$ based only on the current channel state $\vec{S}(t)$, and satisfies for all slots $t$:

$$\mathbb{E}\{\mu_m(t)\} = \lambda_m \text{ for all } m \in \{1, \ldots, M\} \quad (3)$$

The expectation in (3) is taken over the probability distribution of the current channel state vector $\vec{S}(t)$ and the distribution of the randomized resource allocation decisions that depend on $\vec{S}(t)$. Because channel states are i.i.d. from slot to slot, the above expectation is the same every timeslot, and does not depend on system history from previous slots. The above capacity region is always compact and convex [2].

### A. The Control Objective

Let $r_m$ represent the long term rate that data from stream $A_m(t)$ is delivered to the network by its flow controller, and let $\vec{r} = (r_1, \ldots, r_M)$. Clearly $r_m \leq \lambda_m$ for all $m$. Further, to ensure network stability we must have $\vec{r} \in \Lambda$. As a conventional measure of the total utility associated with supporting a traffic rate $r_m$, for each stream $m$ we define a *utility function* $g_m(r)$. Utility functions are assumed to be non-negative, non-decreasing, concave, and to have the property that $g_m(0) = 0$ for all $m$ (so that zero throughput for a given stream implies zero utility for that stream). Such utility functions are often used as a quantitative measure of *network fairness* [35] [11] [19] [17] [14] [30] [3] [2], and different choices of $g_m(r)$ lead to different fairness properties [36]. We further assume that all utility functions have finite right derivatives. The *optimal network utility* $g^*$ is defined as the solution of the following problem:

$$\begin{aligned} \text{Maximize:} \quad & \textstyle\sum_{m=1}^{M} g_m(r_m) \\ \text{Subject to:} \quad & \vec{r} \in \Lambda \\ & r_m \leq \lambda_m \quad \text{for all } m \in \{1, \ldots, M\} \end{aligned} \quad (4)$$

Note that no control algorithm that stabilizes the network can achieve an overall utility larger than $g^*$. The goal is to develop a joint algorithm for flow control, routing, and resource allocation that stabilizes the network and yields a total utility that can be pushed arbitrarily close to the optimal utility $g^*$, with a corresponding optimal tradeoff in end-to-end network delay.

Because the capacity region $\Lambda$ is compact and the utility functions are continuous, there exists a (potentially non-unique) optimal rate vector $\vec{r}^*$ such that $\sum_m g_m(r_m^*) = g^*$. We refer to such a rate vector as an *optimally fair operating point*. Note that if $\vec{\lambda}$ is strictly interior to the capacity region, then $\vec{r}^* = \vec{\lambda}$, and hence the exact optimal utility can be achieved simply by stabilizing the network. This can be accomplished with finite average delay (for both one-hop and multi-hop networks) using the DRPC algorithm of [2] [27] [34]. Hence, the notion of a fundamental utility-delay tradeoff only makes sense in the case when the input rate vector is either on the boundary or strictly outside of $\Lambda$.

In this paper, we focus on the case when the rate vector is strictly outside of $\Lambda$. More precisely, we shall assume there exists a positive value $\epsilon_{max}$ and an optimally fair operating point $\vec{r}^*$ such that $\lambda_m \geq r_m^* + \epsilon_{max}$ for all sessions $m \in \{1, \ldots, M\}$. Such a scenario occurs, for example, when all sessions have infinite storage reservoirs that are infinitely backlogged so that there is always data to send (as in [30] [29] [19] [11] [21]), or when input rates are sufficiently large so that the vector $\vec{\lambda}$ dominates the optimally fair operating point $\vec{r}^*$ in each entry (see Fig. 1). Under this assumption, in the next section we show that a "super-fast" logarithmic delay tradeoff is possible, improving upon the linear tradeoff shown to be achievable for all rate matrices in [3].

## IV. The Dynamic Control Algorithm

To motivate the control algorithm, first note that the optimization problem (4) is equivalent to the following:

$$\text{Maximize:} \quad \textstyle\sum_{m=1}^{M} g_m(\gamma_m) \quad (5)$$

$$\text{Subject to:} \quad r_m \geq \gamma_m \quad \text{for all } m \quad (6)$$

$$\vec{r} \in \Lambda \quad (7)$$

$$r_m \leq \lambda_m \quad \text{for all } m \quad (8)$$

The additional linear constraint (6) acts to decouple the throughput values $\vec{r}$ from the new optimization variables $\vec{\gamma}$. To build intuition, suppose we have a network algorithm controlling the system that stabilizes all queues and yields well defined time averages for throughput and transmission rates. Note that any such algorithm will have a throughput vector $\vec{r}$ that automatically satisfies (7) and (8). Now define $\overline{\mu}_m$ as the time average transmission rate offered over link $m$, and let $\hat{\mu}_m$ represent the time average rate that actual data is delivered over link $m$ (assuming for now that such time averages exist). Note that $\hat{\mu}_m \leq \overline{\mu}_m$, where strict inequality is possible due to "edge effects" that arise when the queue backlog $U_m(t)$ is frequently zero or near-zero so that the offered transmission rate $\mu_m(t)$ is under-utilized. The time average departure rate of bits from link $m$ must be less than or equal to the admitted input rate to this link, and hence:

$$r_m \geq \hat{\mu}_m \quad (9)$$

However, if we can by some means ensure that edge events arise very infrequently, then $\hat{\mu}_m \approx \overline{\mu}_m$, and hence for some small value $\delta > 0$ we have:

$$r_m + \delta \geq \overline{\mu}_m \tag{10}$$

Hence, if we can design a stabilizing control algorithm that maximizes (5), keeps queues far away from the edge region, and that satisfies the following constraints for all $m$:

$$\overline{\mu}_m \quad \geq \quad \gamma_m \tag{11}$$

then from (10) we have that the constraint (6) will be within $\delta$ of being satisfied, and so the resulting network utility will be close to the optimal utility $g^*$. Proximity to the optimal solution thus relies entirely on our ability to avoid edge effects. We note that this technique of indirectly satisfying inequality (6) through the inequality (11) is quite novel, as we cannot achieve "super-fast" delay tradeoffs by working directly with the inequality (6).

To design such a control policy, we use a novel combination of *stochastic optimal Lyapunov scheduling* (developed in [2] [3] [4] [34]), together with the concept of *buffer partitioning* from [5]. In particular, for a particular threshold parameter $Q > 0$ (to be determined later), we consider a policy that tends to decrease the queue backlog $U_m(t)$ when this backlog is greater than or equal to $Q$, and tends to *increase* queue backlog when $U_m(t) < Q$. This ensures stability while also keeping backlog sufficiently far from the edge region. However, choosing a large value of $Q$ will also directly increase average queue backlogs within the network. Actually, in this special case of *one-hop networks*, we find that it suffices to consider $Q$ as an absolute threshold, so that network queues $U_m(t)$ can be treated as having finite buffers of size $Q$. The multi-hop networking case (treated in Section VI) uses a related method that does not have finite buffer network queues.

### A. Lyapunov Functions and Virtual Queues

Let $\vec{U}(t) = (U_1(t), \ldots, U_M(t))$ represent the vector of current queue backlogs, and assume these queues have *maximum buffer size* $Q$, so that the queueing dynamics of (1) are modified as follows:

$$U_m(t+1) = \min[\max[U_m(t) - \mu_m(t), 0] + R_m(t), Q] \tag{12}$$

Thus, $U_m(t) \leq Q$ for all $t$. Any excess data admitted to the network layer according to the $R_m(t)$ process but which does not fit into the network layer queue $U_m(t)$ (due to the finite buffer constraint) is simply placed back into the transport layer storage reservoir, or dropped if there is no room for transport layer storage.

For given parameters $Q > 0$, $\omega > 0$ (to be determined later), we define the following non-negative *Lyapunov function*:

$$L(\vec{U}) \triangleq \sum_{m=1}^{M} \left[ e^{\omega(Q - U_m)} - 1 \right] \tag{13}$$

This Lyapunov function achieves its minimum value of $L(\vec{U}) = 0$ when $U_m = Q$ for all $m$, and increases exponentially when the backlog in any queue deviates from the $Q$ threshold. Minimizing the drift of this Lyapunov function from one timeslot to the next thus tends to maintain all queue backlogs near the $Q$ threshold, and we shall find that the resulting edge probabilities decay exponentially in $Q$.

Next, we must ensure that the constraints (11) are satisfied. To this end. we use the concept of a *virtual queue* developed in [4]. For each session $m \in \{1, \ldots, M\}$, we define a virtual queue $Z_m(t)$ with a dynamic update equation as follows:

$$Z_m(t+1) = \max[Z_m(t) - \mu_m(t), 0] + \gamma_m(t) \tag{14}$$

where $\mu_m(t)$ and $\gamma_m(t)$ are control decision variables used by the network controller. Note that the above update equation can be viewed as the dynamic equation of a discrete time queue with input process $\gamma_m(t)$ and server process $\mu_m(t)$. Now define time averages $\overline{\gamma}_m(t)$, $\overline{\mu}_m(t)$ as follows:

$$\overline{\gamma}_m(t) \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\gamma_m(\tau)\} \quad , \quad \overline{\mu}_m(t) \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_m(\tau)\}$$

Further define $\mu_{max}$ as the maximum possible transmission rate out of any link, considering all possible channel states and resource allocations.

*Lemma 1:* If the network controller stabilizes all virtual queues $Z_m(t)$, then:

$$\liminf_{t \to \infty} [\overline{\mu}_m(t) - \overline{\gamma}_m(t)] \geq 0 \tag{15}$$

*Proof:* The proof follows directly from the fact that if a queue with a bounded transmission rate is stable, then the $\liminf$ of the difference between the time average transmission rate and arrival rate is non-negative [37]. □

The above lemma ensures that stabilizing all virtual queues yields inequalities similar to (11).

### B. The Tradeoff-Optimal Control Policy

Assume that exogenous arrivals to any link are bounded by a value $A_{max}$ every slot, so that $A_m(t) \leq A_{max}$ for all $t$. Let $R_{max}$ be any value greater than or equal to $A_{max}$. Recall that $L_m(t)$ is the current data in transport reservoir $m$. We have the following control algorithm, defined in terms of positive constants $\omega, Q, V$ (to be specified later in Section IV-C).

*Utility-Delay Optimal Algorithm (UDOA One Hop):* Initialize all virtual queues so that $Z_m(0) = 0$ for all $m$. The control policy is decoupled into the following policies for flow control and resource allocation, implemented every timeslot:

- *Flow Control:* The flow controller for each link $m$ observes $U_m(t)$ and $Z_m(t)$ and makes these decisions:
  1) Choose $R_m(t) = \min[A_m(t) + L_m(t), R_{max}]$.
  2) Choose $\gamma_m(t) = \gamma_m$, where $\gamma_m$ solves:

$$\begin{aligned} \text{Maximize:} \quad & V g_m(\gamma_m) - 2Z_m(t)\gamma_m \\ \text{Subject to:} \quad & 0 \leq \gamma_m \leq R_{max} \end{aligned}$$

The virtual queues $Z_m(t)$ are then updated according to (14), using the $\gamma_m(t)$ values computed above and the $\mu_m(t)$ values computed by the following resource allocation algorithm. The actual queues $U_m(t)$ are updated according to the finite buffer queueing law (12). Note that any admitted data that would cause the network layer

queue to exceed its buffer constraint $Q$ is placed back in the transport layer reservoir, or dropped if there is no room for storage.

- *Resource Allocation:* Every timeslot $t$ the network controllers observe the current channel state vector $\vec{S}(t)$ and allocate transmission rates $\vec{\mu}(t) = (\mu_1(t), \ldots, \mu_m(t))$, where $\vec{\mu}(t)$ solves the following optimization problem:

$$\begin{array}{ll} \text{Maximize:} & \sum_{m=1}^{M} W_m(t)\mu_m \\ \text{Subject to:} & \vec{\mu} \in \Omega_{\vec{S}(t)} \end{array}$$

where: $W_m(t) \triangleq - e^{\omega(Q-U_m(t))} + \frac{2}{\omega} Z_m(t)$

The flow control and resource allocation layers are decoupled, but the algorithms effect each other through key parameters that are passed between layers. The flow control algorithm can be implemented separately at each link using only queue length information of that link. Note that the computation of $\gamma_m(t)$ for each link $m$ is a simple convex optimization of one variable, and can easily be solved in real time for any concave utility function.

Note that the resource allocation maximizes a weighted sum of instantaneous transmission rates, as in [22] [25] [26] [27]. It may be difficult to precisely implement the resource allocation algorithm when links are distributed over a multi-node network, although distributed implementations exist when links can be grouped into independent sets as in (2), and distributed approximations can be implemented (often to within a constant factor) using methods similar to those given in [34] [38] [39] [40] [2] [27]. The particular weights $W_m(t)$ above include the virtual queues $Z_m(t)$, and are designed to complement the flow control decisions. Below we show that the algorithm yields a "super-fast" utility-delay tradeoff curve.

### C. Algorithm Performance

Let $\vec{r}^*$ represent an optimally fair operating point that solves (4). We assume that the arrival rate vector $\vec{\lambda}$ *strictly dominates the optimally fair operating point* $\vec{r}^*$. Specifically, we define $\epsilon_{max}$ as the largest value of $\epsilon$ that satisfies $\epsilon \leq \min[\lambda_m/2, \lambda_m - r_m^*]$ for all $m \in \{1, \ldots, M\}$, and assume that $\epsilon_{max} > 0$ (see Fig. 1). To analyze the UDOA algorithm, it is useful to define $\delta_{max}$ as the largest possible change in the individual queue backlog at any link during a timeslot:

$$\delta_{max} \triangleq \max[R_{max}, \mu_{max}]$$

Define time averages $\overline{U}_m(t)$ and $\overline{r}_m(t)$ as follows:

$$\overline{U}_m(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{U_m(\tau)\right\} \;\;,\;\; \overline{r}_m(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{R_m^{admit}(\tau)\right\}$$

where $R_m^{admit}(\tau)$ is the amount of data *actually admitted* to the network layer queue $m$ during slot $t$ (which is equal to $R_m(t)$ minus the *excess* data (if any) that did not fit into the network layer queue $U_m(t)$ due to the finite buffer size $Q$)).

*Theorem 2:* (One-Hop UDOA Performance) Fix parameters $V$, $\epsilon$ such that $V > 0$ and $0 < \epsilon \leq \epsilon_{max}$, and choose any positive value $\omega$ that satisfies:

$$\omega \delta_{max} e^{\omega \delta_{max}} \leq 2\epsilon/\delta_{max} \tag{16}$$

Further define:[3]

$$Q \triangleq \frac{2}{\omega}\log(V) \tag{17}$$

Then the UDOA algorithm stabilizes all actual and virtual queues of the system, and yields:

$$\begin{aligned} U_m(t) \leq Q &= O\left(\log(V)\right) \text{ for all } t \\ \liminf_{t\to\infty} \sum_m g_m\left(\overline{r}_m(t)\right) &\geq g^* - O(1/V) \end{aligned}$$

*Proof:* See Appendix A. □

Because $V$ is an arbitrary control parameter, it can be made as large as desired, pushing total system utility arbitrarily close to the optimal utility $g^*$ with a corresponding logarithmic increase in average congestion (and hence, by Little's Theorem, average delay). Note that this performance holds *regardless of the transport layer reservoir buffer size*. Thus, a non-zero reservoir buffer does not contribute to achieving these super-fast tradeoffs. However, large transport layer reservoirs can be used in practice to preserve data from individual streams and maintain FIFO packet admission.

Note that if the $\omega$ parameter is chosen as follows:

$$\omega \triangleq \frac{2\epsilon}{\delta_{max}^2} e^{-2\epsilon/\delta_{max}}$$

then the inequality (16) is necessarily satisfied. This follows directly from the fact that for any positive value $c$, the inequality $xe^x \leq c$ is satisfied by the variable $x = ce^{-c}$.

## V. Optimality of the Logarithmic Tradeoff

Here we show that it is not possible to improve upon the $O(\log(V))$ delay characteristic, and hence our scheduling algorithm captures the tightest tradeoff. We again consider a one-hop network with $M$ links. We further assume the network has the following characteristics:

1) Flow control reservoirs have zero buffer space, so that admission/rejection decisions are made immediately upon packet arrival.[4]
2) Arrivals are i.i.d. over timeslots and independent of channel states, and there exists a probability $p > 0$ that no new packets arrive from *any* data stream.
3) Channel states are i.i.d. over timeslots, and there exists a rate $\theta$ and a probability $q$ such that: For each link $m$, with probability at least $q$ a channel state arises that would allow link $m$ to transmit with rate at least $\theta$ (if all resources were to be completely devoted to link $m$).
4) If it is possible to transmit with rates $(\mu_1(t), \ldots, \mu_M(t))$ during slot $t$, then it is also possible to transmit with rates $(\tilde{\mu}_1(t), \ldots, \tilde{\mu}_M(t))$ during slot $t$, where $\tilde{\mu}_m(t) \leq \mu_m(t)$ for all $m \in \{1, \ldots, M\}$.

Assume that utility functions $g_m(r)$ are differentiable, strictly increasing, and concave, and that the input rate vector $\vec{\lambda}$ is finite and outside of the capacity region. Recall that $\mu_{max}$ is the maximum possible transmission rate, and define $\beta \triangleq \min_{m\in\{1,\ldots,M\}} \frac{dg_m(\mu_{max})}{dr}$, which is a lower bound on the

[3] All $\log(\cdot)$ functions in this paper denote the natural logarithm.

[4] The same result can be proven when all transport layer flow control reservoirs have finite buffers, but the result is not true in the case when reservoirs have infinite buffers.

minimum possible slope of any utility function over the rate region of interest. Note that our assumptions imply that $\beta > 0$. Further, we restrict attention to the class of scheduling policies that are ergodic with well defined steady state averages.

*Theorem 3:* If a control policy of the type described above yields a total utility that differs from the optimal utility by no more than $1/V$, then average congestion must be greater than or equal to $\Omega(\log(V))$.

*Proof:* Consider an ergodic control policy that yields total throughput within $1/V$ of the optimal throughput $\sum_m g_m(r_m^*)$, where $(r_1^*, \ldots, r_M^*)$ solves (4). Let $(\overline{r}_1, \ldots, \overline{r}_M)$ represent the throughput vector achieved by this policy. Thus:

$$1/V \geq \sum_{m=1}^{M} g_m(r_m^*) - \sum_{m=1}^{M} g_m(\overline{r}_m) \qquad (18)$$

Note that $(\overline{r}_1, \ldots, \overline{r}_M) \in \Lambda$, and $\overline{r}_m \leq \lambda_m$ for all $m$. Because $\vec{\lambda} \notin \Lambda$, there must be a link $k \in \{1, \ldots, M\}$ such that $\lambda_k > \overline{r}_k$. Define $\tilde{\theta} = \min[\theta, (\lambda_k - \overline{r}_k)]$.

Let $\overline{U}$ denote the total average bit occupancy in the system. Define $T = 2\lceil 2\overline{U}/\tilde{\theta} + M \rceil$. Let $U_1(t), \ldots, U_M(t)$ represent the queue backlogs in a particular slot $t$. We say that a $T$ slot interval beginning at time $t$ is an *edge interval* if the following sequence of events occurs:

- The total queue backlog $U_1(t) + \ldots + U_M(t)$ in the system at time $t$ is less than or equal to $2\overline{U}$.
- There are no new arrivals to any queue of the system during the $T$ slot interval.
- For the first $\lceil U_1(t)/\tilde{\theta} \rceil$ slots, channel states arise that would allow link 1 to transmit at rate at least $\theta$. For the next $\lceil U_2(t)/\tilde{\theta} \rceil$ slots, channel states arise that would allow link 2 to transmit at rate at least $\theta$, and so forth, so that all links sequentially would be able to transmit their full starting load, taking up a total of $\lceil U_1(t)/\tilde{\theta} \rceil + \ldots + \lceil U_M(t)/\tilde{\theta} \rceil$ timeslots. This is called the *first phase* of the interval. For all remaining timeslots up to $T$, channel states arise that would allow the special link $k$ to transmit at rate at least $\theta$.

Assuming the system is in steady state at the beginning of an edge interval, by the Markov inequality for non-negative random variables we have:

$$Pr[\textstyle\sum_m U_m(t) \leq 2\overline{U}] \geq 1/2$$

Let $\delta$ represent the steady state probability of a particular set of $T$ slots being an edge interval. We thus have:

$$\delta \geq \frac{1}{2}(pq)^T \geq \frac{1}{2}(pq)^{(4\overline{U}/\tilde{\theta} + 2M + 2)} \qquad (19)$$

Note that the number of timeslots in the first phase of an edge interval satisfies:

$$\begin{aligned} \sum_{m=1}^{M} \lceil U_m(t)/\tilde{\theta} \rceil &\leq \sum_{m=1}^{M} (U_m(t)/\tilde{\theta} + 1) \\ &\leq 2\overline{U}/\tilde{\theta} + M \\ &\leq T/2 \end{aligned}$$

Hence, this phase takes up at most half of the slots of an edge interval. Further, if the controller knew in advance that this was an edge interval, it could clearly schedule so that all initial backlog is cleared during this first phase. In particular, it is possible to make a sequence of transmission decisions that yields a per-slot empirical average transmission rate during the first $T/2$ slots that is exactly equal to the per-slot empirical average throughput of the actual control policy over the full $T$ slots.

Consider now an alternate transmission strategy that runs on a "parallel" system with the same channel states: The timeline $t \in \{0, 1, 2, \ldots\}$ is partitioned into successive disjoint intervals of $T$ slots. If the current interval is an edge interval, during the first phase we transmit to yield a per-slot empirical average transmission rate exactly equal to the per-slot empirical average throughput attained by the actual control policy over the full $T$ slots. During the second phase, we remain idle until slot $T/2$, after which we transmit only over link $k$, with an exact transmission rate of $\tilde{\theta}$ until the $T$ slot interval expires. If the current interval is *not* an edge interval, each link transmits with a rate exactly equal to the amount of bits delivered over the link on that timeslot by the original policy. Note that this alternate transmission strategy is *non-causal* as it requires knowledge of future events to make the transmission decisions. However, the capacity region of the system contains all long term average transmission rate vectors achieved by either causal or non-causal policies [2] [27], and hence the resulting time average transmission rate vector $(\hat{r}_1, \ldots, \hat{r}_M)$ achieved by this alternate policy is within the capacity region $\Lambda$.

Further note that:

$$\hat{r}_m = \overline{r}_m \quad \text{for all } m \neq k \qquad (20)$$

$$\hat{r}_k = \overline{r}_k + \delta\tilde{\theta}/2 \qquad (21)$$

where the second equality follows because, on any edge interval, the alternate policy yields a total average transmission rate on link $k$ that is exactly $\tilde{\theta}/2$ beyond the link $k$ average throughput of the control policy over this interval. From (20) it follows that $\hat{r}_m \leq \lambda_m$ for all $m \neq k$. From (21) it follows that $\hat{r}_k \leq \overline{r}_k + \delta \min[\theta, (\lambda_k - \overline{r}_k)] \leq \overline{r}_k + (\lambda_k - \overline{r}_k)$ and so $\hat{r}_k \leq \lambda_k$.

It follows that $(\hat{r}_1, \ldots, \hat{r}_M)$ satisfies the constraints of the optimization problem (4), and so its utility is less than or equal to the optimal utility. Hence, from (18):

$$\begin{aligned} 1/V &\geq \sum_{m=1}^{M} g_m(r_m^*) - \sum_{m=1}^{M} g_m(\overline{r}_m) \\ &\geq \sum_{m=1}^{M} g_m(\hat{r}_m) - \sum_{m=1}^{M} g_m(\overline{r}_m) \\ &= g_k(\overline{r}_k + \delta\tilde{\theta}/2) - g_k(\overline{r}_k) \end{aligned}$$

where the last line follows by (20) and (21). Using the definition of $\beta$, we have:

$$1/V \geq \beta\delta\tilde{\theta}/2$$

Using the inequality (19), we have:

$$\frac{1}{V} \geq \frac{1}{4}\beta\tilde{\theta}(pq)^{(4\overline{U}/\tilde{\theta} + 2M + 2)} \qquad (22)$$

Taking the logarithm of both sides and shifting terms yields:

$$\overline{U} \geq \frac{\tilde{\theta}}{4}\left(\frac{\log(V\beta\tilde{\theta}/4)}{\log(1/(pq))} - 2M - 2\right)$$

Hence, average backlog grows at least logarithmically in $V$. □

## VI. Multi-Hop Networks

Consider now a multi-hop network with $N$ nodes. Data arrives randomly at each node and must be delivered to a specific other node. We define all data destined for a particular node $c \in \{1, \ldots, N\}$ to be *commodity c data*, regardless of its source. Let $A_n^{(c)}(t)$ represent the amount of new commodity $c$ bits that exogenously enter node $n$ during timeslot $t$. The $A_n^{(c)}(t)$ process is assumed to be i.i.d. over slots with rate $\lambda_n^{(c)} = \mathbb{E}\left\{A_n^{(c)}(t)\right\}$. Let $\underline{\lambda} = (\lambda_n^{(c)})$ represent the *arrival rate matrix*. As before, we assume all data is first placed into a transport layer storage reservoir, and we let $L_n^{(c)}(t)$ and $R_n^{(c)}(t)$ represent the current commodity $c$ data in the reservoir at node $n$, and the amount admitted to the network layer at node $n$, respectively. Flow control decisions must satisfy the constraints $R_n^{(c)}(t) \leq A_n^{(c)}(t) + L_n^{(c)}(t)$ for all $t$.

Let $\underline{S}(t) = (S_{ab}(t))$ represent the *channel state matrix*, specifying the state of all potential links $(a, b)$ between each node pair $(a, b)$. Let $\underline{\mu}(t) = (\mu_{ab}(t))$ represent the current *transition rate matrix*, chosen such that $\underline{\mu}(t) \in \Omega_{\underline{S}(t)}$, where $\Omega_{\underline{S}(t)}$ specifies the set of all feasible transmission rate matrix options under channel state $\underline{S}(t)$.

Let $\mu_{ab}^{(c)}(t)$ represent *routing control variables*, specifying the transmission rate offered to commodity $c$ data over link $(a, b)$ during slot $t$. These new control variables satisfy the following constraints: $\sum_c \mu_{ab}^{(c)}(t) \leq \mu_{ab}(t)$ for all $(a, b)$, and $\mu_{ab}^{(c)}(t) = 0$ if $(a, b) \notin \mathcal{L}_c$, where $\mathcal{L}_c$ denotes the set of all links acceptable for commodity $c$ data to traverse. Note that such $\mathcal{L}_c$ sets can also model one-hop networks by constraining each traffic session to its appropriate single-hop link. Let $U_n^{(c)}(t)$ represent the amount of commodity $c$ bits currently stored in network node $n$ during slot $t$. The queueing dynamics satisfy:

$$U_n^{(c)}(t+1) \leq \max[U_n^{(c)}(t) - \sum_b \mu_{nb}^{(c)}(t), 0] + R_n^{(c)}(t) + \sum_a \mu_{an}^{(c)}(t) \quad (23)$$

Thus, each network queue can receive both exogenous and endogenous data. Note that $\mu_{ab}^{(c)}(t)$ may be less than the actual commodity $c$ data transmitted over link $(a, b)$ during slot $t$ if there is not enough of this commodity to transmit. We assume that $U_n^{(n)}(t) = 0$ for all $t$, as data that reaches its destination is removed from the network. Further, let $\mathcal{K}$ represent the set of all $(n, c)$ pairs for which it is possible to have a non-zero $U_n^{(c)}(t)$ value, and let $|\mathcal{K}|$ denote the number of such pairs. We assume that $\lambda_n^{(c)} = 0$ whenever $(n, c) \notin \mathcal{K}$.

The network capacity region $\Lambda$, consisting of all stabilizable rate matrices $\underline{\lambda}$, is described in [2] [27] [34]. Our goal is to support an optimally fair throughput matrix, where optimality is defined according to utility functions $g_n^{(c)}(r)$. Specifically, we seek to maximize $\sum_{(n,c)\in\mathcal{K}} g_n^{(c)}(r_n^{(c)})$ subject to $(r_n^{(c)}) \in \Lambda$ and $r_n^{(c)} \leq \lambda_n^{(c)}$ for all $(n, c) \in \mathcal{K}$. Let $(r_n^{*(c)})$ represent an *optimally fair throughput matrix* that solves this problem, and define $g^* \triangleq \sum_{(n,c)\in\mathcal{K}} g_n^{(c)}(r_n^{*(c)})$. As in the one-hop network problem, we use auxiliary variables $\gamma_n^{(c)}$ and modify the optimization problem to maximizing $\sum_{(n,c)\in\mathcal{K}} g_n^{(c)}(\gamma_n^{(c)})$, with a new constraint $r_n^{(c)} \geq \gamma_n^{(c)}$. Analogous to the one-hop network inequality (9), we note that: $r_n^{(c)} + \sum_a \overline{\mu}_{an}^{(c)} \geq \sum_b \hat{\mu}_{nb}^{(c)}$, where $\overline{\mu}_{ab}^{(c)}$ and $\hat{\mu}_{ab}^{(c)}$ represent the time average transmission rate offered to commodity $c$ data over link $(a, b)$, and the time average rate of actual commodity $c$ data transfered over this link, respectively. If edge events are rare, then $\overline{\mu}_{ab}^{(c)} \approx \hat{\mu}_{ab}^{(c)}$, and the constraint $r_n^{(c)} \geq \gamma_n^{(c)}$ is close to being satisfied provided that we ensure for all $(n, c) \in \mathcal{K}$:

$$\sum_b \overline{\mu}_{nb}^{(c)} - \sum_a \overline{\mu}_{an}^{(c)} \quad \geq \quad \gamma_n^{(c)}$$

which is analogous to the one-hop constraint (11). To satisfy this constraint, we define virtual queues $Z_n^{(c)}(t)$ for all $(n, c) \in \mathcal{K}$, where $Z_n^{(c)}(0) = 0$ and evolves as follows:

$$Z_n^{(c)}(t+1) = \max[Z_n^{(c)}(t) - \sum_b \mu_{nb}^{(c)}(t), 0] + \gamma_n^{(c)}(t) + \sum_a \mu_{an}^{(c)}(t) \quad (24)$$

### A. Multi-Hop Assumptions for $O(\log(V))$ Delay Tradeoffs

To establish $O(\log(V))$ delay tradeoffs, it is important to make the following *fully active* assumption: If $(n, c) \in \mathcal{K}$, then session $A_n^{(c)}(t)$ is active with a non-zero input rate $\lambda_n^{(c)}$. That is, if it is possible to send commodity $c$ data to node $n$ (where $n \neq c$), then node $n$ also has its own independent input stream of commodity $c$ data. This is a reasonable assumption in the case of one-hop networks, as discussed in earlier sections, as the network can be defined in terms of only those links that are active. This assumption also applies to multi-hop networks for which there are one or more commodities, and all network nodes are sources of independent traffic streams for each of these commodities (as in a "uniform all-to-all" situation).

However, the assumption can be quite restrictive for general multi-hop networks, as it does not include the simple case of a single node sending data to a single destination over multiple relay nodes (without having each of these relays also sending data to the same destination). This is a new assumption and is not required in the cross layer control algorithms of [2] [3] [34] that achieve a $[O(1/V), O(V)]$ utility-delay tradeoff. However, it is essential in our analysis for demonstrating "super-fast" $[O(1/V), O(\log(V))]$ tradeoffs.[5] We conjecture that in particular special cases (such as non-stochastic networks), it is also possible to achieve logarithmic tradeoffs without this new assumption, although the general stochastic problem seems quite difficult and perhaps un-achievable in some cases.

[5]The original conference version of this paper [1] used this assumption implicitly in the proofs (where it was assumed that the set of all active sessions was the same as the set of all valid network queues), but did not state this assumption explicitly. We would like to thank the reviewers for their helpful comments that led us to clarify this issue.

### B. The Multi-Hop Lyapunov Function

For given parameters $Q > 0$, $\omega > 0$ (to be determined later), we define the following *bi-modal Lyapunov function*:

$$L(\underline{U}) \triangleq \sum_{(n,c)\in\mathcal{K}} \left[ e^{\omega(Q-U_n^{(c)})} + e^{\omega(U_n^{(c)}-Q)} - 2 \right]$$

This Lyapunov function achieves its minimum value of $L(\underline{U}) = 0$ when $U_n^{(c)} = Q$ for all $(n,c) \in \mathcal{K}$, and increases exponentially when the backlog in any queue deviates from the $Q$ threshold either to the right or to the left. Let $\underline{X} \triangleq [\underline{U}, \underline{Z}]$ be the combined system state, and define the Lyapunov function $\Psi(\underline{X}) \triangleq L(\underline{U}) + \sum_{(n,c)\in\mathcal{K}} (Z_n^{(c)})^2$. Using (23) and (24) the Lyapunov drift can be calculated in a manner similar to that of the one-hop network. Specifically, define $\delta_n^{(c)}(t)$ as follows:

$$\delta_n^{(c)}(t) \triangleq \sum_b \mu_{nb}^{(c)}(t) - \sum_a \mu_{an}^{(c)}(t) - R_n^{(c)}(t)$$

For $\epsilon, \omega$ parameters specified in Section VI-C, the drift satisfies (see [1] for details):

$$\begin{aligned}
&\Delta(\underline{X}(t)) - V\mathbb{E}\left\{ \sum_{(n,c)\in\mathcal{K}} g_n^{(c)}(\gamma_n^{(c)}(t)) \mid \underline{X}(t) \right\} \le C \\
&-\omega \sum_{(n,c)\in\mathcal{K}} 1_n^{(c)R}(t) e^{\omega(U_n^{(c)}(t)-Q)} \left[ \mathbb{E}\left\{ \delta_n^{(c)}(t) \mid \underline{X}(t) \right\} - \frac{\epsilon}{2} \right] \\
&-\omega \sum_{(n,c)\in\mathcal{K}} 1_n^{(c)L}(t) e^{\omega(Q-U_n^{(c)}(t))} \left[ \mathbb{E}\left\{ -\delta_n^{(c)}(t) \mid \underline{X}(t) \right\} - \frac{\epsilon}{2} \right] \\
&-2 \sum_{(n,c)\in\mathcal{K}} Z_n^{(c)}(t) \mathbb{E}\left\{ \sum_b \mu_{nb}^{(c)}(t) - \sum_a \mu_{an}^{(c)}(t) \mid \underline{X}(t) \right\} \\
&+2 \sum_{(n,c)\in\mathcal{K}} Z_n^{(c)}(t) \mathbb{E}\left\{ \gamma_n^{(c)}(t) \mid \underline{X}(t) \right\} \\
&-V\mathbb{E}\left\{ \sum_{(n,c)\in\mathcal{K}} g_n^{(c)}(\gamma_n^{(c)}(t)) \mid \underline{X}(t) \right\}
\end{aligned}$$

for some positive constant $C$, and where indicator functions $1_n^{(c)L}(t)$ and $1_n^{(c)R}(t)$ are defined for all $(n,c) \in \mathcal{K}$ as follows:

$$1_n^{(c)L}(t) \triangleq \begin{cases} 1 & \text{if } U_n^{(c)}(t) < Q \\ 0 & \text{if } U_n^{(c)}(t) \ge Q \end{cases}$$

and $1_n^{(c)R}(t) = 1 - 1_n^{(c)L}(t)$. That is, if $(n,c) \in \mathcal{K}$, then $1_n^{(c)L}(t)$ and $1_n^{(c)R}(t)$ take the value 1 if and only if the corresponding queue backlog is to the "Left" and "Right" of the $Q$ threshold, respectively. For convenience, for all $(n,c) \notin \mathcal{K}$ we define $1_n^{(c)R}(t) = 1_n^{(c)L}(t) = Z_n^{(c)}(t) = 0$ for all $t$, and $g_n^{(c)}(r) = 0$ for all $r$. This allows the summations "$\sum_{(n,c)\in\mathcal{K}}$" in the above drift bound to be replaced with "$\sum_{n=1}^{N}\sum_{c=1}^{N}$." Designing a control policy to minimize the right hand side of this drift expression every timeslot leads to the following *Multi-Hop UDOA Algorithm*:

*Multi-Hop UDOA Algorithm:* The control policy is decoupled into the following policies for flow control, routing, and resource allocation, implemented every timeslot:

- *Flow Control:* The flow controller at each node $n$ observes $U_n^{(c)}(t)$ and $Z_n^{(c)}(t)$ and makes the following decisions for each commodity $c$ such that $(n,c) \in \mathcal{K}$:
  1) If $U_n^{(c)}(t) \ge Q$, then choose $R_n^{(c)}(t) = 0$. Next, list all remaining $(n,c)$ streams at node $n$ in order of increasing $U_n^{(c)}(t)$, and sequentially assign $R_n^{(c)}(t)$ to be as large as possible for the commodities $c$ with the smallest values of $U_n^{(c)}(t)$ (noting that $R_n^{(c)}(t) \le A_n^{(c)}(t) + L_n^{(c)}(t)$), subject to the constraint $\sum_c R_n^{(c)}(t) \le R_{max}$.
  2) Choose $\gamma_n^{(c)}(t) = \gamma_n^{(c)}$, where $\gamma_n^{(c)}$ solves:

$$\begin{aligned} \text{Maximize:} \quad & V g_n^{(c)}(\gamma_n^{(c)}) - 2Z_n^{(c)}(t)\gamma_n^{(c)} \\ \text{Subject to:} \quad & 0 \le \gamma_n^{(c)} \le R_{max} \end{aligned}$$

  The virtual queues $Z_n^{(c)}(t)$ are then updated according to (24), using the $\gamma_n^{(c)}(t)$ values computed above and the $\mu_{an}^{(c)}(t)$, $\mu_{nb}^{(c)}(t)$ values computed by the following routing and resource allocation algorithms.
- *Routing:* The controller at each node $n$ observes the queue backlogs of its neighboring nodes, and for each neighbor node $b$ and each commodity $c$ such that $(n,b) \in \mathcal{L}_c$, it computes $W_{nb}^{(c)}(t)$, defined as follows:

$$\begin{aligned} W_{nb}^{(c)}(t) \triangleq & \left[ 1_n^{(c)R}(t) e^{\omega(U_n^{(c)}(t)-Q)} - 1_b^{(c)R}(t) e^{\omega(U_b^{(c)}(t)-Q)} \right] \\ & - \left[ 1_n^{(c)L}(t) e^{\omega(Q-U_n^{(c)}(t))} - 1_b^{(c)L}(t) e^{\omega(Q-U_b^{(c)}(t))} \right] \\ & + \frac{2}{\omega} \left[ Z_n^{(c)}(t) - Z_b^{(c)}(t) \right] \end{aligned}$$

  (recall that $1_b^{(b)R}(t) = 1_b^{(b)L}(t) = 0$, $Z_b^{(b)}(t) = 0$ for all $t$ and $b \in \{1, \ldots, N\}$). The *optimal commodity* $c_{nb}^*(t)$ is then chosen as the commodity $c^*$ that maximizes $W_{nb}^{(c)}(t)$ over all $c \in \{1, \ldots, N\}$ such that $(n,b) \in \mathcal{L}_c$ (ties are broken arbitrarily). Define $W_{nb}^*(t) \triangleq \max[W_{nb}^{(c^*)}(t), 0]$. Routing variables are then chosen as follows:

$$\mu_{nb}^{(c)}(t) = \begin{cases} 0 & \text{if } W_{nb}^*(t) = 0 \text{ or } c \ne c_{nb}^*(t) \\ \mu_{nb}(t) & \text{otherwise} \end{cases}$$

  where the transmission rates $\mu_{nb}(t)$ are computed by the resource allocation algorithm below.
- *Resource Allocation:* Every timeslot $t$ the network controllers observe the current channel state matrix $\underline{S}(t)$ and allocate transmission rates $(\mu_{ab}(t)) = (\mu_{ab})$, where $(\mu_{ab})$ solves the following optimization problem:

$$\begin{aligned} \text{Maximize:} \quad & \textstyle\sum_{ab} W_{ab}^*(t)\mu_{ab} \\ \text{Subject to:} \quad & (\mu_{ab}) \in \Omega_{\underline{S}(t)} \end{aligned}$$

  where the $W_{ab}^*(t)$ weights are obtained from the above routing algorithm.

The flow control algorithm can be implemented separately at each node using only queue length information and utility functions associated with that node. The routing algorithm can also be implemented in a distributed manner provided that nodes are aware of the backlog levels of their neighbors. It is interesting to note that if a given node has backlog that is below the $Q$ threshold, the weights are impacted negatively which tends to reduce the amount of data transmitted from this

node and increase the amount of data transmitted *to* this node. The opposite occurs in the case when backlog is above the $Q$ threshold. The resource allocation policy is the most complex part of the Multi-Hop UDOA algorithm, and is identical to that of One-Hop UDOA.

### C. Algorithm Performance

Assume that exogenous arrivals to any node are bounded by a value $A_{max}$ on every timeslot, so that: $\sum_c A_n^{(c)}(t) \leq A_{max}$ for all $n \in \{1, \ldots, N\}$. Let $R_{max}$ be any value greater than or equal to $A_{max}$. We further make the *fully active* assumption discussed in Section VI-A. Specifically, we assume that there is an optimally fair operating point $(r_n^{*(c)})$ with $r_n^{*(c)} > 0$ for all $(n,c) \in \mathcal{K}$, and that the input rate matrix $\underline{\lambda}$ strictly dominates this operating point. Define $\epsilon_{max}$ as the largest value of $\epsilon$ such that $\epsilon \leq r_n^{*(c)} \leq \lambda_n^{(c)} - \epsilon$ for all sessions $(n,c) \in \mathcal{K}$ (see Fig. 1). It is also useful to define $\delta_{max}$ as the largest possible change in the total queue backlog at any node during a timeslot:

$$\delta_{max} \triangleq \max[R_{max} + \mu_{max}^{in}, \mu_{max}^{out}]$$

Define time averages $\overline{U}_n^{(c)}(t)$ and $\overline{r}_n^{(c)}(t)$ as follows:

$$\overline{U}_n^{(c)}(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{U_n^{(c)}(\tau)\right\} \ , \ \overline{r}_n^{(c)}(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{R_n^{(c)}(\tau)\right\}$$

*Theorem 4:* (Multi-Hop UDOA Performance) Fix parameters $V$, $\epsilon$ such that $V > 0$ and $0 < \epsilon \leq \epsilon_{max}$, and choose any positive value $\omega$ that satisfies:

$$\omega \delta_{max} e^{\omega \delta_{max}} \leq \epsilon / \delta_{max}$$

Further define:

$$Q \triangleq \frac{2}{\omega}\log(V)$$

Then the Multi-Hop UDOA algorithm stabilizes all actual and virtual queues of the system, and yields:

$$\begin{aligned} \limsup_{t\to\infty} \frac{1}{|\mathcal{K}|}\sum_{(n,c)\in\mathcal{K}} \overline{U}_n^{(c)}(t) &\leq O\left(\log(V)\right) \\ \liminf_{t\to\infty} \sum_{(n,c)\in\mathcal{K}} g_n^{(c)}\left(\overline{r}_n^{(c)}(t)\right) &\geq g^* - O(1/V) \end{aligned}$$

*Proof:* The proof is similar to that of the One-Hop UDOA algorithm (see [1] for details). □

Simulation results illustrating the $[O(1/V), O(\log(V))]$ performance of the UDOA algorithm for a simple two-queue downlink are presented in [1] (omitted here for brevity), where simple modifications that yield (constant factor) improvements are also discussed. It was noted that average queue congestion is quite close to the $Q$ threshold. Further, the magnitude of $Q$ was chosen to ensure a sufficiently small edge probability. However, our analysis was conservative and various simulations revealed that queues never re-entered the edge region after leaving it. Constant factor delay improvements were thus observed by appropriately decreasing the value of $Q$, without significantly effecting the edge probability or network utility.

## VII. Conclusions

This work presents a theory of utility and delay tradeoffs for general data networks with stochastic channel conditions and randomly arriving traffic. A novel control technique was developed and shown to push total system utility arbitrarily close to the optimal operating point, with a corresponding logarithmic tradeoff in average end-to-end network delay. The algorithm is decoupled into separate strategies for flow control, routing, and resource allocation, and can be implemented without requiring knowledge of traffic rates or channel statistics. Further, we proved that no other algorithm can improve upon the logarithmic tradeoff curve. This work establishes a new and important relationship between utility and delay, and introduces fundamentally new techniques for performance optimal scheduling. These techniques can likely be applied in other areas of stochastic optimization and control to yield simple solutions that offer significant performance improvements.

## Appendix A — Performance Analysis

Here we prove the performance bounds expressed in Theorem 2 for the One-Hop UDOA control algorithm. We first review a central result from [2] [3] [4] [34] concerning performance optimal Lyapunov analysis, presented here in a modified form. Consider any queueing network with queue backlogs expressed as a vector $\vec{X}(t) = (X_1(t), \ldots, X_M(t))$. Let $\vec{R}(t) = (R_1(t), \ldots, R_M(t))$ represent an associated control process confined to some compact control space. Let $g(\vec{r})$ be any non-negative, concave utility function, and let $g(\vec{r}^*)$ represent a target utility value for the time average of the $\vec{R}(t)$ process. Define $g_{max}$ as the maximum of $g(\vec{R}(t))$ over the control space. Let $\Psi(\vec{X})$ be a non-negative function of the queue backlog vector. We call $\Psi(\vec{X})$ a *Lyapunov function*, and define the *conditional Lyapunov drift* $\Delta(\vec{X}(t))$ as follows:

$$\Delta(\vec{X}(t)) \triangleq \mathbb{E}\left\{\Psi(\vec{X}(t+1)) - \Psi(\vec{X}(t)) \mid \vec{X}(t)\right\}$$

*Lemma 2:* (Lyapunov Drift with Performance Optimization) If there exist positive values $B, \epsilon, V$ and a non-negative function $f(\vec{X})$ such that every timeslot $t$ and for all possible $\vec{X}(t)$, the conditional Lyapunov drift satisfies:

$$\Delta(\vec{X}(t)) - V\mathbb{E}\left\{g(\vec{R}(t)) \mid \vec{X}(t)\right\} \leq B - \epsilon f(\vec{X}(t)) - V g(\vec{r}^*)$$

then we have:

$$\begin{aligned} \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{f(\vec{X}(\tau))\right\} &\leq \frac{B + V g_{max}}{\epsilon} \\ \liminf_{t\to\infty} g(\overline{r}_1(t), \ldots, \overline{r}_M(t)) &\geq g(\vec{r}^*) - B/V \end{aligned}$$

where

$$\overline{r}_m(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{R_m(\tau)\right\} \text{ for } m \in \{1, \ldots, M\}$$

*Proof:* The proof is almost identical to the proofs of similar statements in [2] [3] [4] [34], and is omitted for brevity. □

If $V$ is a free control variable, then the lemma shows that total utility can be pushed arbitrarily close to the target utility value, with a corresponding linear increase in the time average

of $f(\vec{X}(t))$. Below we construct a Lyapunov function and show that the UDOA control law yields a drift similar to the form given in the lemma above.

### A. Computing the Drift

Define $\vec{X}(t) \triangleq [\vec{U}(t); \vec{Z}(t)]$ as the collective state of the actual and virtual queues of the system. Consider the Lyapunov function $L(\vec{U})$ defined in (13), and let $\Delta_L(\vec{X}(t))$ represent the conditional drift. Note that this is a function of the full system state because control decisions that effect the drift can, in general, depend on both $\vec{U}(t)$ and $\vec{Z}(t)$. To compute $\Delta_L(\vec{X}(t))$, for notational convenience we define the following set of variables $\delta_m(t)$:

$$\delta_m(t) \triangleq \mu_m(t) - R_m(t) \tag{25}$$

Note by definition that $|\delta_m(t)| \leq \delta_{max}$.

*Lemma 3:* For any $\omega > 0$, we have:

$$\sum_{m=1}^{M} e^{\omega(Q-U_m(t+1))} \leq M + \sum_{m=1}^{M} e^{\omega(Q-U_m(t))} \left[1 + \omega\delta_m(t) + \frac{\omega^2\delta_{max}^2}{2} e^{\omega\delta_{max}}\right]$$

*Proof:* From the finite buffer queueing law (12) we have:

$$U_m(t+1) \geq \min[U_m(t) - \delta_m(t), Q]$$

We thus have:

$$\begin{aligned} e^{-\omega U_m(t+1)} &\leq e^{-\omega Q} + e^{-\omega U_m(t)} e^{\omega\delta_m(t)} \\ &\leq e^{-\omega Q} + \\ &\quad e^{-\omega U_m(t)} \left[1 + \omega\delta_m(t) + \frac{\omega^2\delta_{max}^2}{2} e^{\omega\delta_{max}}\right] \end{aligned}$$

where the final inequality follows by the Taylor expansion of $e^{\omega\delta}$, noting that $|\delta_m(t)| \leq \delta_{max}$. Summing the above inequality over all links $m$ and multiplying by $e^{\omega Q}$ proves the lemma. □

*Lemma 4:* If $\omega$ satisfies (16), then:

$$\Delta_L(\vec{X}(t)) \leq M - \omega \sum_m e^{\omega(Q-U_m(t))} \left[\mathbb{E}\left\{-\delta_m(t) \mid \vec{X}(t)\right\} - \epsilon\right]$$

*Proof:* Note that if $\omega$ satisfies (16), then:

$$\frac{\omega^2\delta_{max}^2}{2} e^{\omega\delta_{max}} \leq \omega\epsilon$$

Substituting this inequality into the expressions of Lemma 3 and shifting terms yields the result. □

Lemma 4 establishes the drift of the Lyapunov function $L(\vec{U}(t))$ associated with the actual queues of the system. Now let $\vec{Z}(t) = (Z_1(t), \ldots, Z_M(t))$ represent the vector of virtual queue processes, and define an additional Lyapunov function $H(\vec{Z}(t))$ as follows:

$$H(\vec{Z}(t)) \triangleq \sum_{m=1}^{M} (Z_m(t))^2$$

Define $\Delta_H(\vec{X}(t))$ as the conditional drift of the Lyapunov function $H(\vec{Z}(t))$. Using the update equation (14) for the $Z_m(t)$ queues together with a standard computation for quadratic Lyapunov drift, we have (see, for example, [34]):

$$\Delta_H(\vec{X}(t)) \leq M(\mu_{max}^2 + R_{max}^2) - 2\sum_m Z_m(t)\mathbb{E}\left\{\mu_m(t) - \gamma_m(t) \mid \vec{X}(t)\right\} \tag{26}$$

### B. Controlling the Drift

Define an aggregate Lyapunov function $\Psi(\vec{X}(t)) = L(\vec{U}(t)) + H(\vec{Z}(t))$, and let $\Delta(\vec{X}(t))$ represent the conditional drift. Thus:

$$\begin{aligned} \Delta(\vec{X}(t)) - V\mathbb{E}\left\{\textstyle\sum_m g_m(\gamma_m(t)) \mid \vec{X}(t)\right\} &= \Delta_L(\vec{X}(t)) \\ &\quad + \Delta_H(\vec{X}(t)) - V\mathbb{E}\left\{\textstyle\sum_m g_m(\gamma_m(t)) \mid \vec{X}(t)\right\} \end{aligned}$$

where we have subtracted the same term from both sides of the above equality. Using the bounds on $\Delta_L(\vec{X}(t))$ and $\Delta_H(\vec{X}(t))$ given in Lemma 4 and in (26), we have:

$$\begin{aligned} \Delta(\vec{X}(t)) - V\mathbb{E}\left\{\sum_m g_m(\gamma_m(t)) \mid \vec{X}(t)\right\} &\leq MB \\ -\omega\sum_m e^{\omega(Q-U_m(t))}\left[\mathbb{E}\left\{-\delta_m(t) \mid \vec{X}(t)\right\} - \epsilon\right] & \\ -2\sum_m Z_m(t)\mathbb{E}\left\{\mu_m(t) - \gamma_m(t) \mid \vec{X}(t)\right\} & \\ -V\mathbb{E}\left\{\sum_m g_m(\gamma_m(t)) \mid \vec{X}(t)\right\} & \end{aligned} \tag{27}$$

where $B \triangleq 1 + \mu_{max}^2 + R_{max}^2$.

The UDOA algorithm is derived directly from the drift bound (27). Indeed, the flow control and resource allocation algorithms of UDOA *were constructed specifically to minimize the right hand side of (27) over all possible choices of the control variables* $\gamma_m(t)$, $\mu_m(t)$, *and* $R_m(t)$. We show this in more detail below by isolating the control variables corresponding to each layer of the algorithm.

Flow Control: Isolating the $\gamma_m(t)$ variables that occur on the right hand side of (27), we have:

$$\mathbb{E}\left\{2Z_m(t)\gamma_m(t) - Vg_m(\gamma_m(t)) \mid \vec{X}(t)\right\}$$

The UDOA flow control algorithm minimizes these terms for each link $m$ by choosing $\gamma_m(t)$ to maximize $Vg_m(\gamma_m) - 2Z_m(t)\gamma_m$ over $0 \leq \gamma_m \leq R_{max}$.

Likewise, isolating the $R_m(t)$ variables in (27) using the definition $\delta_m(t) \triangleq \mu_m(t) - R_m(t)$ from (25), we have:

$$\mathbb{E}\left\{R_m(t) \mid \vec{X}(t)\right\} \omega\left[-e^{\omega(Q-U_m(t))}\right]$$

These terms are negative and hence are minimized by the UDOA algorithm that selects the largest $R_m(t)$ that satisfies the constraint $R_m(t) \leq \min[A_m(t) + L_m(t), R_{max}]$. In particular, this flow control policy yields a lower drift expression than any randomized flow control policy that chooses $R_m(t)$ as a random fraction of the current arrivals $A_m(t)$.

Resource Allocation: Isolating the $\mu_m(t)$ variables in the right hand side of (27) (using the definition of $\delta_m(t)$) we have:

$$\sum_m \mathbb{E}\left\{\mu_m(t) \,\middle|\, \vec{X}(t)\right\}\left[-2Z_m(t) + \omega e^{\omega(Q-U_m(t))}\right]$$

which is minimized by the UDOA algorithm over all possible choices of the transmission rate vector subject to the current channel condition: $\vec{\mu}(t) \in \Omega_{\vec{S}(t)}$.

### C. Computing Time Averages

As the UDOA algorithm chooses control variables $\gamma_m(t), \mu_m(t), R_m(t)$ that minimize the right hand side of the drift expression (27), we can establish a simpler bound by plugging in decision variables $\gamma_m^*(t), \mu_m^*(t), R_m^*(t)$ corresponding to alternative control policies.

To this end, consider the simple alternative flow control policy of choosing $R_m^*(t) = A_m(t)$ for all $m$ and all $t$ (so that all new arrivals are immediately admitted), and choosing $\gamma_m^*(t) = 0$, $\mu_m^*(t) = 0$ for all $m$ and all $t$. It follows that $\mathbb{E}\left\{-\delta_m^*(t) \mid \vec{X}(t)\right\} = \mathbb{E}\left\{A_m(t) - 0\right\} = \lambda_m$ for all $m$. Thus, plugging these alternative control decisions into the right hand side of the drift bound (27) preserves the inequality and yields:

$$\Delta(\vec{X}(t)) - V\mathbb{E}\left\{\sum_m g_m(\gamma_m(t)) \mid \vec{X}(t)\right\} \leq MB - \omega\sum_m e^{\omega(Q-U_m(t))}\left[\lambda_m - \epsilon\right] \quad (28)$$

The above inequality is in the exact form for application of Lemma 2 (Lyapunov Optimization), and hence:

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_m(\lambda_m-\epsilon)\mathbb{E}\left\{e^{\omega(Q-U_m(t))}\right\} \leq \frac{MB + Vg_{max}}{\omega} \quad (29)$$

### D. Deriving the Edge Probability Bound

Define $\alpha_m^E(t) \triangleq Pr[U_m(t) < \mu_{max}]$, and note that:

$$\mathbb{E}\left\{e^{\omega(Q-U_m(t))}\right\} \geq \alpha_m^E(t)e^{\omega(Q-\mu_{max})}$$

It follows that $\overline{e^{\omega(Q-U_m)}} \geq \overline{\alpha_m^E}e^{\omega(Q-\mu_{max})}$, where:

$$\overline{e^{\omega(Q-U_m)}} \triangleq \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\left\{e^{\omega(Q-U_m(t))}\right\}$$

$$\overline{\alpha_m^E} \triangleq \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\alpha_m^E(t)$$

Using this inequality together with (29) yields:

$$\begin{aligned}\overline{\alpha_m^E} &\leq \left[\frac{MB+Vg_{max}}{\omega(\lambda_m-\epsilon)}\right]e^{\omega\mu_{max}}e^{-\omega Q} \\ &\leq \left[\frac{MB+Vg_{max}}{\omega(\lambda_m/2)}\right]e^{\omega\mu_{max}}\frac{1}{V^2} \\ &= O(1/V)\end{aligned} \quad (30)$$

where we have used the fact that $Q = \frac{2}{\omega}\log(V)$ and that $\epsilon \leq \epsilon_{max} \leq \lambda_m/2$.

### E. Deriving the Utility Bound

We now consider an alternative choice of control variables $\gamma_m^*(t), \mu_m^*(t), R_m^*(t)$ to derive the utility bound. Let $\vec{r}^*$ represent the optimal solution of (4). Because $\vec{r}^* \in \Lambda$, we know by Theorem 1 that there must exist a stationary randomized resource allocation policy that chooses variables $\mu_m^*(t)$ independently of queue backlog, and yields:

$$\mathbb{E}\left\{\mu_m^*(t) \mid \vec{X}(t)\right\} = r_m^* \quad \text{for all } m \in \{1,\ldots,M\} \quad (31)$$

Now consider the flow control policy that chooses:

$$\gamma_m^*(t) = r_m^* \text{ for all } t \text{ and all } m \quad (32)$$

Further, consider the flow control algorithm that makes independent probabilistic admission decisions $R_m^*(t)$ every timeslot and for each $m$ as follows:

$$R_m^*(t) = \begin{cases} A_m(t) & \text{with probability } p_m \\ 0 & \text{otherwise}\end{cases}$$

where $p_m = (r_m^* + \epsilon)/\lambda_m$. These are valid probabilities because of the assumption that $r_m^* + \epsilon \leq \lambda_m$ for all $m$. Hence, we have:

$$\mathbb{E}\left\{R_m^*(t) \mid \vec{X}(t)\right\} = r_m^* + \epsilon \quad (33)$$

Using (33) and (31) in the definition of $\delta_m(t)$ (given in (25)), we have: $\mathbb{E}\left\{-\delta_m^*(t) \mid \vec{X}(t)\right\} = r_m^* + \epsilon - r_m^* = \epsilon$. Plugging these alternative decisions (including (32) and (31)) into the right hand side of the drift expression (27) thus yields:

$$\Delta(\vec{X}(t)) - V\mathbb{E}\left\{\sum_m g_m(\gamma_m(t)) \Big| \vec{X}(t)\right\} \leq MB - V\sum_m g_m(r_m^*)$$

Using Lemma 2 on the above drift expression yields:

$$\liminf_{t\to\infty}\sum_m g_m(\overline{\gamma}_m(t)) \geq \sum_m g_m(r_m^*) - \frac{MB}{V} \quad (34)$$

where $\overline{\gamma}_m(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\left\{\gamma_m(\tau)\right\}$.

We now use the following lemmas relating $\overline{\gamma}_m(t)$ and $\overline{r}_m(t)$ (where $\overline{r}_m(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\left\{R_m^{admit}(\tau)\right\}$).

*Lemma 5:* All virtual queues are stable and satisfy:

$$Z_m(t) \leq Vg_m'(0)/2 + R_{max} \text{ for all } t$$

*Lemma 6:* Under UDOA, we have:

$$\limsup_{t\to\infty}\max[\overline{\gamma}_m(t) - \overline{r}_m(t), 0] \leq \mu_{max}\overline{\alpha_m^E}$$

*Proof:* (Lemma 5) To show that the virtual queues are bounded, note that $\gamma_m(t)$ is chosen to maximize $Vg_m(\gamma) - 2Z_m(t)\gamma$ over $0 \leq \gamma \leq R_{max}$. It follows that $\gamma_m(t) = 0$ whenever $2Z_m(t) > Vg_m'(0)$. It follows from (14) that $Z_m(t)$ cannot further increase if this condition is satisfied, so that $Z_m(t) \leq Vg_m'(0)/2 + R_{max}$, proving Lemma 5. □

*Proof:* (Lemma 6) Define time average edge probability $\overline{\alpha}_m^E(t)$ as follows: $\overline{\alpha}_m^E(t) = \frac{1}{t}\sum_{\tau=0}^{t-1}Pr[U_m(\tau) < \mu_{max}]$. It is not difficult to show (see [1] for details):

$$\overline{r}_m(t) \geq \overline{\mu}_m(t) - \overline{\alpha}_m^E(t)\mu_{max} \quad (35)$$

However, because all virtual queues are stable (Lemma 5), we have that the $\liminf$ expression (15) from Lemma 1 holds. Substituting (35) into (15) yields:

$$\liminf_{t\to\infty}\left[\overline{r}_m(t) + \mu_{max}\overline{\alpha}_m^E(t) - \overline{\gamma}_m(t)\right] \geq 0 \quad (36)$$

The inequality (36) directly implies that:

$$\limsup_{t\to\infty}(\overline{\gamma}_m(t) - \overline{r}_m(t)) \leq \mu_{max}\overline{\alpha_m^E}$$

which implies the conclusion of the lemma. □

Now define $\nu$ to be the largest right derivative of any utility function. We thus have for all $m$:

$$\begin{aligned} g_m(\overline{r}_m(t)) &= g_m\left(\overline{\gamma}_m(t) - [\overline{\gamma}_m(t) - \overline{r}_m(t)]\right) \\ &\geq g_m(\overline{\gamma}_m(t)) - \nu \max[0, \overline{\gamma}_m(t) - \overline{r}_m(t)] \end{aligned}$$

We thus have:

$$\begin{aligned} \sum_m g_m(\overline{r}_m(t)) &\geq \sum_m g_m(\overline{\gamma}_m(t)) \\ &\quad -\nu \sum_m \max[0, \overline{\gamma}_m(t) - \overline{r}_m(t)] \end{aligned}$$

Taking the $\liminf$ and using Lemma 6 and (30) yields:

$$\begin{aligned} \liminf_{t\to\infty} \sum_m g_m(\overline{r}_m(t)) &\geq \liminf_{t\to\infty} \sum_m g_m(\overline{\gamma}_m(t)) - O(1/V) \\ &\geq \sum_m g_m(r^*_m) - O(1/V) \end{aligned}$$

where the final inequality follows by (34). This proves the utility bound and completes the proof of Theorem 2.

## REFERENCES


[1] M. J. Neely. Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks. *Proceedings of IEEE INFOCOM*, April 2006.

[2] M. J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. PhD thesis, Massachusetts Institute of Technology, LIDS, 2003.

[3] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *Proceedings of IEEE INFOCOM*, March 2005.

[4] M. J. Neely. Energy optimal control for time varying wireless networks. *Proceedings of IEEE INFOCOM*, March 2005.

[5] R. Berry and R. Gallager. Communication over fading channels with delay constraints. *IEEE Transactions on Information Theory*, vol. 48, no. 5, pp. 1135-1149, May 2002.

[6] M. J. Neely and E. Modiano. Improving delay in ad-hoc mobile networks via redundant packet transfers. *Proc. of the Conference on Information Sciences and Systems*, Johns Hopkins University: March 2003.

[7] M. J. Neely and E. Modiano. Capacity and delay tradeoffs for ad-hoc mobile networks. *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1917-1937, June 2005.

[8] A. El Gammal, J. Mammen, B. Prabhakar, and D. Shah. Throughput-delay trade-off in wireless networks. *IEEE Proc. of INFOCOM*, 2004.

[9] S. Toumpis and A. J. Goldsmith. Large wireless networks under fading, mobility, and delay constraints. *IEEE Proceddings of INFOCOM*, 2004.

[10] X. Lin and N. B. Shroff. The fundamental capacity-delay tradeoff in large mobile ad hoc networks. *Purdue University Tech. Report*, 2004.

[11] F.P. Kelly, A.Maulloo, and D. Tan. Rate control for communication networks: Shadow prices, proportional fairness, and stability. *Journ. of the Operational Res. Society*, 49, p.237-252, 1998.

[12] S. H. Low. A duality model of tcp and queue management algorithms. *IEEE Trans. on Networking*, Vol. 11(4), August 2003.

[13] L. Xiao, M. Johansson, and S. Boyd. Simultaneous routing and resource allocation for wireless networks. *Proc. of the 39th Annual Allerton Conf. on Comm., Control, Comput.*, Oct. 2001.

[14] D. Julian, M. Chiang, D. O'Neill, and S. Boyd. Qos and fairness constrained convex optimization of resource allocation for wireless cellular and ad hoc networks. *Proc. INFOCOM*, 2002.

[15] P. Marbach. Priority service and max-min fairness. *IEEE Proceedings of INFOCOM*, 2002.

[16] P. Marbach and R. Berry. Downlink resource allocation and pricing for wireless networks. *IEEE Proc. of INFOCOM*, 2002.

[17] R. Berry, P. Liu, and M. Honig. Design and analysis of downlink utility-based schedulers. *Proceedings of the 40th Allerton Conference on Communication, Control, and Computing*, Oct. 2002.

[18] B. Krishnamachari and F. Ordonez. Analysis of energy-efficient, fair routing in wireless sensor networks through non-linear optimization. *IEEE Vehicular Technology Conference*, Oct. 2003.

[19] J. W. Lee, R. R. Mazumdar, and N. B. Shroff. Downlink power allocation for multi-class cdma wireless networks. *IEEE Proceedings of INFOCOM*, 2002.

[20] R. Cruz and A. Santhanam. Optimal routing, link scheduling, and power control in multi-hop wireless networks. *IEEE Proceedings of INFOCOM*, April 2003.

[21] M. Chiang. Balancing transport and physical layer in wireless multihop networks: Jointly optimal congestion control and power control. *IEEE J. Sel. Area Comm.*, Jan. 2005.

[22] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transacations on Automatic Control*, Vol. 37, no. 12, Dec. 1992.

[23] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. on Inform. Theory*, vol. 39, pp. 466-478, March 1993.

[24] P.R. Kumar and S.P. Meyn. Stability of queueing networks and scheduling policies. *IEEE Trans. on Automatic Control*, Feb. 1995.

[25] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, and P. Whiting. Providing quality of service over a shared wireless link. *IEEE Communications Magazine*, 2001.

[26] M. J. Neely, E. Modiano, and C. E. Rohrs. Power allocation and routing in multi-beam satellites with time varying channels. *IEEE Transactions on Networking*, Feb. 2003.

[27] 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*, January 2005.

[28] E. M. Yeh and A. S. Cohen. Throughput optimal power and rate control for multiaccess and broadcast communications. *Proc. of the International Symposium on Information Theory*, June 2004.

[29] X. Liu, E. K. P. Chong, and N. B. Shroff. A framework for opportunistic scheduling in wireless networks. *Computer Networks*, vol. 41, no. 4, pp. 451-474, March 2003.

[30] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE Proceedings of INFOCOM*, March 2005.

[31] A. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, vol. 50, pp. 401-457, 2005.

[32] J. W. Lee, R. R. Mazumdar, and N. B. Shroff. Opportunistic power scheduling for dynamic multiserver wireless systems. *IEEE Transactions on Wireless Communications*, vol. 5, no.6, pp. 1506-1515, June 2006.

[33] M. J. Neely. Optimal energy and delay tradeoffs for multi-user wireless downlinks. *Proceedings of IEEE INFOCOM*, April 2006.

[34] 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. 1-149, 2006.

[35] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 1997.

[36] A. Tang, J. Wang, and S. Low. Is fair allocation always inefficient. *IEEE Proceedings of INFOCOM*, March 2004.

[37] M. J. Neely. Energy optimal control for time varying wireless networks. *IEEE Transactions on Information Theory*, July 2006.

[38] X. Lin and N. B. Shroff. The impact of imperfect scheduling on cross-layer rate control in wireless networks. *IEEE Proceedings of INFOCOM*, 2005.

[39] P. Chaporkar, K. Kar, and S. Sarkar. Throughput guarantees through maximal scheduling in wireless networks. *Proceedings of 43rd Annual Allerton Conference on Communication Control and Computing*, September 2005.

[40] L. Bui, E. Eryilmaz, R. Srikant, and X. Wu. Joint asynchronous congestion control and distributed scheduling for multi-hop wireless networks. *IEEE Proc. of INFOCOM*, 2006.


**Michael J. Neely** received B.S. degrees in both Electrical Engineering and Mathematics from the University of Maryland, College Park, in 1997. He was then awarded a 3 year Department of Defense NDSEG Fellowship for graduate study at the Massachusetts Institute of Technology, where he received an M.S. degree in 1999 and a Ph.D. in 2003, both in Electrical Engineering. During the Summer of 2002, he worked in the Distributed Sensor Networks group at Draper Labs in Cambridge. In 2004 he joined the faculty of the Electrical Engineering Department at the University of Southern California, where he is currently an Assistant Professor. His research is in the area of stochastic network optimization for satellite and wireless networks, mobile ad-hoc networks, and queueing systems. Michael is a member of Tau Beta Pi and Phi Beta Kappa.