Fairness and Optimal Stochastic Control for Heterogeneous Networks

Michael J. Neely , Eytan Modiano , Chih-Ping Li

***Abstract*— We consider optimal control for general networks with both wireless and wireline components and time varying channels. A dynamic strategy is developed to support all traffic whenever possible, and to make optimally fair decisions about which data to serve when inputs exceed network capacity. The strategy is decoupled into separate algorithms for flow control, routing, and resource allocation, and allows each user to make decisions independent of the actions of others. The combined strategy is shown to yield data rates that are arbitrarily close to the optimal operating point achieved when all network controllers are coordinated and have perfect knowledge of future events. The cost of approaching this fair operating point is an end-to-end delay increase for data that is served by the network. Analysis is performed at the packet level and considers the full effects of queueing.**

***Index Terms*— Stochastic Optimization, Queueing Analysis, Multi-Hop Wireless, Distributed Computing**

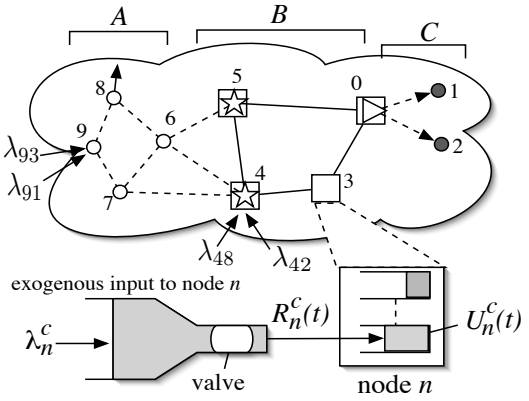

Fig. 1. (a) A heterogeneous network with wireless and wireline data links, and (b) a close-up of one node, illustrating the internal queues and the storage reservoir for exogenous arrivals.

I. INTRODUCTION

Modern data networks consist of a variety of heterogeneous components, and continue to grow as new applications are developed and new technologies are integrated into the existing communication infrastructure. While network resources are expanding, the demand for these resources is also expanding, and it is often the case that data links are loaded with more traffic than they were designed to handle. In order to provide high speed connectivity for future personal computers, hardware devices, wireless units, and sensor systems, it is essential to develop fair networking techniques that take full advantage of all resources and system capabilities. Such techniques must be implemented through simple, localized message passing protocols between neighboring network elements.

In this paper, we design a set of decoupled algorithms for resource allocation, routing, and flow control for general networks with both wireless and wireline data links and time varying channels. Specifically, we treat a network with N nodes and L links. The condition of each link at a given time t is described by a *link state vector* $\vec{S}(t) = (S_1(t), \ldots, S_L(t))$, where $S_l(t)$ is a parameter characterizing the communication channel for link l. For example, if l is a wireless link, $S_l(t)$ may represent the current attenuation factor or noise level. In an unreliable wired link, $S_l(t)$ may take values in the two-element set $\{ON, OFF\}$, indicating whether link l is available for communication. For simplicity of exposition, we consider a slotted system model with slots normalized to integral units $t \in \{0, 1, 2, \ldots\}$. Channels hold their state for the duration of a timeslot, and potentially change states on slot boundaries. We assume there are a finite number of channel state vectors $\vec{S}$. For each $\vec{S}$, let $\Gamma_{\vec{S}}$ denote the set of link transmission rates available for resource allocation decisions when $\vec{S}(t) = \vec{S}$. In particular, every timeslot t the network controllers are constrained to choosing a transmission rate vector $\vec{\mu}(t) = (\mu_1(t), \ldots, \mu_L(t))$ such that $\vec{\mu}(t) \in \Gamma_{\vec{S}(t)}$ (where $\mu_l(t)$ is the transmit rate over link l and has units of bits/slot). Use of this abstract set of transmission rates $\Gamma_{\vec{S}}$ maintains a simple separation between network layer and physical layer concepts, yet is general enough to allow network control to be suited to the unique capabilities of each data link.

As an example, consider the heterogeneous network of Fig. 1 consisting of three separate groups of links A, B, and C: Set A represents a wireless sensor system that connects to a wired infrastructure through two uplink access points, set B represents the wired data links, and set C represents the two downlink channels of a basestation that transmits to two different users 1 and 2. For a given channel state $\vec{S}$, the set of feasible transmission rates $\Gamma_{\vec{S}}$ reduces to a product of rates corresponding to the three independent groups:

$$\Gamma_{\vec{S}} = \Gamma^A_{\vec{S}_A} \times \Gamma^B \times \Gamma^C_{\vec{S}_C}$$

Set $\Gamma^A_{\vec{S}_A}$ might contain a continuum of link rates associated with the channel interference properties and power allocation options of the sensor nodes, and depends only on the link

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

E. Modiano is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (email: modiano@mit.edu). Chih-Ping Li is with the department of Electrical Engineering, University of Southern California, Los Angeles (chihpinl@usc.edu).

states $\vec{S}_A$ of these nodes. Set Γ^B might contain a single vector $(C_1, \ldots, C_k)$ representing the fixed capacities of the k wired links. Set $\Gamma^C_{\vec{S}_C}$ might represent a set of two vectors $\{(\phi_{S_1}, 0), (0, \phi_{S_2})\}$, where ϕ_{S_i} is the rate available over link i if this link is selected to transmit on the given timeslot t when $S_i(t) = S_i$.

Data is transmitted from node to node over potentially multi-hop paths to reach its destination. Let (λ_{nc}) represent the matrix of exogenous arrival rates, where λ_{nc} is the rate of new arrivals to source node n intended for destination node c (in units of bits/slot). The *network layer capacity region* Λ is defined as the closure of the set of all arrival matrices that are stably supportable by the network, considering all possible routing and resource allocation policies (possibly those with perfect knowledge of future events). In [16], a routing and power allocation policy was developed to stabilize a general wireless network whenever the rate matrix (λ_{nc}) is within the capacity region Λ. The purpose of our current paper is to treat heterogeneous networks and develop distributed algorithms for flow control, routing, and resource allocation that provide optimal fairness in cases when arrival rates are *either inside or outside the network capacity region*.

Specifically, we define a set of *utility functions* $g_{nc}(r)$, representing the 'satisfaction' received by sending data from node n to node c at a time average rate of r bits/slot. The goal is to support a fraction of the traffic demand matrix (λ_{nc}) to achieve throughputs (r_{nc}) that maximize the sum of user utilities. We thus have the optimization:

$$\text{Maximize:} \quad \sum_{n,c} g_{nc}(r_{nc}) \tag{1}$$

$$\text{Subject to:} \quad (r_{nc}) \in \Lambda \tag{2}$$

$$0 \leq (r_{nc}) \leq (\lambda_{nc}) \tag{3}$$

where the matrix inequality in (3) is considered entrywise. Inequality (2) is the *stability constraint* and ensures that the admitted rates are stabilizable by the network. Inequality (3) is the *demand constraint* that ensures the rate provided to session (n, c) is no more than the incoming traffic rate of this session.

Let (r^*_{nc}) represent the solution of the above optimization. Assuming that functions $g_{nc}(r)$ are non-decreasing, it is clear that $(r^*_{nc}) = (\lambda_{nc})$ whenever $(\lambda_{nc}) \in \Lambda$. If $(\lambda_{nc}) \notin \Lambda$ there must be at least one value r^*_{nc} that is strictly less than λ_{nc}. The above optimization could in principle be solved if the arrival rates (λ_{nc}) and the capacity region Λ were known in advance, and all users could coordinate by sending data according to the optimal solution. However, the capacity region depends on the channel dynamics, which are unknown to the network controllers and to the individual users. Furthermore, the individual users do not know the data rates or utility functions of other users. In this paper, we develop a practical dynamic control strategy that yields a resulting set of throughputs $(\overline{r}_{nc})$ that are arbitrarily close to the optimal solution of (1)-(3). The distance to the optimal solution is shown to decrease like $1/V$, where V is a control parameter affecting a tradeoff in average delay for data that is served by the network.

Previous work on network fairness and optimization is found in [2]-[12]. In [2], an optimization problem similar to (1)-(2) is considered for a static wireless downlink with infinite backlog, and pricing schemes are developed to enable convergence to a fair power allocation vector. Further static resource allocation problems for wireless systems and sensor networks are treated in [3]-[7], and game theory approaches for wired flow networks are treated in [8]-[11]. These approaches use convex optimization and Lagrangian duality to achieve a fixed resource allocation that is optimal with respect to various utility metrics. In [9] [10], pricing mechanisms are constructed to enable *proportionally fair* routing. Related work in [8] considers *max-min fairness*, and recent applications to the area of internet congestion control are developed in [12].

We note that fixed allocation solutions may not be appropriate in cases when optimal control involves *dynamic resource allocation*. Indeed, in [14] it is shown that energy optimal power allocation in a static ad-hoc network with interference involves the computation of a *periodic transmission schedule*. A similar scheduling problem is shown to be NP-complete in [28]. The capacity of a multi-user wireless downlink with *randomly varying channels* is established in [29], and utility optimization in a similar system is treated in [13]. These formulations do not consider stochastic arrivals and queueing, and solutions require perfect knowledge of channel statistics (approximate policies can be implemented based on long-term measurements).

Stochastic control policies for wireless queueing networks are developed in [15]-[21] based on a theory of *Lyapunov drift*. This theory has been extremely powerful in the development of stabilizing control laws for data networks [15]-[23], but cannot be used to address performance optimization and fairness. Dynamic algorithms for fair scheduling in wireless downlinks are addressed in [24] [25] [26], but do not yield optimal performance for all input rates, as discussed in the next section. A wireless downlink with deterministic ON/OFF channels and arbitrary input rates is developed in [27], and a modified version of the *Serve-the-Longest-ON-Queue* policy is shown to yield maximum throughput. However, the analysis in [27] is closely tied to the channel modeling assumptions, and does not appear to offer solutions for more general networks or fairness criteria.

The main contribution of our work is the development of a novel control policy that yields optimal performance for general stochastic networks and general fairness metrics. The policy does not require knowledge of channel statistics, input rates, or the global network topology. Our analysis uses a new Lyapunov drift technique that enables stability and performance optimization to be achieved simultaneously, and presents a fundamental approach to *stochastic network optimization*.

In the next section, we consider a simple wireless downlink and describe the shortcomings of previously proposed algorithms in terms of fairness. In Section III we develop a fair scheduling algorithm for general networks under the special case when all active input reservoirs are 'infinitely backlogged.' In Section V we construct a modified algorithm that yields optimal performance without the infinite backlog assumption. Example simulations for wireless networks and $N \times N$ packet switches are presented in Section VI.

II. A Downlink Example

Consider a wireless basestation that transmits data to two downlink users 1 and 2 over two different channels (as illustrated by considering only the triangle-node of the network in Fig. 1). Time is slotted and packets for each user arrive to the basestation according to independent Bernoulli processes with rates λ_1 and λ_2. Let $U_1(t)$ and $U_2(t)$ represent the current backlog of packets waiting for transmission to user 1 and user 2, respectively. Channels independently vary between ON and OFF states every slot according to Bernoulli processes, with ON probabilities p_1 and p_2, and we assume that $p_1 < p_2$. Every timeslot, a controller observes the channel states and chooses to transmit over either channel 1 or channel 2. We assume that a single packet can be transmitted if a channel is ON and no packet can be transmitted when a channel is OFF, so that the only decision is which channel to serve when both channels are ON.

The capacity region Λ for this system is described by the set of all rates (λ_1, λ_2) that satisfy:

$$\lambda_1 \leq p_1 \quad , \quad \lambda_2 \leq p_2$$
$$\lambda_1 + \lambda_2 \leq p_1 + (1-p_1)p_2$$

These conditions are necessary for stability because the output rate from any channel i is at most p_i, and the maximum sum rate out of the system is $p_1 + (1-p_1)p_2$. Furthermore, it is shown in [19] that the 'Maximum Weight Match' (MWM) policy of serving the ON queue with the largest backlog achieves stability whenever input rates are strictly interior to the above region.

Now define $g_1(r) = g_2(r) = \log(r)$, and consider the *proportional fairness* control objective of maximizing $\log(r_1) + \log(r_2)$, where r_1 and r_2 are the delivered throughputs over channels 1 and 2 (see [10] for a discussion of proportional fairness). We evaluate three well known algorithms with respect to this fairness metric: The Borst algorithm [24], the 'proportionally fair' Max μ_i/r_i algorithm [25] [26], and the MWM policy [19].

The Borst algorithm chooses the non-empty channel i with the largest $\mu_i(t)/\overline{\mu}_i$ index, where $\mu_i(t)$ is the current channel rate and $\overline{\mu}_i$ is the average of $\mu_i(t)$. This algorithm is shown in [24] to provide optimal fairness for wireless networks with an 'infinite' number of channels, where each incoming packet is destined for a unique user with its own channel. Although the algorithm was not designed for the 2-queue downlink described above, it is closely related to the Max μ_i/r_i policy, and it is illuminating to evaluate its performance in this context. Applied to the 2-queue downlink, the Borst algorithm reduces to serving the non-empty ON queue with the largest value of $1/p_i$. Because $p_1 < p_2$, this algorithm effectively gives packets destined for channel 1 strict priority over channel 2 packets. Thus, the service of queue 1 is independent of the state of channel 2, and conditioning on the event that a packet is served from channel 1 during a particular timeslot does not change the probability that channel 2 is ON. It follows that the rate of serving channel 1 packets while channel 2 is ON is given by $\lambda_1 p_2$ (assuming queue 1 is stable so that all λ_1 traffic is served). Thus, the stability region of the Borst algorithm is given by:

$$\lambda_1 \leq p_1 \quad (4)$$
$$\lambda_2 \leq p_2 - \lambda_1 p_2 \quad (5)$$

which is a strict subset of the capacity region (see Fig. 2).

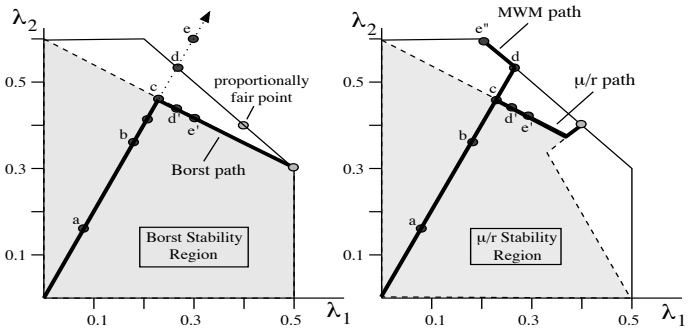

Fig. 2. The downlink capacity region Λ and the stability regions of the Borst policy and the Max μ_i/r_i policy. Input rates (λ_1, λ_2) are pushed toward point $(0.5, 1.0)$, and the simulated throughputs under the Borst, Max μ_i/r_i, and MWM policies are illustrated.

Consider now the related policy of serving the non-empty queue with the largest value of $\mu_i(t)/r_i(t)$, where $r_i(t)$ is the empirical throughput achieved over channel i. This differs from the Borst algorithm in that transmission rates are weighted by the throughput actually delivered rather than the average transmission rate that is offered. This Max μ_i/r_i policy is proposed in [25] [26] and shown to have desirable proportional fairness properties when all queues of the downlink are infinitely backlogged. To evaluate its performance for arbitrary traffic rates (λ_1, λ_2), suppose the running averages $r_1(t)$ and $r_2(t)$ are accumulated over the entire timeline, and suppose the system is stable so that $r_1(t)$ and $r_2(t)$ converge to λ_1 and λ_2. It follows that the algorithm eventually reduces to giving channel 1 packets strict priority if $\lambda_1 < \lambda_2$, and giving channel 2 packets strict priority if $\lambda_2 < \lambda_1$. Thus, if $\lambda_1 < \lambda_2$ then these rates must also satisfy the inequalities (4) and (5), while $\lambda_2 < \lambda_1$ implies the rates must satisfy the inverted inequalities $\lambda_2 \leq p_2$ and $\lambda_1 \leq p_1 - \lambda_2 p_1$. Thus, at first glance it seems that the stability region of this policy is a subset of the stability region of the Borst algorithm. However, its stability region has the peculiar property of including all feasible rate pairs (λ, λ) (see Fig. 2).

In Fig. 2 we consider the special case when $p_1 = 0.5, p_2 = 0.6$, and plot the achieved throughput of the Borst, Max μ_i/r_i, and MWM policies when the rate vector (λ_1, λ_2) is scaled linearly towards the vector $(0.5, 1.0)$, illustrated by the ray in Fig. 2(a). One hundred different rate points on this ray were considered (including example points a - e), and simulations were performed for each point over a period of 5 million timeslots. Fig 2(a) illustrates the resulting throughput of the Borst algorithm, where we have included example points d' and e' corresponding to input rate points d and e. Note that the Borst algorithm always results in throughput that is strictly interior to the capacity region, even when input rates are outside of capacity. Fig. 2(b) illustrates performance of the Max μ_i/r_i and MWM policies. Note that the MWM

policy supports all (λ_1, λ_2) traffic when this rate vector is within the capacity region. However, when traffic is outside of the capacity region the achieved throughput moves along the boundary in the wrong direction, yielding throughputs that are increasingly unfair because it favors service of the higher traffic rate stream. Like the Borst policy, the Max μ_i/r_i policy leads to instability for all (stabilizable) input rates on the ray segment c-d, and yields throughput that is strictly interior to the capacity region even when inputs exceed system capacity (compare points e and e'). However, the throughput eventually touches the capacity region boundary, reaching the proportionally fair point $(0.4, 0.4)$ when input rates are sufficiently far outside of the capacity region.

It is clear from this simple downlink example that there is a need for a 'universally fair' algorithm, one that performs well regardless of whether inputs are inside or outside of the capacity region. For this example, such an algorithm would yield throughput that increases toward the point d of the figure, and then moves on the boundary of the capacity region toward the fair operating point thereafter. In the following, we develop such an algorithm for general multihop networks.

III. Control of Heterogeneous Networks

Consider a heterogeneous network with N nodes, L links, and time varying channels $\vec{S}(t)$, as shown in Fig. 1. Each link $l \in \{1, \ldots, L\}$ represents a directed communication channel for transmission from one node to another, and we define $tran(l)$ and $rec(l)$ as the corresponding transmitting and receiving nodes, respectively. Each node of the network maintains a set of output queues for storing data according to its destination. All data (from any source node) that is destined for a particular node $c \in \{1, \ldots, N\}$ is classified as *commodity c data*, and we let $U_n^{(c)}(t)$ represent the backlog of commodity c data currently stored in node n (see Fig. 1). At the network layer, a control algorithm makes decisions about routing, scheduling, and resource allocation in reaction to current channel state and queue backlog information. The objective is to deliver all data to its proper destination, potentially by routing over multi-hop paths.

As a general algorithm might schedule multiple commodities to flow over the same link on a given timeslot, we define $\mu_l^{(c)}(t)$ as the rate offered to commodity c traffic along link l during timeslot t. [1] The transmission rates and routing variables are chosen by a *dynamic scheduling and routing algorithm*. Specifically, the network makes the following control decisions every slot:

- *Resource (Rate) Allocation:* Choose $\vec{\mu}(t) = (\mu_1(t), \ldots, \mu_L(t))$ such that $\vec{\mu}(t) \in \Gamma_{\vec{S}(t)}$
- *Routing/Scheduling:* For each link l, choose $\mu_l^{(c)}(t)$ to satisfy the link rate constraint:

$$\sum_c \mu_l^{(c)}(t) \leq \mu_l(t)$$

A set of *flow controllers* act at every node to limit the new data admitted into the network. Specifically, new data of commodity c that arrives to source node n is first placed in a *storage reservoir* (n, c). A control valve determines the amount of data $R_{nc}(t)$ released from this reservoir on each timeslot. The $R_{nc}(t)$ process acts as the exogenous arrival process affecting behavior of queue backlog $U_n^{(c)}(t)$. Endogenous arrivals consist of commodity c data transmitted to node n from other network nodes. Define Ω_n as the set of all links l such that $tran(l) = n$, and define Θ_n as the set of all links such that $rec(l) = n$. Every timeslot the backlog $U_n^{(c)}(t)$ changes according to the following queueing law:

$$U_n^{(c)}(t+1) \leq \max\left[U_n^{(c)}(t) - \textstyle\sum_{l\in\Omega_n} \mu_l^{(c)}(t), 0\right] + \textstyle\sum_{l\in\Theta_n} \mu_l^{(c)}(t) + R_{nc}(t) \quad (6)$$

The expression above is an inequality rather than an equality because the endogenous arrivals may be less than $\sum_{l\in\Theta_n} \mu_l^{(c)}(t)$ if nodes have little or no commodity c data to transmit. The above dynamics hold for all node pairs $n \neq c$. Data leaves the network when it reaches its destination, and so we define $U_n^{(n)} \triangleq 0$ for all n.

Define $\overline{r}_{nc} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R_{nc}(\tau)\}$ as the time average admission rate of (n, c) data. The goal is to design a joint strategy for resource allocation, routing, and flow control that yields an admitted throughput matrix $(\overline{r}_{nc})$ that maximizes the utility metric (1) subject to the stability constraint (2) and the demand constraint (3). In order to limit congestion in the network, it is important to restrict flow control decisions so that $\sum_c R_{nc}(t) \leq R_n^{max}$ for all nodes n and slots t, where R_n^{max} is defined as the largest possible transmission rate out of node n (summed over all possible outgoing links of n that can be activated simultaneously). We note that any time average rate matrix (r_{nc}) that is within the capacity region Λ necessarily satisfies $\sum_c r_{nc} \leq R_n^{max}$ for all n. Indeed, rates (r_{nc}) violating this constraint cannot be supported, as they would inevitably overload the source queues.

A. Dynamic Control for Infinite Demand

Here we develop a practical control algorithm that stabilizes the network and ensures that utility is arbitrarily close to optimal, with a corresponding tradeoff in network delay. Recall that functions $g_{nc}(r)$ represent the utility of supporting rate r communication from node n to node c (we define $g_{nc}(r) = 0$ if there is no active session of traffic originating at node n and destined for node c). To highlight the fundamental issues of routing, resource allocation, and flow control, we assume that all active sessions (n, c) have *infinite backlog* in their corresponding reservoirs, so that flow variables $R_{nc}(t)$ can be chosen without first establishing that this much data is available in the reservoir. Flow control is imperative in this infinite backlog scenario, and the resulting problem is simpler as it does not involve the demand constraint (3). A modified algorithm is developed in Section V for the general case of finite demand matrices (λ_{nc}) and finite buffer reservoirs.

The following control strategy is decoupled into separate algorithms for resource allocation, routing, and flow control. The strategy combines a novel flow control technique together with a generalization of the DRPC power allocation strategy of [16].

[1] We find that the capacity achieving solution needs only route a single commodity over any given link during a timeslot.

Cross-Layer Control Algorithm 1 (CLC1):

- *Flow Control* — (algorithm FLOW) The flow controller at each node n observes the current level of queue backlogs $U_n^{(c)}(t)$ for each commodity $c \in \{1, \ldots, N\}$. It then sets $R_{nc}(t) = r_{nc}$, where the r_{nc} values are solutions to the following optimization:

$$\text{Maximize :} \quad \sum_{c=1}^{N} \left[V g_{nc}(r_{nc}) - 2 r_{nc} U_n^{(c)}(t) \right] \qquad (7)$$

$$\text{Subject to:} \quad \sum_{c=1}^{N} r_{nc} \leq R_n^{max}$$

  where $V > 0$ is a chosen constant that effects the performance of the algorithm.

- *Routing and Scheduling* — Each node n observes the backlog in all neighboring nodes j to which it is connected by a link l (where $tran(l) = n, rec(l) = j$). Let $W_l^{(c)} = U_{tran(l)}^{(c)}(t) - U_{rec(l)}^{(c)}(t)$ represent the *differential backlog* of commodity c data. Define $W_l^* \triangleq \max_c\{W_l^{(c)}, 0\}$ as the maximum differential backlog over link l (maxed with 0), and let c_l^* represent the maximizing commodity. Data of commodity c_l^* is selected for (potential) routing over link l whenever $W_l^* > 0$.

- *Resource Allocation* — The current channel state $\vec{S}(t)$ is observed, and a transmission rate vector $\vec{\mu}(t)$ is selected by maximizing $\sum_l W_l^* \mu_l(t)$ subject to the constraint $\vec{\mu}(t) \in \Gamma_{\vec{S}(t)}$. The resulting transmission rate of $\mu_l(t)$ is offered to commodity c_l^* data on link l. If any node does not have enough bits of a particular commodity to send over all outgoing links requesting that commodity, *null bits* are delivered.

The flow control algorithm is decentralized, where the control valves for each node n require knowledge only of the queue backlogs in node n. The routing and scheduling algorithm acts according to a differential backlog strategy similar to the backpressure strategy developed in [15], and is decentralized provided that each node i knows the backlog levels of its neighbors. The resource allocation strategy of maximizing $\sum_l W_l^* \mu_l(t)$ is the most complex part of the algorithm, but can be distributed over the independent portions of the network. Specifically, if the network links are grouped into K independent components, the set constraint for each channel state $\vec{S}$ has the product form:

$$\Gamma_{\vec{S}} = \Gamma^1_{\vec{S}_1} \times \Gamma^2_{\vec{S}_2} \times \ldots \times \Gamma^K_{\vec{S}_K}$$

Define β_k as the set of links contained in component k. It follows that resource allocation is decoupled across network components, where each component k independently chooses transmission rates for its own links to maximize $\sum_{l \in \beta_k} W_l^* \mu_l(t)$ subject to $(\mu_l(t))\,|_{\,l \in \beta_k} \in \Gamma^k_{\vec{S}_k}$. In particular, network components only require knowledge of the channel conditions on their own links.

B. Intuitive Description of the Policy

The flow control policy (7) uses a parameter V that determines the extent to which utility optimization is emphasized. Indeed, if V is large relative to the current backlog in the source queues, then the admitted rates $R_{nc}(t)$ will be large, increasing the time average utility while consequently increasing congestion. This effect is mitigated as backlog grows at the source queues and flow control decisions become more conservative.

The routing and scheduling algorithm uses backpressure from neighboring nodes to *equalize differential backlog*. Indeed, allocating resources to maximize a product of transmission rate and differential backlog ensures that highly congested links receive larger transmission rates. This effect is most pronounced when congestion is large, so that the algorithm 'learns' from any past scheduling mistakes. Note that in cases when input rates are very low, there may be little information contained in the differential backlog values, and hence delay may be large even though overall congestion small. This problem can be solved by either restricting routing options to paths that make progress to the destination (which may also restrict network capacity), or by using an enhanced algorithm that weights differential backlog of each commodity by a hop-count estimate of the distance to the destination. For simplicity of exposition, here we analyze only the basic algorithm (see [1] for details on the enhanced strategy).

C. Algorithm Performance

To analyze the performance of the above CLC1 algorithm, we define the maximum transmission rate out of any node and into any node as follows:

$$\mu_{max}^{out} \triangleq \max_{[n, \vec{S}, \vec{\mu} \in \Gamma_{\vec{S}}]} \sum_{l \in \Omega_n} \mu_l \quad , \quad \mu_{max}^{in} \triangleq \max_{[n, \vec{S}, \vec{\mu} \in \Gamma_{\vec{S}}]} \sum_{l \in \Theta_n} \mu_l$$

We further define the value μ_{sym} as the largest rate that is simultaneously supportable by all sessions (n, c) :

$$\mu_{sym} \triangleq \text{ Largest scalar such that } (\mu_{sym}) \in \Lambda \qquad (8)$$

While the parameter μ_{sym} does not appear to be related to our design objectives, it will unexpectedly arise in the delay analysis. For simplicity of exposition, we assume channel states are i.i.d. every timeslot,[2] and let $\pi_{\vec{S}}$ represent the probability that $\vec{S}(t) = \vec{S}$. Assume utilities $g_{nc}(r)$ are non-negative, non-decreasing, and concave, and define $G_{max} \triangleq \max_{[n, \sum_c r_{nc} \leq R_n^{max}]} \sum_c g_{nc}(r_{nc})$.

Theorem 1: If channel states are i.i.d. over timeslots and all active reservoirs have infinite backlog, then for any flow parameter $V > 0$ the CLC1 algorithm stabilizes the network and yields time average congestion bound:

$$\overline{\sum_{nc} U_n^{(c)}} \leq \frac{N(B + VG_{max})}{2\mu_{sym}} \qquad (9)$$

[2]The algorithms developed in this paper yield similar results for general ergodic channel processes, with modified but more involved expressions for average delay [1].

where:

$$\overline{\sum_{nc} U_n^{(c)}} \triangleq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\left[\sum_{nc}\mathbb{E}\left\{U_n^{(c)}(\tau)\right\}\right]$$
$$B \triangleq \left(\mu_{max}^{in} + \frac{1}{N}\sum_{n=1}^{N} R_n^{max}\right)^2 + (\mu_{max}^{out})^2 \quad (10)$$

Further, network performance satisfies:

$$\sum_{nc} g_{nc}(\overline{r}_{nc}) \geq \sum_{nc} g_{nc}(r_{nc}^*) - \frac{BN}{V} \quad (11)$$

where (r_{nc}^*) is the optimal solution of (1) subject to constraint (2).

The above result holds for all $V > 0$. Thus, the value of V can be chosen so that BN/V is arbitrarily small, resulting in achieved utility that is arbitrarily close to optimal. This performance comes at the cost of a linear increase in network congestion with the parameter V. By Little's theorem, average queue backlog is proportional to average bit delay, and hence performance can be pushed towards optimality with a corresponding tradeoff in end-to-end network delay.

We note that although the CLC1 policy assumes all active sessions have unlimited backlog in their reservoirs, in practice the policy yields similar performance when input rate matrices (λ_{nc}) are finite. This holds because for many networks the policy either stabilizes all queues and reservoirs (yielding optimal throughput performance) or leads to instability in all active reservoirs (creating an effective 'infinite backlog' scenario because these unstable reservoirs always have sufficient data to be scheduled).

The proof of Theorem 1 follows from a novel Lyapunov drift argument, where the utility metric is incorporated into the drift condition so that stability and utility optimization can be simultaneously achieved. This analysis is provided in Section IV. In the following we consider the implications of this result.

D. Maximum Throughput and the Threshold Rule

Suppose utilities are linear, so that $g_{nc}(r) = \alpha_{nc} r$ for some non-negative weights α_{nc}. The resulting objective is to maximize the weighted sum of throughput, and the resulting FLOW algorithm has a simple threshold form, where some commodities receive as much of the R_n^{max} delivery rate as possible, while others receive none. In the special case where the user at node n desires communication with a single destination node c_n (so that $g_{nc}(r) = 0$ for all $c \neq c_n$), the flow control algorithm (7) reduces to maximizing $V\alpha_{nc_n} r - 2U_n^{(c_n)} r$ subject to $0 \leq r \leq R_n^{max}$, and the solution is the following threshold rule:

$$R_{nc_n}(t) = \begin{cases} R_n^{max} & \text{if } U_n^{(c_n)}(t) \leq \frac{V\alpha_{nc_n}}{2} \\ 0 & \text{otherwise} \end{cases}$$

The qualitative structure of this flow control rule is intuitive: When backlog in the source queue is large, we should refrain from sending new data. The simple threshold form is qualitatively similar to the threshold scheduling rule developed in [27] for server scheduling in a downlink with ON/OFF channels and deterministic constraints on the channel states and packet arrivals. Specifically, the analysis of [27] demonstrates that there exists a threshold T such that serving the longest queue maximizes throughput, where all queues with backlog greater than T are treated as having backlog that is equal to this threshold. Although the structure of the downlink scheduling problem in [27] is different from our problem structure, as are the analytical techniques and resulting scheduling rules, the objective of maximizing a weighted sum of throughput is the same, and hence it is interesting that both sets of results yield threshold-type policies.

E. Proportional Fairness and the $1/U$ Rule

Consider now utility functions of the form $g_{nc}(r) = \log(1+r_{nc})$. It is shown in [10] that maximizing a sum of such utilities over any convex set Λ leads to *proportional fairness*.[3] In the special case when there is only one destination c_n for each user n, the flow control algorithm reduces to maximizing $V\log(1+r) - 2U_n^{(c_n)} r$ subject to $0 \leq r \leq R_n^{max}$, which leads to the following '$1/U$' flow control function:

$$R_{nc_n}(t) = \min\left[\max\left[\frac{V}{2U_n^{(c_n)}(t)} - 1, 0\right], R_n^{max}\right]$$

Here we see that the flow control valve restricts flow according to a continuous function of the backlog level at the source queue, being less conservative in its admission decisions when backlog is low and more conservative when backlog is high.

One drawback of this $1/U$ policy is that the resulting flow control variables $R_{nc}(t)$ are real numbers (not necessarily integers or integer multiples of a given packet length), and hence it is implicitly assumed that packets can be fragmented for admission to the network. This problem arises in the CLC1 algorithm whenever the utility function is non-linear. In Section V, a modified algorithm CLC2 is presented that overcomes this problem by allowing admissions to be restricted to integer multiples of a common packet length, without loss of optimality.

F. Mechanism Design and Network Pricing

The flow control policy (7) has a simple interpretation in terms of network pricing. Specifically, consider a scenario where the $g_{nc}(r)$ functions are measured in units of dollars, representing the amount the user at source node n is willing to pay for rate r service to destination c. The *social optimum operating point* (r_{nc}^*) is defined as the point that maximizes the sum of utilities $\sum_{nc} g_{nc}(r_{nc})$ subject to $(r_{nc}) \in \Lambda$. Every timeslot, each user n determines the amount of data $R_{nc}(t)$ it desires to send based on a per-unit price $PRICE_{nc}(t)$ charged by the network. The transaction between user and network takes place in a distributed fashion at each node n. We assume all users are 'greedy' and send data every timeslot by maximizing total utility minus total cost, subject to an R_n^{max}

[3] Strictly speaking, the proportionally fair allocation seeks to maximize $\sum_{nc}\log(r_{nc})$, leading to $\sum_{nc}\frac{\overline{r}_{nc}^{opt} - r_{nc}}{\overline{r}_{nc}^{opt}} \geq 0$ for any other operating point $(r_{nc}) \in \Lambda$. We use non-negative utilities $\log(1+r)$, and thereby obtain a proportionally fair allocation with respect to the quantity $\overline{r}_{nc}^{opt} + 1$, leading to $\sum_{nc}\frac{\overline{r}_{nc}^{opt} - r_{nc}}{\overline{r}_{nc}^{opt}+1} \geq 0$.

constraint imposed by the network. That is, each user n selects $R_{nc}(t) = r_{nc}$, where the r_{nc} values solve:

$$\begin{array}{rl} \text{Maximize :} & \sum_c \left[g_{nc}(r_{nc}) - PRICE_{nc}(t) r_{nc}\right] \\ \text{Subject to:} & \sum_c r_{nc} \leq R_n^{max} \end{array} \quad (12)$$

Consider now the following dynamic pricing strategy used at each network node n:

$$PRICE_{nc}(t) = \frac{2U_n^{(c)}(t)}{V} \text{ dollars/bit} \quad (13)$$

We note that this pricing strategy is independent of the particular $g_{nc}(r)$ functions, and so the network does not require knowledge of the user utilities. Using this pricing strategy in (12), it follows that users naturally send according to processes $R_{nc}(t)$ that exactly correspond to the FLOW algorithm (7), and hence the performance bounds (9) and (11) are satisfied.

IV. Performance Analysis

Here we prove Theorem 1. We first develop a novel Lyapunov drift result enabling stability and performance optimization to be performed using a single drift analysis.

A. Lyapunov Drift with Utility Metric

Let $\underline{U}(t) = (U_n^{(c)}(t))$ represent a process of queue backlogs, and define the *Lyapunov function* $L(\underline{U}) = \sum_{nc}(U_n^{(c)})^2$. Let $R_{nc}(t)$ represent the input process driving the system, and suppose these values are bounded so that $\sum_c g_{nc}(R_{nc}(t)) \leq G_{max}$ for all n and all t (for some value G_{max}). Assume utility functions $g_{nc}(r)$ are non-negative and concave, and let (r_{nc}^*) represent a 'target throughput' matrix with ideal utility $\sum_{nc} g_{nc}(r_{nc}^*)$.

Lemma 1: (Lyapunov Drift) If there are positive constants V, ϵ, B such that for all timeslots t and all unfinished work matrices $\underline{U}(t)$, the Lyapunov drift satisfies:

$$\begin{aligned} \Delta(\underline{U}(t)) \triangleq & \mathbb{E}\left\{L(\underline{U}(t+1)) - L(\underline{U}(t)) \mid \underline{U}(t)\right\} \leq \\ & B - \epsilon \textstyle\sum_{nc} U_n^{(c)}(t) - V \sum_{nc} g_{nc}(r_{nc}^*) \\ & + V \textstyle\sum_{nc} \mathbb{E}\left\{g_{nc}(R_{nc}(t)) \mid \underline{U}(t)\right\} \end{aligned}$$

then the system is stable, and time average backlog and time average performance satisfies:

$$\overline{\sum_{nc} U_n^{(c)}} \leq (B + VNG_{max})/\epsilon \quad (14)$$

$$\sum_{nc} g_{nc}(\overline{r}_{nc}) \geq \sum_{nc} g_{nc}(r_{nc}^*) - B/V \quad (15)$$

where: $\overline{\sum_{nc} U_n^{(c)}} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{nc} \mathbb{E}\left\{U_n^{(c)}(\tau)\right\}$

$$\overline{r}_{nc} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left\{R_{nc}(\tau)\right\}$$

Proof: The proof is given in Appendix A. □

To prove Theorem 1, we first develop an expression for Lyapunov drift from the queueing dynamics (6). To start, note that any general queue with backlog $U(t)$ and queueing law $U(t+1) = \max[U(t) - \mu(t), 0] + A(t)$ has a Lyapunov drift given by:

$$\begin{aligned} \mathbb{E}\left\{U^2(t+1) - U^2(t) \mid U(t)\right\} \leq & \mu_{max}^2 + A_{max}^2 \\ & - 2U(t)\mathbb{E}\left\{\mu(t) - A(t) \mid U(t)\right\} \end{aligned} \quad (16)$$

where A_{max} and μ_{max} are upper bounds on the arrival and server variables $A(t)$ and $\mu(t)$. This well known fact follows simply by squaring the queueing equation and taking expectations. Applying the general formula (16) to the specific queueing law (6) for queue (n, c) and summing the result over all (n, c) pairs yields the following expression for Lyapunov drift (see [1] [16] for details):

$$\begin{aligned} \Delta(\underline{U}(t)) \leq NB - 2\sum_{nc} U_n^{(c)}(t) \mathbb{E}\Bigg\{ & \sum_{l\in\Omega_n} \mu_l^{(c)}(t) \\ & - \sum_{l\in\Theta_n} \mu_l^{(c)}(t) - R_{nc}(t) \mid \underline{U}(t) \Bigg\} \end{aligned} \quad (17)$$

where B is defined in (10).

Now define the *network function* $\Phi(\underline{U}(t))$ and the *flow function* $\Psi(\underline{U}(t))$ as follows:

$$\Phi(\underline{U}(t)) \triangleq 2\sum_{nc} U_n^{(c)} \mathbb{E}\left\{\sum_{l\in\Omega_n} \mu_l^{(c)} - \sum_{l\in\Theta_n} \mu_l^{(c)} \mid \underline{U}\right\} \quad (18)$$

$$\Psi(\underline{U}(t)) \triangleq \sum_{nc} \mathbb{E}\left\{V g_{nc}(R_{nc}) - 2U_n^{(c)} R_{nc} \mid \underline{U}\right\} \quad (19)$$

where we have represented $\underline{U}(t), \mu_l^{(c)}(t)$, and $R_{nc}(t)$ as $\underline{U}, \mu_l^{(c)}$, and R_{nc} for notational convenience. Adding and subtracting the optimization metric $V\sum_{nc}\mathbb{E}\left\{g_{nc}(R_{nc}) \mid \underline{U}\right\}$ to the right hand side of (17) yields:

$$\begin{aligned} \Delta(\underline{U}(t)) \leq & NB - \Phi(\underline{U}(t)) - \Psi(\underline{U}(t)) \\ & + V \sum_{nc} \mathbb{E}\left\{g_{nc}(R_{nc}(t)) \mid \underline{U}\right\} \end{aligned} \quad (20)$$

The CLC1 policy is designed to minimize the second and third terms on the right hand side of (20) over all possible routing, resource allocation, and flow control policies. Indeed, it is clear that the flow control strategy (7) maximizes $\Psi(\underline{U}(t))$ over all feasible choices of the $R_{nc}(t)$ values (compare (7) and (19)). That $\Phi(\underline{U}(t))$ is maximized by CLC1 is proven in [1] by switching the sums to express $\Phi(\underline{U}(t))$ in terms of differential backlog.

B. A Near-Optimal Operating Point

In order to use the Lyapunov drift result to establish the performance of the CLC1 algorithm, it is important to first compare performance to the utility of a *near-optimal* solution to the optimization problem (1)-(3). Specifically, for any $\epsilon > 0$, we define the set Λ_ϵ as follows:

$$\Lambda_\epsilon \triangleq \{(r_{nc}) \mid (r_{nc} + \epsilon) \in \Lambda, r_{nc} \geq 0 \text{ for all } (n, c)\ \}$$

Thus, the set Λ_ϵ can be viewed as the resulting set of rate matrices within the network capacity region when an "ϵ-layer" of the boundary is stripped away. Note that this set is non-empty whenever $\epsilon < \mu_{sym}$, defined in (8). The near-optimal

operating point $(r^*_{nc}(\epsilon))$ is defined as the optimal solution to the following optimization problem:[4]

$$\begin{aligned} \text{Maximize :} \quad & \textstyle\sum_{nc} g_{nc}(r_{nc}) \\ \text{Subject to:} \quad & (r_{nc}) \in \Lambda_\epsilon \\ & (r_{nc}) \leq (\lambda_{nc}) \end{aligned} \tag{21}$$

This optimization differs from the optimization in (1)-(3) in that the set Λ is replaced by the set Λ_ϵ. In [1] it is shown that $\Lambda_\epsilon \to \Lambda$ as $\epsilon \to 0$, and that:

$$\sum_{nc} g_{nc}(r^*_{nc}(\epsilon)) \to \sum_{nc} g_{nc}(r^*_{nc}) \quad \text{as } \epsilon \to 0 \tag{22}$$

C. Derivation of Theorem 1

The proof of Theorem 1 relies on the following two lemmas. Lemma 2 is proven in [1] [16], and Lemma 3 is proven at the end of this subsection.

Lemma 2: If the channel process $\vec{S}(t)$ is i.i.d. over timeslots, then for any ϵ in the open interval $(0, \mu_{sym})$, allocating resources and routing according to CLC1 yields:

$$\Phi(\underline{U}(t)) \geq 2 \sum_{nc} U^{(c)}_n(t)(r^*_{nc}(\epsilon) + \epsilon)$$

where $(r^*_{nc}(\epsilon))$ is the optimal solution of problem (21).

Lemma 3: If the channel process is i.i.d. over timeslots and all reservoirs are infinitely backlogged, then for any $\epsilon \in (0, \mu_{sym})$ the flow control algorithm of CLC1 yields:

$$\Psi(\underline{U}(t)) \geq V \sum_{nc} g_{nc}(r^*_{nc}(\epsilon)) - 2 \sum_{nc} U^{(c)}_n(t) r^*_{nc}(\epsilon)$$

Plugging the bounds of Lemmas 2 and 3 directly into the drift expression (20) yields:

$$\begin{aligned} \Delta(\underline{U}(t)) \leq NB &- 2 \sum_{nc} U^{(c)}_n(t)(r^*_{nc}(\epsilon) + \epsilon) \\ &- V \sum_{nc} g_{nc}(r^*_{nc}(\epsilon)) + 2 \sum_{nc} U^{(c)}_n(t) r^*_{nc}(\epsilon) \\ &+ V \sum_{nc} \mathbb{E}\left\{ g_{nc}(R_{nc}(t)) \mid \underline{U} \right\} \end{aligned}$$

Canceling common terms yields:

$$\begin{aligned} \Delta(\underline{U}(t)) \leq NB &- 2\epsilon \sum_{nc} U^{(c)}_n(t) - V \sum_{nc} g_{nc}(r^*_{nc}(\epsilon)) \\ &+ V \sum_{nc} \mathbb{E}\left\{ g_{nc}(R_{nc}(t)) \mid \underline{U} \right\} \end{aligned}$$

The above drift expression is in the exact form specified by Lemma 1. Thus, network congestion satisfies:

$$\overline{\sum_{nc} U^{(c)}_n} \leq (NB + VNG_{max})/(2\epsilon) \tag{23}$$

and time average performance satisfies:

$$\sum_{nc} g_{nc}(\overline{r}_{nc}) \geq \sum_{nc} g_{nc}(r^*_{nc}(\epsilon)) - NB/V \tag{24}$$

[4]Note that the final constraint $(r_{nc}) \leq (\lambda_{nc})$ is satisfied automatically in the case of infinite traffic demand. We include the constraint here as this optimization is also important in the treatment of general traffic matrices (λ_{nc}) in Section V.

The performance bounds in (23) and (24) hold for any value ϵ such that $0 < \epsilon < \mu_{sym}$. However, the particular choice of ϵ only affects the bound calculation and does not affect the CLC1 control policy or change any sample path of system dynamics. We can thus optimize the bounds separately over all possible ϵ values. The bound in (24) is clearly maximized by taking a limit as $\epsilon \to 0$, yielding by (22): $\sum_{nc} g_{nc}(\overline{r}_{nc}) \geq \sum_{nc} g_{nc}(r^*_{nc}) - NB/V$. Conversely, the bound in (23) is minimized as $\epsilon \to \mu_{sym}$, yielding: $\overline{\sum_{nc} U^{(c)}_n} \leq (NB + VNG_{max})/(2\mu_{sym})$. This proves Theorem 1. □

We complete the analysis by proving Lemma 3.

Proof. (Lemma 3) By definition, the flow control policy (7) maximizes $\Psi(\underline{U}(t))$ over all possible strategies [compare (7) and (19)]. Now plug into (19) the particular strategy $R_{nc}(t) = r^*_{nc}(\epsilon)$ for all t. This is a *valid strategy* because 1) all reservoirs are assumed to be infinitely backlogged, so there are always $r^*_{nc}(\epsilon)$ units of data available, and 2) $\sum_c r^*_{nc}(\epsilon) \leq R^{max}_n$ (because $(r^*_{nc}(\epsilon)) \in \Lambda$). Thus:

$$\Psi(\underline{U}(t)) \geq \sum_{nc} \left[V g_{nc}(r^*_{nc}(\epsilon)) - 2U^{(c)}_n(t) r^*_{nc}(\epsilon) \right] \quad \square$$

V. Scheduling with Arbitrary Input Rates

The algorithm CLC1 assumes there is always an amount of data $R_{nc}(t)$ available in reservoir (n, c), where the flow variable $R_{nc}(t)$ is chosen only with respect to the R^{max}_n constraint. Here we assume that all reservoirs have a finite (possibly zero) buffer for data storage, and let $L_{nc}(t)$ represent the current backlog in the reservoir buffer. The flow control decisions are now subject to the additional constraint $R_{nc}(t) \leq L_{nc}(t) + A_{nc}(t)$ (where $A_{nc}(t)$ is the amount of new commodity c data exogenously arriving to node n at slot t). Any arriving data that is not immediately admitted to the network is stored in the reservoir, or dropped if the reservoir has no extra space.

Assume the $A_{nc}(t)$ arrivals are i.i.d. over timeslots with arrival rates $\lambda_{nc} = \mathbb{E}\{A_{nc}(t)\}$. It can be shown that for any matrix (λ_{nc}) (possibly outside of the capacity region), modifying the CLC1 flow algorithm to maximize (7) subject to the additional reservoir backlog constraint yields the same performance guarantees (9) and (11) *when utility functions are linear* [1]. For nonlinear utilities, such a strategy can be shown to maximize the time average of $\sum_{nc} \mathbb{E}\{g_{nc}(R_{nc}(t))\}$ over all strategies that make immediate admission/rejection decisions upon arrival, but may not necessarily maximize $\sum_{nc} g_{nc}(\mathbb{E}\{R_{nc}(t)\})$, which is the utility metric of interest. We solve this problem with a novel technique of defining additional *flow state variables* $Z_{nc}(t)$. The result can be viewed as a general framework for stochastic network optimization.

Define *flow state variables* $Z_{nc}(t)$ for each reservoir (n, c), and assume $Z_{nc}(0) = VR^{max}_n/2$ for all (n, c). For each flow control process $R_{nc}(t)$, we define a new process $Y_{nc}(t)$ as follows:

$$Y_{nc}(t) \triangleq R^{max}_n - R_{nc}(t) \tag{25}$$

and note that $Y_{nc}(t) \geq 0$ for all t. The $Y_{nc}(t)$ variables represent the difference between the maximum value and the actual value of admitted data on session (n, c). The

$Z_{nc}(t)$ state variables are updated every slot according to the following 'queue-like' iteration:

$$Z_{nc}(t+1) = \max[Z_{nc}(t) - \gamma_{nc}(t), 0] + Y_{nc}(t) \quad (26)$$

where $\{\gamma_{nc}(t)\}$ are additional flow control decision variables. Define the 'cost' function:

$$h_{nc}(\gamma) \triangleq g_{nc}(R_n^{max}) - g_{nc}(R_n^{max} - \gamma) \quad (27)$$

Let $\overline{\gamma}_{nc}$ represent the time average value of the decision variables $\gamma_{nc}(t)$. We design a policy to stabilize the network queues $U_n^{(c)}(t)$ *and* the flow state 'queues' $Z_{nc}(t)$ while minimizing the cost $\sum_{nc} h_{nc}(\overline{\gamma}_{nc})$. The intuitive interpretation of this goal is as follows: If the $Z_{nc}(t)$ queues are stabilized, it must be the case that the time average of the 'server process' $\gamma_{nc}(t)$ is greater than or equal to the time average of the 'arrival process' $Y_{nc}(t)$: $\overline{Y}_{nc} \leq \overline{\gamma}_{nc}$. From (25), this implies $\overline{r}_{nc} \geq R_n^{max} - \overline{\gamma}_{nc}$, and hence:

$$\begin{aligned} \sum_{nc} h_{nc}(\overline{\gamma}_{nc}) &= \sum_{nc} g_{nc}(R_n^{max}) - \sum_{nc} g_{nc}(R_n^{max} - \overline{\gamma}_{nc}) \\ &\geq \sum_{nc} g_{nc}(R_n^{max}) - \sum_{nc} g_{nc}(\overline{r}_{nc}) \end{aligned}$$

Thus, minimizing $\sum_{nc} h_{nc}(\overline{\gamma}_{nc})$ over all feasible $\overline{\gamma}_{nc}$ values is intimately related to maximizing $\sum_{nc} g_{nc}(\overline{r}_{nc})$ over all feasible $\overline{r}_{nc}$ values.

Cross Layer Control Policy 2 (CLC2): Every timeslot and for each node n, choose $R_{nc}(t) = r_{nc}$ to solve:

$$\begin{aligned} \text{Maximize:} \quad & \textstyle\sum_c [\frac{Z_{nc}(t)}{N} - U_n^{(c)}(t)] r_{nc} \\ \text{Subject to:} \quad & \textstyle\sum_c r_{nc} \leq R_n^{max} \\ & r_{nc} \leq L_{nc}(t) + A_{nc}(t) \end{aligned} \quad (28)$$

Additionally, the flow controllers at each node n choose $\gamma_{nc}(t)$ for each session (n, c) to solve:

$$\begin{aligned} \text{Maximize:} \quad & V g_{nc}(R_n^{max} - \gamma_{nc}) + \tfrac{2Z_{nc}(t)}{N}\gamma_{nc} \\ \text{Subject to:} \quad & 0 \leq \gamma_{nc} \leq R_n^{max} \end{aligned} \quad (29)$$

The flow states $Z_{nc}(t)$ are then updated according to (26). Routing and resource allocation within the network is the same as in CLC1.

The optimization of $R_{nc}(t)$ in (28) is solved by a simple 'bang-bang' control policy, where no data is admitted from reservoir (n, c) if $U_n^{(c)}(t) > Z_{nc}(t)/N$, and otherwise as much data as possible is delivered from the commodities of node n with the largest non-negative values of $[Z_{nc}(t)/N - U_n^{(c)}(t)]$, subject to the R_n^{max} constraint. These bang-bang decisions also enable the strategy to be implemented optimally in systems where admitted data is constrained to integral units, a feature that CLC1 does not have. The $\gamma_{nc}(t)$ variable assignment in (29) involves maximizing a concave function of a single variable, and can be solved easily by finding the critical points over $0 \leq \gamma_{nc} \leq R_n^{max}$. For example, if $g_{nc}(r) = \log(1 + r)$, it can be shown that:

$$\gamma_{nc} = \min\left[\max\left[1 + R_n^{max} - \frac{VN}{2Z_{nc}(t)}, 0\right], R_n^{max}\right]$$

Suppose channels and arrivals are i.i.d. over timeslots, and let $\lambda_{nc} = \mathbb{E}\{A_{nc}(t)\}$. For simplicity of exposition, we further assume that new arrivals to node n are deterministically bounded by R_n^{max}, so that $\sum_c A_{nc}(t) \leq R_n^{max}$ every slot.

Theorem 2: For arbitrary rate matrices (λ_{nc}) (possibly outside of the capacity region), for any $V > 0$, and for any reservoir buffer size (possibly zero), the CLC2 algorithm stabilizes the network and yields a congestion bound:

$$\overline{\sum_{nc} U_{nc}} \leq \frac{NB + 2\sum_n (R_n^{max})^2 + VNG_{max}}{2\mu_{sym}}$$

Further, the time average utility satisfies:

$$\sum_{nc} g_{nc}(\overline{r}_{nc}) \geq \sum_{nc} g_{nc}(r_{nc}^*) - \frac{NB + 2\sum_n (R_n^{max})^2}{V}$$

The proof is given in Appendix B. Although the reservoir buffer size does not impact the above result, in practice a large reservoir buffer helps to 'smooth out' any dropped data (preferably by buffering all data corresponding to the same file) so that FIFO admission can be maintained amongst the various network flows.

VI. Simulation Results

Here we simulate the CLC2 policy for three simple network examples. We begin with the 2-queue downlink example of Section II. Packets arrive from each stream according to Bernoulli processes, and we assume they are placed into infinite buffer storage reservoirs. As before, we assume channel probabilities are given by $p_1 = 0.5, p_2 = 0.6$, and consider one hundred different rate pairs (λ_1, λ_2) that are linearly scaled towards the point $(0.5, 1.0)$. For each point we simulate the CLC2 algorithm for 3 million timeslots, using $V = 10000$, $R_n^{max} = 2$, and $g_1(r) = g_2(r) = \log(1 + r)$. Note that in this case, we have $\mu_{max}^{in} = 0$, $\mu_{max}^{out} = 1$, so by (10) we have $B = 5$. Thus, for $V = 10000$ we are ensured by Theorem 2 that the resulting utility associated with each rate vector (λ_1, λ_2) differs from the optimum utility by no more than $(5+8)/V = 0.0013$ (note that $N = 1$ for this simple example, as there is only 1 transmitting node). The simulation results are shown in Fig. 3(a), where the achieved throughput increases to the capacity boundary and then moves directly to the fair point $(0.4, 0.4)$.

In Fig. 3(b) we treat the same situation with the exception that utility for user 2 is modified to $1.28\log(1 + r)$. This illustrates the ability to handle *priority service*, as user 2 traffic is given favored treatment without starving user 1 traffic. From the figure, we see that as input rates are increased the resulting throughput reaches the capacity boundary and then *moves in a new direction*, settling and remaining on the new optimal operating point $(0.23, 0.57)$ once input rates dominate this point.

Note that for this example, we have $\mu_{sym} = 0.4$ and $G_{max} = 0.784$. Thus, by Theorem 2, we know:

$$\overline{U}_1 + \overline{U}_2 \leq \frac{13 + 0.784V}{0.8} \quad (30)$$

The above bound holds for any input rate vector (λ_1, λ_2), including vectors that are far outside of the capacity region. We next keep the same utility as in Fig. 3(b) but fix the input rate to $(\lambda_1, \lambda_2) = (0.5, 1.0)$, which dominates the

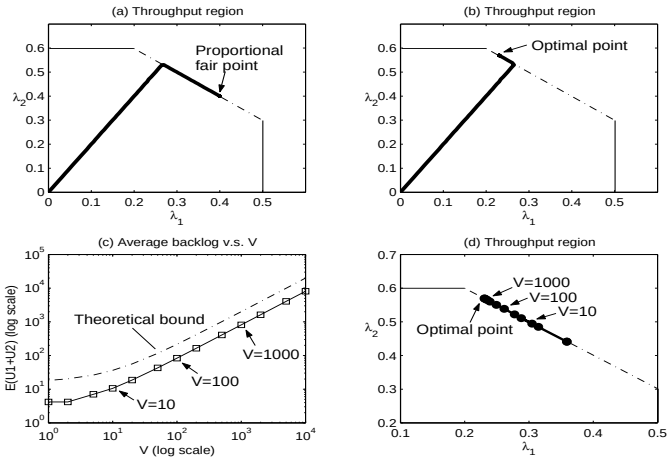

Fig. 3. Simulation of CLC2: (a) Linearly increasing (λ_1, λ_2) to $(0.5, 1.0)$ for $V = 10000$ and $g_1(r) = g_2(r) = \log(1 + r)$. (b) Modifying utility 2 to: $g_2(r) = 1.28\log(1 + r)$. (c)-(d) Fixing $(\lambda_1, \lambda_2) = (0.5, 1.0)$ and illustrating delay and throughput versus V.

Rates (λ_{ij})			Throughput (r_{ij})			Backlog $(\overline{U}_{ij})$		
.45	.1	.4	.450	.100	.399	3.3	2.4	3.6
.1	.7	.15	.100	.695	.148	2.4	2.9	2.7
.4	.15	.4	.399	.149	.400	3.6	2.7	3.4

(a) Simulation of a switch with feasible traffic

Rates (λ_{ij})			Throughput (r_{ij})			Backlog $(\overline{U}_{ij})$		
.9	.2	.3	.598	.100	.298	31.6	45.3	32.1
0	.4	.2	0	.399	.200	0	14.1	.29
0	.5	0	0	.500	0	0	14.2	0

(b) Simulation of an overloaded switch

Fig. 4. Simulation results for the CLC2 algorithm with $V = 100$ and zero reservoir buffers. Simulations were run over four million timeslots.

optimal operating point $(0.23, 0.57)$ (so that utility optimal control should achieve this point). In Fig. 3(c) we plot the resulting average queue congestion as V is varied from 1 to 10^4, together with the bound (30). As suggested by the bound, the delay grows linearly with V. In Fig. 3(d) we see how the achieved throughput of CLC2 approaches the optimal operating point $(0.23, 0.57)$ as V is increased.

A. Packet Switches

Here we consider a simple 3×3 packet switch with a crossbar switch fabric [22] [23]. Packets arrive from three different input ports, and each packet is destined for one of three output ports. We let λ_{ij} represent the rate of packets arriving to input i and destined for output j. All packets are stored in *virtual input queues* according to their destinations, and we let U_{ij} represent the current number of backlogged packets waiting at input i to be delivered to output j. The system is timeslotted, and the crossbar fabric limits scheduling decisions to *permutation matrices*, where no input can transfer more than one packet per slot, and no output can receive more than one packet per slot. Thus, the following *feasibility constraints* are required for stability:

$$\sum_{i=1}^{3} \lambda_{ij} \leq 1 \ \forall j \in \{1, 2, 3\} \ , \ \sum_{j=1}^{3} \lambda_{ij} \leq 1 \ \forall i \in \{1, 2, 3\}$$

We consider i.i.d. Bernoulli arrivals, and apply the CLC2 algorithm using $\log(1 + r)$ utility functions, $V = 100$, and $R_{max} = 3$ (the maximum number of arrivals to a single input during a slot). In this example we assume that all reservoirs have zero buffers, so that admission/rejection decisions must be made immediately upon packet arrival. In Fig. 4(a) we present simulation results for a case when the sum rate to every input port and every output port is exactly 0.95. Note that average queue backlog is kept very low, while the resulting throughput is almost identical to the input rate. This illustrates that the CLC2 algorithm accepts almost all packets into the system, and accomplishes this without a-priori knowledge that the input traffic is feasible.

We next apply the same CLC2 algorithm to a switch where input port 1 and output port 2 are overloaded, as shown in Fig. 4(b). The resulting throughput from the simulation is given in the figure, and is almost indistinguishable from the utility maximizing solution of the optimization problem (1)-(3). The average backlog in all queues is less than or equal to 45.3 packets.

B. Heterogeneous Multi-hop Networks

Here we consider the multi-hop network of Fig. 1, consisting of wireless sensor nodes (nodes $\{6, \ldots, 9\}$), a wireline network, and a wireless basestation that transmits to two mobile users. All packets have fixed lengths. The wireline links are bidirectional and can transmit 3 packets in each direction during a single slot. The basestation node 0 can transmit to only one mobile user per slot, and the downlinks to each user independently vary between ON and OFF states according to Bernoulli processes, with equal likelihood of being ON or OFF. The wireless links of the sensor network are always ON, and can support one packet transfer per slot. However, due to interference between the various sensor nodes, we assume that only one sensor link can be activated per slot (including the outgoing wireless links of the access nodes 4 and 5).

We assume there are four independent sessions using the network: Two sessions originate from node 9 and consist of packets destined for nodes 3 and 1, and two sessions originate from node 4 and consist of packets destined for nodes 8 and 2. All arrival processes are i.i.d. and Bernoulli, with arrival rates $\lambda_{93} = \lambda_{91} = \lambda_{48} = \lambda_{42} = 0.7$.

These arrival rates are not supportable by the network. Indeed, note that all packets originating at node 9 must travel over at least 2 sensor links before reaching a wireline access node. Likewise, all data from the λ_{48} stream requires at least two hops through the sensor network. Because at most one sensor link can be activated on any timeslot, it follows that $2r_{93} + 2r_{91} + 2r_{48} \leq 1$ is a necessary condition for network stability, where r_{ij} is the throughput of (i, j) traffic. The

basestation places the following additional constraints on the network flows: $r_{91} \leq 1/2$, $r_{42} \leq 1/2$, and $r_{91} + r_{42} \leq 3/4$. It is not difficult to verify that these necessary conditions describe the feasible flows for the network, as the wired links to not impose further constraints. Assuming that all sessions have identical utility functions $g_{ij}(r) = \log(1 + r)$, the optimally fair flows are thus given by $r^*_{91} = r^*_{93} = r^*_{48} = 1/6 = 0.1667$, $r^*_{42} = 0.5$.

We implement CLC2 for this network, using $V = 1000$, $R_{max} = 2$, and assuming infinite buffer reservoirs. Note that sensor link activations are determined every slot by choosing the link with the largest differential backlog. The resulting average queue length, summed over all queues in the system, is 858.9 packets. The throughputs are: $r_{91} = 0.1658, r_{93} = 0.1662, r_{48} = 0.1678, r_{42} = 0.5000$. We note that this performance is not possible unless almost all packets take their optimal 2-hop paths through the sensor network, and hence the backpressure routing learns the optimal routes.

VII. Conclusions

We have presented a fundamental approach to stochastic network control for heterogeneous data networks. Simple strategies were developed that perform arbitrarily close to the optimally fair throughput point (regardless of the input traffic matrix), with a corresponding tradeoff in end-to-end network delay. The strategies involve resource allocation and routing decisions that are decoupled over the independent portions of the network, and flow control algorithms that are decoupled over independent control valves at every node. Flow controllers require knowledge only of the queue backlog in their respective source nodes. We note that this technique of implementing flow control only at the sources is crucial to ensure no network resources are wasted transmitting data that will eventually be dropped. It is remarkable that the overall strategy does not require knowledge of input rates, channel statistics, or the global network topology. Although i.i.d. assumptions were made to simplify exposition, the same policies can be shown to offer similar performance (with modified delay expressions) for arbitrary ergodic arrivals and channels, and are robust to cases when channel probabilities or arrival rates change over time [1]. We believe that such theory-driven networking strategies will impact the design and operation of future data networks.

Appendix A—Proof of Lemma 1

Proof: The drift condition for $\Delta(\underline{U}(t))$ in Lemma 1 holds for all timeslots t. Taking expectations over the distribution of $\underline{U}(t)$ and summing over $t \in \{0, \ldots, M-1\}$ yields:

$$\begin{aligned}\mathbb{E}\{L(\underline{U}(M)) - L(\underline{U}(0))\} \leq BM \\ -\epsilon \textstyle\sum_{\tau=0}^{M-1} \sum_{nc} \mathbb{E}\left\{U_n^{(c)}(\tau)\right\} - VM \sum_{nc} g_{nc}(r^*_{nc}) \\ +V \textstyle\sum_{\tau=0}^{M-1} \sum_{nc} \mathbb{E}\{g_{nc}(R_{nc}(\tau))\} \qquad (31)\end{aligned}$$

Using non-negativity of the Lyapunov function and the utility functions as well as the fact that $\sum_{nc} g_{nc}(R_{nc}(\tau)) \leq NG_{max}$, we rearrange the terms of (31) and divide by $M\epsilon$ to yield:

$$\frac{1}{M}\sum_{\tau=0}^{M-1}\sum_{nc}\mathbb{E}\left\{U_n^{(c)}(\tau)\right\} - \frac{\mathbb{E}\{L(\underline{U}(0))\}}{M\epsilon} \leq \frac{B + VNG_{max}}{\epsilon}$$

Taking limits as $M \to \infty$ yields the backlog bound (14). This backlog bound implies stability of all queues [1].

The utility bound (15) is proved similarly. Indeed, we again rearrange (31) and divide by MV to yield:

$$\begin{aligned}\textstyle\sum_{nc} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{g_{nc}(R_{nc}(\tau))\} \geq \sum_{nc} g_{nc}(r^*_{nc}) \\ - \frac{B + \mathbb{E}\{L(\underline{U}(0))\}/M}{V} \qquad (32)\end{aligned}$$

By concavity of $g_{nc}(r)$ together with Jensen's inequality, it follows that $\frac{1}{M}\sum_{\tau=0}^{M-1}\mathbb{E}\{g_{nc}(R_{nc}(\tau))\} \leq g_{nc}\left(\frac{1}{M}\sum_{\tau=0}^{M-1}\mathbb{E}\{R_{nc}(\tau)\}\right)$. Using this fact in (32) and taking limits as $M \to \infty$ yields the result. □

Appendix B—Proof of Theorem 2

Proof: Define the Lyapunov function $L(\underline{U}, \underline{Z}) = \sum_{nc} U_n^{(c)} + \frac{1}{N}\sum_{nc} Z_{nc}$. The drift expression for this function is given by summing the drift of the $U_n^{(c)}(t)$ queues and the $Z_{nc}(t)$ queues using the general formula (16), where the queueing laws are given by (6) and (26). Omitting arithmetic details for brevity, we have the following drift expression:

$$\begin{aligned}\Delta(\underline{U}(t), \underline{Z}(t)) \leq NB + 2\sum_n (R_n^{max})^2 - \Phi(\underline{U}(t)) \\ +2\sum_{nc}\mathbb{E}\left\{U_n^{(c)}(t)R_{nc}(t) + Y_{nc}(t)\frac{Z_{nc}(t)}{N} \mid \underline{U}, \underline{Z}\right\} \\ -\sum_{nc}\mathbb{E}\left\{2\frac{Z_{nc}(t)}{N}\gamma_{nc}(t) - Vh_{nc}(\gamma_{nc}(t)) \mid \underline{U}, \underline{Z}\right\} \\ -\sum_{nc} V\mathbb{E}\{h_{nc}(\gamma_{nc}(t)) \mid \underline{U}, \underline{Z}\}\end{aligned}$$

where we have added and subtracted the optimization metric $\sum_{nc} V\mathbb{E}\{h_{nc}(\gamma_{nc}(t)) \mid \underline{U}, \underline{Z}\}$ in the right hand side of the above expression. The CLC2 policy is designed *to minimize the third, fourth, and fifth terms on the right hand side of the above expression over all possible policies.* Indeed, we already know that the routing and resource allocation policy maximizes $\Phi(\underline{U}(t))$. The fourth term in the right hand side is minimized by the strategy (28) that chooses $R_{nc}(t)$ (considering the definition of $Y_{nc}(t)$ in (25)). The fifth term is minimized by the strategy (29) that chooses $\gamma_{nc}(t)$ (considering the definition of $h_{nc}(\gamma)$ given in (27)).

For a given $\epsilon \in (0, \mu_{sym})$, a bound on $\Phi(\underline{U}(t))$ is given by Lemma 2 in terms of values $(r^*_{nc}(\epsilon) + \epsilon)$. Now consider the following alternative flow control strategies: Fix $\gamma_{nc}(t) = R_n^{max} - r^*_{nc}(\epsilon) \triangleq \gamma^*_{nc}(\epsilon)$ for all slots t. Then, every timeslot independently admit all new arrivals $A_{nc}(t)$ with probability $p_{nc} = r^*_{nc}(\epsilon)/\lambda_{nc}$ (this is a valid probability (≤ 1) by problem (21), and the admitted data satisfies the R_n^{max} constraint by the deterministic arrival bound). This yields $\mathbb{E}\{R_{nc}(t) \mid \underline{U}, \underline{Z}\} = p_{nc}\mathbb{E}\{A_{nc}(t)\} = r^*_{nc}(\epsilon)$, and hence $\mathbb{E}\{Y_{nc}(t) \mid \underline{U}, \underline{Z}\} = R_n^{max} - r^*_{nc}(\epsilon) \triangleq \gamma^*_{nc}(\epsilon)$. Plugging these expectations into the third, fourth, and fifth terms of the above

drift expression maintains the bound and creates many terms that can be cancelled. The simplified drift expression becomes:

$$\Delta(\underline{U}(t)) \leq NB + 2\sum_n (R_n^{max})^2 - 2\epsilon \sum_{nc} U_n^{(c)}(t)$$
$$+V \sum_{nc} h_{nc}(\gamma_{nc}^*(\epsilon)) - V \sum_{nc} \mathbb{E}\left\{h_{nc}(\gamma_{nc}(t)) \mid \underline{U}, \underline{Z}\right\}$$

Plugging in the definitions of $\gamma_{nc}^*(\epsilon)$ and $h_{nc}(\gamma)$ yields:

$$\Delta(\underline{U}(t)) \leq NB + 2\sum_n (R_n^{max})^2 - V \sum_{nc} g_{nc}(r_{nc}^*(\epsilon))$$
$$-2\epsilon \sum_{nc} U_n^{(c)}(t) + V \sum_{nc} \mathbb{E}\left\{g_{nc}(R_n^{max} - \gamma_{nc}(t)) \mid \underline{U}, \underline{Z}\right\}$$

The above expression is in the exact form for application of Lemma 1, and it follows that unfinished work satisfies:

$$\overline{\sum_{nc} U_n^{(c)}} \leq \frac{NB + 2\sum_n (R_n^{max})^2 + VNG_{max}}{2\epsilon}$$

and performance satisfies:

$$\sum_{nc} g_{nc}(R_n^{max} - \overline{\gamma}_{nc}) \geq \sum_{nc} g_{nc}(r_{nc}^*(\epsilon))$$
$$- \frac{NB + 2\sum_n (R_n^{max})^2}{V}$$

However, it can similarly be shown that all $Z_{nc}(t)$ queues are stable, and hence $\overline{r}_{nc} \geq R_n^{max} - \overline{\gamma}_{nc}$ must hold [recall discussion after (27)]. The result of Theorem 2 follows by optimizing the performance bounds over $0 < \epsilon < \mu_{sym}$ in a manner similar to the proof of Theorem 1. □

REFERENCES

[1] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] R. Johari and J. N. Tsitsiklis. Network resource allocation and a congestion game. *Submitted to Math. of Oper. Research*, 2003.

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

[13] 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.

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

[15] 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.

[16] 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, pp. 89-103, January 2005.

[17] 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.

[18] E. M. Yeh and A. S. Cohen. Throughput and delay optimal resource allocation in multiaccess fading channels. *Proceedings of the International Symposium on Information Theory*, May 2003.

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

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

[21] N. Kahale and P. E. Wright. Dynamic global packet routing in wireless networks. *IEEE Proceedings of INFOCOM*, 1997.

[22] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *Proc. INFOCOM*, 1996.

[23] E. Leonardi, M. Melia, F. Neri, and M. Ajmone Marson. Bounds on average delays and queue size averages and variances in input-queued cell-based switches. *Proc. of IEEE INFOCOM*, 2001.

[24] S. Borst. User-level performance of channel-aware scheduling algorithms in wireless data networks. *IEEE INFOCOM*, 2003.

[25] D. N. Tse. Optimal power allocation over parallel broadcast channels. *Proceedings of the International Symposium on Information Theory*, June 1997.

[26] H. Kushner and P. Whiting. Asymptotic properties of proportional-fair sharing algorithms. *40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.

[27] V. Tsibonis, L. Georgiadis, and L. Tassiulas. Exploiting wireless channel state information for throughput maximization. *IEEE Proceedings of INFOCOM*, April 2003.

[28] U. C. Kozat, I. Koutsopoulos, and L. Tassiulas. A framework for cross-layer design of energy-efficient communication with qos provisioning in multi-hop wireless networks. *INFOCOM*, 2004.

[29] L. Li and A. Goldsmith. Capacity and optimal resource allocation for fading broadcast channels: Part i: Ergodic capacity. *IEEE Trans. Inform. Theory*, ppl. 1083-1102, March 2001.