A Novel Architecture for Reduction of Delay and Queueing Structure Complexity in the Back-pressure Algorithm

Loc Bui, R. Srikant, Alexander Stolyar

***Abstract*—The back-pressure algorithm is a well-known throughput-optimal algorithm. However, its implementation requires that each node has to maintain a separate queue for each commodity in the network, and only one queue is served at a time. This fact may lead to a poor delay performance even when the traffic load is not close to network capacity. Also, since the number of commodities in the network is usually very large, the queueing data structure has to be maintained at each node is respectively complex. In this paper, we present a solution to address both of the above issues in the case of *fixed-routing* network scenario where the route of each flow is chosen upon arrival. Our proposed architecture allows each node to maintain only per-neighbor queues, and moreover, improves the delay performance of the back-pressure algorithm.**

I. Introduction

Resource allocation in wireless networks is complicated due to the shared nature of wireless medium. One particular allocation algorithm called the *back-pressure algorithm* which encompasses several layers of the protocol stack from MAC to routing was proposed by Tassiulas and Ephremides, in their seminal paper [1]. The back-pressure algorithm was shown to be *throughput-optimal*, i.e., it can support any arrival rate vector which is supportable by any other resource allocation algorithm. Recently, it was shown that the back-pressure algorithm can be combined with congestion control to fairly allocate resources among competing users in a wireless network [2]–[7], thus providing a complete resource allocation solution from the transport layer to the MAC layer. While such a combined algorithm can be used to perform a large variety of resource allocation tasks, in this paper, we will concentrate on its application to routing and scheduling only.

Even though the back-pressure algorithm delivers maximum throughput by adapting itself to network conditions, there are several issues that have to be addressed before it can be widely deployed in practice. As stated in the original paper [1], the back-pressure algorithm requires centralized information and computation, and its computational complexity is too prohibitive for practice. Much progress has been made recently in easing the computational complexity and deriving decentralized heuristics. We refer the interested reader to [8], [9] and references within for some recent results along these lines. We do not consider complexity or decentralization issues in this paper; our proposed solutions can be approximated well by the solutions suggested in the above papers.

Besides complexity and decentralization issues which have received much attention recently, the back-pressure algorithm can also have poor delay performance. To understand that, let us consider the *fixed-routing* scenario where the route for each flow is chosen upon arrival by some standard multi-hop wireless network routing algorithm such as DSR or AODV and the back-pressure algorithm is simply used to schedule packets. In operation, the back-pressure algorithm assigns a weight to each flow on each link. The weight is equal to the flow's queue backlog at the transmitter of the link minus the flow's queue backlog at the receiver. The weight of a link is equal to the maximum weight of any flow that uses the link. The back-pressure algorithm then selects a schedule which maximizes the sum of the weights of the links included in the schedule. Under such an algorithm, for a link to be scheduled, its weight should be slightly larger than zero. Now, we consider a flow that traverses K links, and use an informal argument to show why it is very intuitive that the flow's total queue accumulation along its route should grow quadratically with the route length. The queue length at the destination for this flow is equal to zero. The queue length at the first upstream node from the destination will be some positive number, say, ϵ. The queue length at the second upstream node from the destination will be even larger and for the purposes of obtaining insight, let us say that it is 2ϵ. Continuing this reasoning further, the total queue length for the flow will be $\epsilon(1+2+\ldots+K) = \Theta(K^2)$. Thus, the total backlog on a path is intuitively expected to grow quadratically in the number of hops. On the other hand, suppose a fixed service rate is allocated to each flow on each link on its path, then the queue length at each hop will be roughly $O(1)$ depending on the utilization at that link. With such a fixed service rate allocation, the total end-to-end backlog should then grow linearly in the number of hops. However, such an allocation is possible only if the packet arrival rate generated by each flow is known to the network a priori. One of the contributions of this paper is to use counters called *shadow queues* introduced in [10] to allocate service rates to each flow on each link in an adaptive fashion without knowing the set of packet arrival rates.

We will also show that the concept of shadow queues can reduce the number of real queues maintained at each node sig-

An earlier version of this work was presented at the IEEE INFOCOM Conference, Rio de Janeiro, Brazil, April 2009.

L. Bui is with Department of Management Science and Engineering, Stanford University, Stanford, CA 94305, USA (e-mail: locbui@stanford.edu).

R. Srikant is with Department of Electrical and Computer Engineering and Coordinated Science Lab, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA (e-mail: rsrikant@illinois.edu).

A. Stolyar is with Bell Labs, Alcacel-Lucent, Murray Hill, NJ 07974, USA (e-mail: stolyar@research.bell-labs.com).

nificantly. Notice that in the case of fixed-routing, it is natural to implement back-pressure algorithm using per-flow queues, but it is also possible to implement it using per-destination queues. In other words, the traditional back-pressure algorithm requires either per-flow or per-destination queues. We will show that it is sufficient to maintain only *per-neighbor* queues at each node, instead of per-flow or per-destination queues required by the traditional back-pressure algorithm. In large networks, the number of flows (or destinations) is typically much larger compared to the number of neighbors of each node, thus using per-neighbor queues can result in significant reduction in implementation complexity.

An attempt to reduce the number of queues using clustering architecture was proposed in [11]. However, in [11], per-destination queues are still maintained at each node, one for each destination cluster and one for each node within its own cluster. Clustering also requires additional communication between clusters to ensure stability. The *shadow queueing* architecture in this paper is fundamentally different from that clustering approach: first, it allows *per-neighbor* queues to be maintained at each node, which leads to a drastic reduction in the number of real data queues to be implemented; and second, it has the additional benefit of delay reduction. The reader is also referred to [12] in which the authors tried to reduce delay of back-pressure algorithm by considering a simplified, sub-optimal version of the network utility maximization problem.

In summary, the main contributions of this paper are as follows:

- We present the disadvantages of back-pressure algorithm in the aspects of number of queues and delay performance. We formally characterize the delay performance of back-pressure algorithm using the number of hops as the metric. In particular, we show that, under the back-pressure algorithm, the worst-case total queueing backlog for any flow scales quadratically in its number of hops. (Section III.)
- We then propose the *shadow queueing* architecture to address those disadvantages of back-pressure algorithm. This architecture is twofold: first, it allows each node to maintain only per-neighbor FIFO queues instead of per-flow queues required by the back-pressure algorithm, and hence, reduces its implementation complexity; second, it significantly improves the delay performance of traditional back-pressure algorithm with a small cost of throughput degradation. (Section IV.)
- We establish the stabilities of both shadow and real queueing systems. The stability result for real queues are not obvious due to the fact that now each node maintains only per-neighbor FIFO queues. Because of the radically different architectures of real queues and shadow queues, to the best of our knowledge, there is no sample-path relationship between them which would allow us to conclude stability. We show that the stability is possible because of its connection to Bramson's stability result for FIFO queues [13]. (Section V.)
- We then present extensive simulation results to show significant improvements on delay performance of the proposed shadow queueing architecture. Although we cannot provide any analytical result, the simulation results confirm our intuition of its delay reduction from quadratic to linear (in terms of the number of hops). (Section VI.)

II. System Model

Let us consider a network modeled by a graph, $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of links. We assume that time is slotted, with a typical time slot denoted by t. If a link (n, m) is in $\mathcal{L}$, then it is possible to transmit packets from node n to node m subject to the interference constraints which will be described shortly. We will use both notations (n, m) and l interchangeably to indicate a network link.

We let $\mathcal{F}$ be the set of flows that share the network resources. Packets of each flow enter the network at one node, travel along multiple hops, and then exit the network at another node. In this paper, we only consider the ***fixed routing*** scenario, i.e., the route of each flow is pre-determined and fixed during the time of interest. For each $f \in \mathcal{F}$, let $L(f)$ denote the set of links forming the route of f, and let:

- $first(f) \in L(f)$ be the first ***link*** on the route of f;
- $last(f) \in L(f)$ be the last ***link*** on the route of f;
- $b(f) \in \mathcal{N}$ be the begin (entering) node, i.e., $b(f)$ belongs to $first(f)$; and,
- $e(f) \in \mathcal{N}$ be the end (exiting) node, i.e., $e(f)$ belongs to $last(f)$.

Also, for each pair (f, l) where $l \in L(f)$, let us define:

- $next(f, l) \in L(f)$ as the link after l in the route of f if $l \neq last(f)$; and,
- $prev(f, l) \in L(f)$ be the link before l in the route of f if $l \neq first(f)$.

We define a valid schedule $\pi = \left(c_1^\pi, c_2^\pi, \ldots, c_{|\mathcal{L}|}^\pi\right)$ to be a set of link rates (measured in terms of number of packets) that can be simultaneously supported. Note that due to the interference between links, for each π, some c_l^π could be zero. Moreover, we make a natural and nonrestrictive assumption that if π is a valid schedule, then if we replace any subset of its components by zeros, the modified schedule is valid as well. We also assume that c_l^π is upper-bounded by some c_{max} for any π and l. Let Γ be the set of all possible valid schedules, and $co(\Gamma)$ denote the convex hull of Γ.

Let Λ denote the network's *capacity region*, which is defined as the set of all flow rates that are supportable by the network, given the set of flows and their corresponding routes. In particular, $\lambda = \{\lambda_f\}_{f \in \mathcal{F}} \in \Lambda$ if there exists a $\mu = \{\mu_l\}_{l \in \mathcal{L}}$ such that:

- $\lambda_f \leq \mu_{first(f)}$, $\forall f \in \mathcal{F}$, and,
- $\mu_l \leq \mu_{next(f,l)}$, $\forall f \in \mathcal{F}$, $l \in L(f)$, $l \neq last(f)$, and,
- $\mu \in co(\Gamma)$.

The traffic in the network can be *elastic* or *inelastic*. If the traffic is *inelastic*, i.e., the flows' rates are fixed (and within the capacity region), then the goal is to route/schedule the traffic through the network while ensuring that the queues in the network are stable. If the traffic is *elastic*, then the goal is to allocate the network's resources to all flows in some fair manner. More precisely, suppose that each flow has a utility

function associated with it. The utility function of flow f, denoted by $U_f(\cdot)$, is defined as a function of the data rate x_f sent by flow f, and assumed to be concave and nondecreasing. The goal, in the case of elastic traffic, is to determine the optimal solution to the following resource allocation problem:

$$\begin{aligned} \max \quad & \sum_{f\in\mathcal{F}} U_f(x_f) \\ \text{s.t.} \quad & x \in \Lambda, \end{aligned} \tag{1}$$

where Λ is the *capacity region* described above.

III. Traditional Back-Pressure Algorithm

A. Description and Drawbacks

We now describe the traditional back-pressure algorithm, which was first proposed in [1]. As we mentioned earlier, one can implement the traditional back-pressure algorithm using either per-flow or per-destination queues in the fixed-routing scenario. However, for the sake of simplicity, we will consider the per-flow implementation only from now on. In the per-flow implementation, each node maintains a separate queue for each flow going through it. The queue maintained at node n for flow f is for buffering packets of f which reach n. Let $Q_n^f[t]$ denote the length of that queue at the beginning of time slot t. By convention, $Q_{e(f)}^f[t] = 0,\ \forall t$. The traditional back-pressure algorithm is as follows.

1) Back-pressure scheduling by the network:
At time slot t,

- Each link looks at the maximum differential backlog of all flows going through that link:

$$w_{nm}[t] = \max_{f:(n,m)\in L(f)} \left(Q_n^f[t] - Q_m^f[t]\right). \tag{2}$$

- Back-pressure scheduling:

$$\pi^*[t] = \max_{\pi\in\Gamma} \sum_{(n,m)} \pi_{nm} w_{nm}[t]. \tag{3}$$

- If the schedule π^* says, for example, to send c_{nm}^π shadow packets over link (n,m), then link (n,m) transmits up to c_{nm}^π packets from the queue of the flow $f_{(n,m)}^{*t}$ whose differential backlog achieves the maximum in (2).

2) Traffic injection at the sources:
In the case of inelastic traffic, the flow rate λ_f is given to each flow f. At time slot t, the source of flow f will generate traffic to inject into the network according to its given rate. We assume that the arrival processes of the flows are independent of each other, independent from time slot to time slot, and have finite second moments.

In the case of elastic traffic, we assume that each flow f runs the following well-known congestion control algorithm [2]–[7]. (To be precise, the congestion control algorithm that follows from [3]–[6] is somewhat different from the above one, but all results of this paper are valid for such congestion control as well.) At time slot t, the source of flow f computes the rate at which it injects packets into the ingress queue as follows:

$$x_f[t] = \min\left\{ U_f^{'-1}\left(\frac{Q_{b(f)}^f[t]}{M}\right), x_{max}\right\},$$

where x_{max} is an upper-bound of the arrival rates, and M is a positive parameter. We again assume that, conditioned on the rate vector calculated by the above congestion control algorithm, the arrival processes of the flows are independent of each other, independent from time slot to time slot, and have finite second moments.

It has been shown in [1] that, for inelastic traffic, the traditional back-pressure algorithm is *throughput-optimal*. Furthermore, for elastic traffic, the authors in [2]–[7] have shown that this algorithm, jointly with the above congestion control algorithm, can solve the optimal resource allocation problem (1). However, the traditional back-pressure algorithm has several major drawbacks. Its first drawback is the fact that it requires per-flow (per-destination) queues. This fact affects the scalability of the algorithm, since in large communication networks, the number of traffic flows (or traffic destinations) are usually much larger than the typical number of neighbors of a node. Moreover, the delay performance of the traditional back-pressure algorithm can be quite poor, as it is investigated in the next subsection.

B. Delay Performances

In this subsection, we formally characterize the delay performance of back-pressure algorithm using number of hops as the metric.

For inelastic traffic, the following theorem establishes an upper-bound on the end-to-end queue backlog for any flow.

Theorem 1: Consider a general topology network accessed by a set of flows with fixed routes. Let K_{max} be the maximum number of hops in the route of any flow, i.e., $K_{max} = \max_f |L(f)|$. Suppose the arrival rate vector λ is such that, for some $\epsilon > 0$, $(1+\epsilon)\lambda$ lies in the interior of the capacity region of the network. Then, under the back-pressure scheduling algorithm, the expected value of the sum of queue lengths (in steady-state) along the route of any flow f is bounded as follows:

$$\mathbb{E}\left[\sum_{n\in R(f)} Q_n^f[\infty]\right] \ \le\ \left(1+\frac{1}{\epsilon}\right)\frac{b}{\lambda_f}|\mathcal{F}|K_{max}^2\ ,\ \forall f\in\mathcal{F},$$

where constant $b > 0$ depends only on c_{max}.

Proof: The proof is presented in Appendix A. ■

For elastic traffic, it has been proven (e.g., in [3]–[7]) that the above joint congestion control (by the sources) and back-pressure (by the network) algorithm asymptotically achieves the optimal solution for the resource allocation problem (1). In particular, under this joint congestion control and back-pressure algorithm, the long term average flow rates would get close to the optimal flow rates, and the long term average queue lengths (scaled by M) get close to the corresponding Lagrange multipliers (see [3], [4]). Thus, one can also expect

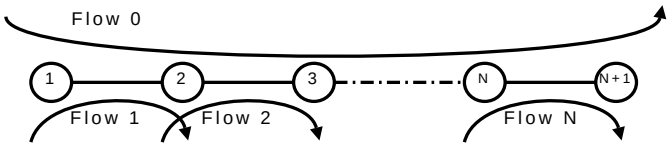

Fig. 1. The linear network with N links.

a similar result as Theorem 1 for elastic traffic, i.e., the total end-to-end queue backlog for any flow is upper-bounded by a quadratic function of the number of hops.

While the above results are only upper-bounds, the quadratic growth of total end-to-end queue backlog is turned out to be the exact bound in some particular linear-topology networks for both inelastic and elastic traffic, as shown in the next theorem.

Theorem 2: Consider a linear network with N links (indexed 1 to N) and $N+1$ flows (indexed 0 to N) as in Figure 1. Flow 0 goes through all N links, and each of other N flows goes through each link. Each link has a unit capacity, and there is no interference between them. Then we have that:

1) For inelastic traffic, let λ_i denote the arrival rate of flow i ($i = 0, \ldots, N$). Under the back-pressure algorithm, if $\lambda_0 > 1/2$, then the expected end-to-end queue backlog of flow 0 (in steady state) grows at least quadratically in N.
2) For elastic traffic with deterministic arrival rates, under the joint back-pressure and congestion control algorithm, the expected end-to-end queue backlog of flow 0 (in steady state) grows at least quadratically in N.

Proof: The proof is presented in Appendix B. ■

IV. A Scheme Based on the Shadow Algorithm

A. Motivation

One great advantage of the back-pressure algorithm is that it can perform resource allocation *adaptively*. However, as we have seen in the previous section, the total end-to-end queue backlog of any flow under back-pressure algorithm is upper-bounded by a quadratic function of the number of hops, and this bound is tight for some linear network configurations. In particular, let us consider a wireline linear network with N links having the same capacity and only one flow going through all these links. Then we can show, as a side result from the proof of Theorem 2, that the flow will have a quadratic end-to-end queue backlog under the back-pressure scheduling algorithm, as long as its rate is large enough but less than the capacity. On the other hand, if a fixed service rate (larger than the flow's arrival rate) is allocated to the flow on each link, then its total end-to-end queue length is expected to grow only linearly in the number of hops. But can such an allocation be done *adaptively*? The main point of this work is to use a fictitious queueing system called the *shadow queueing* system to perform such an allocation in the network adaptively, while using only a single physical FIFO queue for each outgoing link (also known as per-neighbor queueing) at each node. Therefore, it improves the delay performance of the traditional back-pressure algorithm, and at the same time reduces its implementation complexity.

We note that the shadow queue concept was introduced in [10], but the main goal there was to extend the network utility maximization framework for wireless networks to include multicast flows. On the other hand, in this work, we show that shadow queues can be useful even in networks with unicast flows only for the purpose of delay reduction. Furthermore, the idea of using per-neighbor queueing and establishing its stability is another important contribution here.

B. Description

The traditional back-pressure algorithm requires the queue length of every flow that passes through a node to perform resource allocation. The idea of the shadow algorithm is to decouple the storage of this information from the queueing data structure required to store packets at each node. The details of the shadow algorithm are described as follows.

Queues and Counters: At each node, instead of keeping a separate queue for each flow as in the back-pressure algorithm, a FIFO (first-come first-served) queue is maintained for each outgoing link. This FIFO queue stores packets for all flows going through the corresponding link. When a node receives a packet, it looks at the packet's header: if the node is not the final destination of that packet, it will send the packet to the FIFO queue of the next-hop link; otherwise, it will deliver the packet to the upper layer. We let $P_{nm}[t]$ denote the length of the queue maintained at link (n, m) and at the beginning of time slot t.

Each node maintains a separate *shadow* queue (i.e., a counter) for each flow going through it. Let $\tilde{Q}_n^f[t]$ be the length of the shadow queue (i.e., the value of the counter) of flow f at node n at the beginning of time slot t. The shadow queues and real queues are updated according to the scheduling algorithm described next. Note that each node still needs to keep a separate shadow queue for every flow going through it, but these are just counters, not actual physical queues. A counter is much easier to implement than a physical queue.

Back-pressure scheduling using the shadow queue lengths: At time slot t,

- Each link looks at the maximum *shadow* differential backlog of all flows going through that link:

$$w_{nm}[t] = \max_{f:(n,m)\in L(f)} \left(\tilde{Q}_n^f[t] - \tilde{Q}_m^f[t]\right). \qquad (4)$$

- Back-pressure scheduling:

$$\pi^*[t] = \max_{\pi\in\Gamma} \sum_{(n,m)} c_{nm}^{\pi} w_{nm}[t]. \qquad (5)$$

- A schedule $\pi^* = (c_1^{\pi}, c_2^{\pi}, \ldots, c_{|\mathcal{L}|}^{\pi})$ is interpreted by the network as follows: link (n, m) transmits c_{nm}^{π} shadow packets from the shadow queue of the flow whose differential backlog achieves the maximum in (4) (if the shadow queue has fewer than c_{nm}^{π} packets, then it is emptied); link (n, m) also transmits as many real packets

as shadow packets from its real FIFO queue. Again, if the number of real packets in the queue is less than the number of transmitted shadow packets, then all the real packets are transmitted.

We recall that shadow queues are just counters. The action of "transmitting shadow packets" is simply the action of updating the counters' values. In other words, "transmitting" k shadow packets from $\tilde{Q}_n^f$ to $\tilde{Q}_m^f$ means that we subtract k from $\tilde{Q}_n^f$ and add k to $\tilde{Q}_m^f$. From the above description, it should be clear that the shadow packets can be interpreted as permits which allow a link to transmit. Unlike the traditional back-pressure algorithm, the permits are associated with just a link rather than with a link and a flow.

Injection of elastic traffic (congestion control): At time slot t, the source of flow f computes the rate at which it injects packets into the ingress *shadow* queue as follows:

$$x_f[t] = \min\left\{ U_f'^{-1}\left(\frac{\tilde{Q}_{b(f)}^f[t]}{M}\right), x_{max}\right\}, \qquad (6)$$

where x_{max} is an upper-bound of the arrival rates, and M is a positive parameter. The source also generates real traffic at rate $\beta x_f[t]$ where β is a positive number less than 1.

Let $a_f[t]$ and $\tilde{a}_f[t]$ be the number of real and shadow packets generated and injected to the network at time t, respectively. For simplicity, we assume that $a_f[t]$ and $\tilde{a}_f[t]$ are Poisson random variables with means βx_f and x_f, independent of each other across flows and from time slot to time slot. Since the shadow packets are permits that allow real-packet transmission, from basic queueing theory, it follows that the actual packet arrival rate must be slightly smaller than the shadow packet arrival rate to ensure the stability of real queues. The parameter β is chosen to be less than 1 for this purpose. As we will see later in simulations, the queue backlog in the network would be smaller for smaller values of β.

Injection of inelastic traffic: For inelastic traffic, the same shadow algorithm can be used without congestion control. To ensure stability of the real queues, if the real arrival rate of an inelastic flow is λ_f, the shadow arrival rate for this flow must be larger than λ_f. For example, if we wish to make the shadow arrival rate larger than the real arrival rate by a factor of $(1+\epsilon)$, it can be accomplished as follows: for every real packet arrival, generate a shadow packet. Generate an additional shadow packet for each real packet with probability ϵ. This procedure ensures that the shadow arrival rate will be $(1+\epsilon)$ times the real arrival rate. For the algorithm to be stable, the set of arrival rates $\{\lambda_f(1+\epsilon)\}_f$ must lie in the interior of capacity region.

We note that the concept of shadow queues here is different from the notion of virtual queues used in [14] for the Internet and in [5] for wireless networks. In networks with virtual queueing systems, the arrival rates to both the real and virtual queues are the same, but the virtual queue is drained at a slower rate than the real queue. Instead, here the arrival rates to the real queues are slightly smaller than the arrival rates to the corresponding shadow queues. This subtle difference is important in that it allows us to use per-neighbor FIFO queues and prove stability in a multihop wireless network in the next section.

V. Stability of the queues (shadow and real) under the shadow-algorithm based scheme

A. Elastic traffic

In this subsection, we will establish (asymptotic) optimality of the shadow-algorithm based scheme for elastic flows, and its joint stability of shadow and real queues. First, we have the following theorem on the resource allocation's optimality and the *shadow queues'* stability. Its proof follows from [3], [7] (we remind again that the congestion control that follows from the algorithm in [3] is somewhat different); a related result is also proven in [6].

Theorem 3: The congestion control and scheduling algorithms, controlling shadow queues, as described in Section IV above, asymptotically achieve the optimal rate allocation for shadow traffic, i.e.,

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[x[t]] = x^* + o(1), \quad \text{as } M\to\infty, \qquad (7)$$

where x^* is the optimal solution to (1). Furthermore, the *shadow* queues are stable in the sense that the Markov chain of shadow queues $\tilde{Q}[t]$ is positive recurrent and the steady-state expected values of the shadow queue lengths are bounded as follows:

$$\sum_{n,f}\mathbb{E}(\tilde{Q}_n^f[\infty]) = O(M).$$

The remaining goal is to prove the stability of the real queues. Note that the sources are sending real traffic with smaller rates than shadow traffic, and we know that the shadow queues are stable. However, it does not automatically mean that the real queues are stable as well, since each of them is an aggregated FIFO queue storing packets for all flows going through its corresponding link. To the best of our knowledge, there is no sample-path relationship between the shadow and real queueing systems which would allow us to conclude stability. Fortunately, we can apply results from the stochastic networks literature to establish the following result.

Theorem 4 (Elastic traffic): The process describing the joint evolution of shadow and real queues, is an irreducible, aperiodic, positive recurrent Markov chain.

Remark. Note that the complete state of the Markov chain referred to in Theorem 4, is *not* simply

$$\left(\left(\tilde{Q}_n^f[t]\right)_{f\in\mathcal{F},n\in\mathcal{N}}; (P_{nm}[t])_{(n,m)\in\mathcal{L}}\right),$$

because it also includes the order in which packets of different types are placed in each link (n,m) FIFO queue.

The proof is based on the fluid limit approach and a result by Bramson [13]. In his paper, Bramson proved that fluid models of Kelly-type FIFO queueing networks are stable as long as the nominal load on each server is strictly less than its capacity. Thus, the basic idea of the proof is as follows. The random process describing the behavior of *shadow* queues,

under the joint congestion control and scheduling algorithm (running on the shadow system), is positive recurrent (as specified in Theorem 3). Therefore, the *average* service rate on each network *link* that the shadow algorithm yields is strictly greater than the nominal load of the link due to the thinning of actual traffic; moreover, the (random) cumulative amount of service provided on each link up to time t satisfies the functional strong law of large numbers, as t goes to infinity. As a result, if we take the *fluid limit* of the process describing real FIFO queues, it has *exactly same form as if each network link would have constant, non-time-varying capacity (equal to the average rate provided by the shadow algorithm)*. Then, this fluid limit is stable by the results of [13], which implies stability of the process of real queues. The proof's details are presented in Appendix C just for the purpose of completeness.

Note that the real traffic throughput will always be slightly smaller than the optimal solution to (1), but this difference from the optimal solution can be made arbitrarily small by adjusting the parameter β.

B. Inelastic traffic

In the case of inelastic traffic, we first note that the stability result of shadow queueing system (Theorem 3) is automatically extended, since the back-pressure algorithm is still run on shadow queues. Next, it can be show that an identical version of Theorem 4 also holds for inelastic traffic.

Theorem 5 (Inelastic traffic): The process describing the joint evolution of shadow and real queues, is an irreducible, aperiodic, positive recurrent Markov chain.

Proof: This proof can be easily extracted from the proof of Theorem 4 (presented in Appendix C) as a side-result, and hence, is eliminated. ■

VI. Simulation

In this section, we compare and contrast the performances of the traditional back-pressure algorithm and the shadow algorithm for networks with fixed routing.

A. Simulation results for inelastic traffic

To illustrate the queue length behavior under back-pressure algorithm in the case of inelastic traffic, we simulate the linear network in Figure 1. We choose $N = 80$, i.e., the network has 81 nodes and 80 links, with node-exclusive interference between links. Each link has capacity 10, i.e., it can transmit up to 10 packets per time slot. Let λ_0 be the fixed rate of flow 0, and λ_1 be the fixed rate of flows $1, 2, \ldots, 80$. We know that the back-pressure algorithm will stabilize the network as long as $2\lambda_0 + 2\lambda_1 < 10$. We let the sources send shadow traffic at fixed rates λ_i, and send real traffic at a slightly smaller rate $\beta\lambda_i$, with $\beta \in (0, 1)$.

Figure 2 shows the mean queue lengths of all queues maintained at each node when $\lambda_0 = 2.0$ and $\lambda_1 = 2.6$. The value of β here is 0.99. We see that the shadow queue lengths of flow 0 increase linearly when going from the end node to the begin node, which leads to a quadratic growth (in terms of the number of hops) of the end-to-end queue backlog. Moreover, we also see that the real FIFO queue lengths are significantly reduced, even with a small amount thinning of traffic (1%).

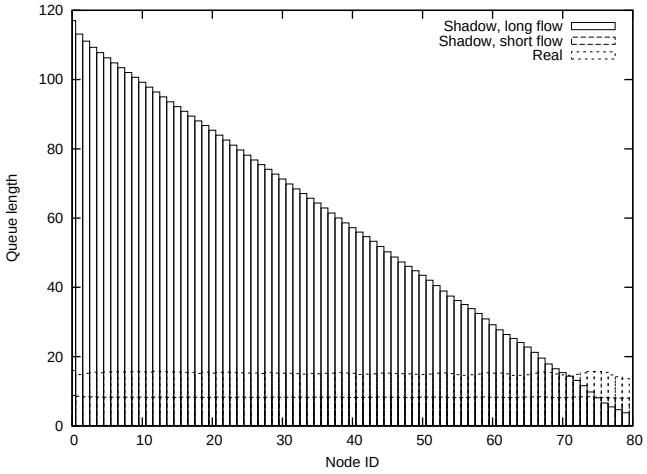

Fig. 2. The queue lengths at each node in the linear network in Figure 1. The solid-line boxes are the lengths of shadow queues of flow 0 (the long flow) maintained at each node. The dash-line boxes are the shadow queue lengths of flows i, $i = 1, \ldots, 80$, (the short flows) at node i, respectively. Finally, the dot-line boxes are the real FIFO queue lengths at each node.

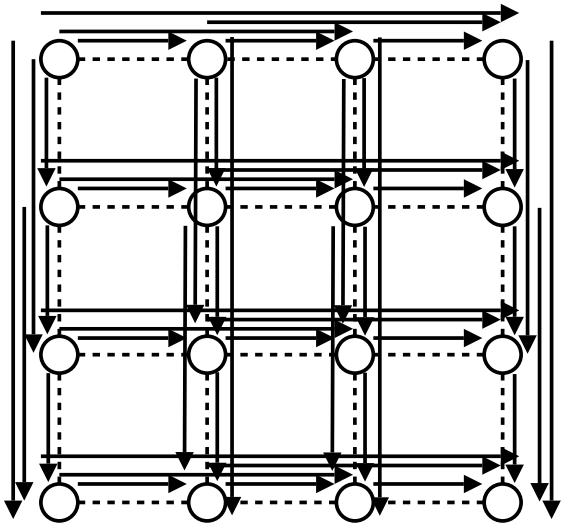

Fig. 3. A grid network with 16 nodes, 24 links, and 48 flows. Links and flows are represented by dash lines and solid arrows, respectively.

B. Simulation results for elastic traffic

In this subsection, we investigate the performance of the shadow algorithm with elastic traffic in a network with a more complicated topology than a line. In particular, we consider a grid network as shown in Figure 3. We assume that all flows have a logarithmic utility function, i.e., $U_f(x_f) = \log x_f$ for all f. The network has 16 nodes (represented by circles) and 24 links (represented by dash lines). We again assume the node-exclusive interference model under which a matching in the graph represents a valid schedule. Each link has an unit capacity, i.e., it can transmit one packet per time slot if scheduled. There are 48 flows (represented by arrows) sharing this network.

We implement the shadow algorithm as described in Section IV with the parameter $M = 20$. In Figure 4, we plot the evolution of total shadow queue length and total real

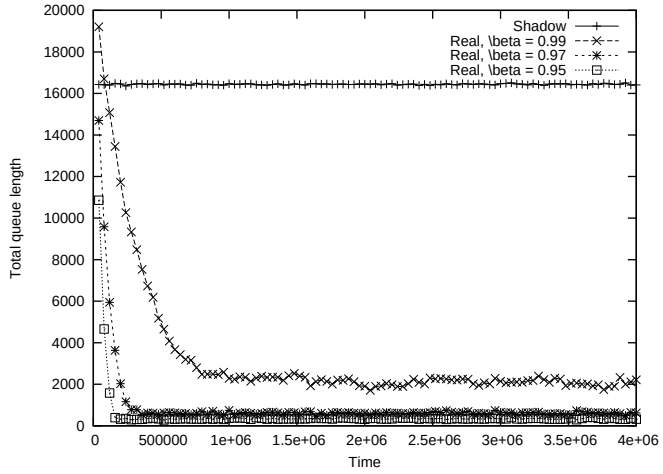

Fig. 4. The evolutions of total shadow queue length and total real queue lengths over time with $M = 20$ and with different values of β for the network in Figure 3.

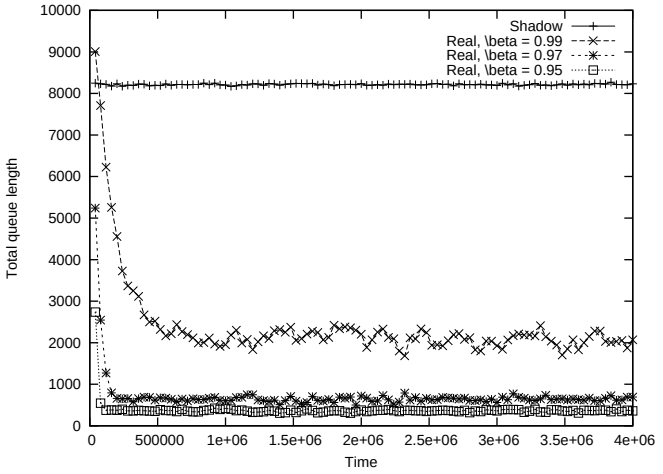

Fig. 5. The evolutions of total shadow queue length and total real queue lengths over time with $M = 10$ and with different values of β for the network in Figure 3.

queue length for several values of parameter β (the total queue length is the sum of all queue lengths in the network). Note that the shadow queue length is also the queue length of the traditional back-pressure scheduling without the shadow algorithm. The figure indicates that the total real queue length with the shadow algorithm decreases dramatically compared to the traditional back-pressure algorithm (although it takes longer time to converge to its stationary value, which can be considered as a disadvantage of the shadow algorithm). Thus, significant gains in performance can be realized at the expense of a small loss in throughput (represented by the parameter $1 - \beta$). Note that the traditional back-pressure algorithm can perform poorly due to many reasons: (i) As in Section III-B, if the number of hops for a flow is large, then the queue backlog can increase quadratically. (ii) The choice of the parameter M in the congestion control algorithm (Equation (6)) can lead to queue backlogs of the order of M (see the upper bound in Theorem 3 and another simulation result running in the same network with $M = 10$ in Figure 5; more results on this relation can be found in [5]). (iii) A separate queue is maintained for each destination. The shadow algorithm solves all of these problems at once by "reserving" capacity between each source-destination pair, i.e., for each flow.

It might be interesting to know how these total queue lengths spread over nodes/flows/links in the network. Particularly, we are interested in the per-node queue lengths in this case, since for the node-exclusive interference model they will give a fair comparison between the traditional back-pressure algorithm and the shadow algorithm. Recall that the shadow queue length is also the queue length of the traditional back-pressure scheduling without the shadow algorithm. Then we define the total shadow queue length at a node as the sum of all shadow queues maintained for all flows going through that node. Similarly, the total real queue length at a node is the sum of all real queues maintained for all neighbors of that node. Table I gives the values of per-node total queue lengths at *steady state* for the grid network in Figure 3 with $M = 20$ and different values of β. (The nodes are indexed from left to right, and then from top to bottom, starting with index 0 for the node in the top-left corner, and ending with index 15 for the node at the bottom-right corner.) We can see that as β decreases, the real queue length decreases *uniformly* at every node, not just some particular nodes.

Node Index	Shadow	Real, $\beta = 0.99$	Real, $\beta = 0.97$	Real, $\beta = 0.95$
0	1233.56	224.89	55.85	33.07
1	1762.18	349.17	83.87	46.05
2	1223.52	236.47	67.09	38.70
3	616.27	114.90	29.56	15.25
4	1763.27	336.61	85.24	51.07
5	2017.84	455.68	142.79	71.04
6	1497.36	437.15	124.44	55.93
7	770.54	131.43	35.08	20.89
8	1222.21	216.61	68.44	40.66
9	1496.89	319.48	121.09	63.02
10	975.49	287.41	90.49	54.27
11	231.71	99.34	24.72	14.90
12	617.39	85.27	27.31	16.51
13	771.28	142.24	41.53	24.41
14	232.14	82.55	29.18	14.10
15	0	0	0	0

TABLE I
PER-NODE QUEUE LENGTHS WITH $M = 20$ AND DIFFERENT VALUES OF β FOR THE NETWORK IN FIGURE 3.

We now turn our attention to the trade-off between throughput optimality and delay performance of the shadow algorithm. In particular, we are interested in how the total network utility of *real traffic* and the total *real queue length* at steady state vary with different values of β and M. For each pair of (β, M), let $Q(\beta, M)$ and $U(\beta, M)$ denote the total real queue length at steady state and the total achieved network

	$r(\beta, M)$	$Q(\beta, M)$
$\beta = 0.95,\ M = 10$	2.2089%	413.2
$\beta = 0.97,\ M = 10$	1.2960%	744.4
$\beta = 0.99,\ M = 10$	0.4273%	2477.5
$\beta = 0.95,\ M = 20$	2.1886%	579.1
$\beta = 0.97,\ M = 20$	1.2848%	1046.3
$\beta = 0.99,\ M = 20$	0.3912%	3494.9
$\beta = 0.95,\ M = 50$	2.1127%	1809.2
$\beta = 0.97,\ M = 50$	1.2076%	3267.8
$\beta = 0.99,\ M = 50$	0.3314%	10180.1

TABLE II
THE VALUES OF $r(\beta, M)$ AND $Q(\beta, M)$ WITH DIFFERENT VALUES OF β AND M FOR THE NETWORK IN FIGURE 3.

utility of real traffic at steady state, respectively. Also, let U^* denote the theoretical maximum network utility. Then the ratio $r(\beta, M) := \frac{U^* - U(\beta,M)}{U^*}$ can be used to evaluate the throughput optimality for real traffic for each (β, M).

Table II shows the values of $Q(\beta, M)$ and the percentage of $r(\beta, M)$ for the same grid network in Figure 3 with different values of β and M. Intuitively, as β or M increases, $r(\beta, M)$ should become smaller (i.e., the total network utility is closer to its optimal value) and $Q(\beta, M)$ should respectively become large. The results from Table II confirm this fact.

VII. CONCLUSIONS

In this paper, we have proposed a new shadow architecture to improve the delay performance of back-pressure scheduling algorithm. The shadow queueing system allows each node to maintain a single FIFO queues for each of its outgoing links, instead of keeping a separate queue for each flow in the network. This architecture not only reduces the queue backlog (or, equivalently, delay by Little's law) but also reduces the number of actual physical queues that each node has to maintain.

We presented the shadow algorithm for the case of fixed routing, i.e., the route for each flow is fixed. The shadow algorithm can also be used in the case of adaptive routing, but a node cannot use just one FIFO queue for each neighbor. If one still maintains a separate queue for each destination at each node, then the extension of the shadow algorithm to the case of adaptive routing is straightforward. On the other hand, it would be interesting to study if a single per-neighbor FIFO queue can be maintained even in the case of adaptive routing. This is an interesting topic for future research.

APPENDIX A
PROOF OF THEOREM 1

Recall that $L(f)$ is the set of links forming the route of flow f. Now, we let $R(f)$ denote the set of nodes forming the route of f (and hence, $|R(f)| = |L(f)| + 1$). For each pair (f, n) such that $n \in R(f)$, we abuse the notation by letting

- $n+1$ denote the next node of n in the route of f ($n \neq e(f)$);
- $n-1$ denote the previous node of n in the route of f ($n \neq b(f)$).

For each $n \in R(f)$, let us define

$$\pi_{out(f,n)}[t] \ := \ \pi^f_{(n,n+1)}[t] \ , \quad n \neq e(f),$$

$$\pi_{in(f,n)}[t] := \begin{cases} a_f[t], & n = b(f), \\ \min\left\{\pi^f_{(n-1,n)}[t], Q^f_{n-1}[t]\right\}, & n \neq b(f), \end{cases}$$

where $a_f[t]$ is the number of external arrivals of flow f at time t. The queue dynamics are then given by

$$Q^f_n[t+1] = \left(Q^f_n[t] - \pi_{out(f,n)}[t]\right)^+ + \pi_{in(f,n)}[t]. \quad (8)$$

Now, consider the Lyapunov function

$$V(Q) \ = \ \frac{1}{2}\sum_{f\in\mathcal{F}}\sum_{n\in R(f)} (Q^f_n)^2.$$

We can rewrite the queues' dynamics (8) as follows:

$$Q^f_n[t+1] = Q^f_n[t] - \pi_{out(f,n)}[t] + \pi_{in(f,n)}[t] + u^f_n[t],$$

where

$$u^f_n[t] = \begin{cases} 0 & \text{if } Q^f_n[t] \geq \pi_{out(f,n)}[t], \\ -Q^f_n[t] + \pi_{out(f,n)}[t] & \text{if } Q^f_n[t] < \pi_{out(f,n)}[t]. \end{cases}$$

The drift of the Lyapunov function is given by

$$\begin{aligned} \Delta V[t] \ &:= \ \mathbb{E}\left[V(Q[t+1]) - V(Q[t])\middle| Q[t]\right] \\ &= \ \frac{1}{2}\sum_{f\in\mathcal{F}}\sum_{n\in R(f)} \mathbb{E}\left[2Q^f_n[t]\left(\pi_{in(f,n)}[t] - \pi_{out(f,n)}[t]\right)\right. \\ &\quad + \left(\pi_{in(f,n)}[t] - \pi_{out(f,n)}[t]\right)^2 \\ &\quad + 2u^f_n[t]\pi_{in(f,n)}[t] + \left(u^f_n[t]\right)^2 \\ &\quad \left. + 2u^f_n[t]\ \left(Q^f_n[t] - \pi_{out(f,n)}[t]\right)\middle| Q[t]\right]. \end{aligned}$$

Recall that $\pi_{in(f,n)}[t] = \pi_{out(f,n-1)}[t] - u^f_{n-1}[t],\ n \neq b(f)$. Thus, we get

$$\begin{aligned} \Delta V[t] \ &= \ B_1[t] + \sum_{f\in\mathcal{F}} Q^f_{b(f)}[t]\lambda_f \\ &\quad - \sum_{f\in\mathcal{F}}\sum_{(n,m)\in L(f)} \left(Q^f_n[t] - Q^f_m[t]\right)\mathbb{E}\left[\pi^f_{nm}[t]\middle| Q[t]\right] \\ &= \ B_1[t] + \sum_{f\in\mathcal{F}} Q^f_{b(f)}[t]\lambda_f \\ &\quad - \sum_{(n,m)\in\mathcal{L}} \pi^*_{nm}[t] \max_{f:(n,m)\in L(f)} \left(Q^f_n[t] - Q^f_m[t]\right)^+, \end{aligned}$$

where the last equality is due to the back-pressure scheduling algorithm, and

$$\begin{aligned} B_1[t] \ &= \ \frac{1}{2}\sum_{f\in\mathcal{F}}\sum_{n\in R(f)} \mathbb{E}\left[\left(\pi_{in(f,n)}[t] - \pi_{out(f,n)}[t]\right)^2\right. \\ &\quad + \left(u^f_n[t]\right)^2 - 2u^f_{n-1}[t]Q^f_n[t] \\ &\quad \left. + \ 2u^f_n[t]\left(Q^f_n[t] + \pi_{in(f,n)}[t] - \pi_{out(f,n)}[t]\right)\middle| Q[t]\right]. \end{aligned}$$

Since λ is strictly inside the region Λ, there exist a positive constant ϵ and a vector of link rates μ such that

$$\mu_{nm} \ \geq \ (1+\epsilon) \sum_{f:(n,m)\in L(f)} \lambda_f \ , \quad \text{and} \quad \mu \ \in \ co(\Gamma).$$

And hence,

$$\sum_{f\in\mathcal{F}} Q^f_{b(f)}[t]\lambda_f \;=\; \sum_{f\in\mathcal{F}}\sum_{(n,m)\in L(f)} \lambda_f\left(Q^f_n[t]-Q^f_m[t]\right)$$
$$\leq\; \frac{1}{1+\epsilon}\sum_{(n,m)\in\mathcal{L}} \mu_{nm}\max_{f:(n,m)\in L(f)}\left(Q^f_n[t]-Q^f_m[t]\right)^+.$$

Therefore,

$$\begin{aligned}\Delta V[t] \;\leq\; & B_1[t]\\ & -\sum_{(n,m)\in\mathcal{L}}(\pi^*_{nm}[t]-\mu_{nm})\max_{f:(n,m)\in L(f)}\left(Q^f_n[t]-Q^f_m[t]\right)^+\\ & -\frac{\epsilon}{1+\epsilon}\sum_{(n,m)\in\mathcal{L}}\mu_{nm}\max_{f:(n,m)\in L(f)}\left(Q^f_n[t]-Q^f_m[t]\right)^+.\end{aligned}$$

Now, for any flow $f\in\mathcal{F}$, we have that

$$\begin{aligned}\sum_{n\in R(f)} Q^f_n[t] \;\leq\; & |R(f)|\sum_{(n,m)\in L(f)}\left(Q^f_n[t]-Q^f_m[t]\right)^+\\ \leq\; & |R(f)|\sum_{(n,m)\in L(f)}\max_{g:(n,m)\in L(g)}\left(Q^g_n[t]-Q^g_m[t]\right)^+\\ \leq\; & \frac{K_{max}}{\mu_{L(f)}}\sum_{(n,m)\in\mathcal{L}}\mu_{nm}\max_{g:(n,m)\in L(g)}\left(Q^g_n[t]-Q^g_m[t]\right)^+,\end{aligned}$$

where $\mu_{L(f)}>0$ is the minimum link rate μ_{nm} of any link which is part of the flow's route; obviously, $\mu_{L(f)}\geq\lambda_f$. Thus, for any flow $f\in\mathcal{F}$,

$$\Delta V[t] \;\leq\; B_1[t]-\frac{\epsilon}{1+\epsilon}\frac{\lambda_f}{K_{max}}\sum_{n\in R(f)} Q^f_n[t]. \qquad (9)$$

Note that $B_1[t]\leq b|\mathcal{F}|K_{max}$, $\forall t$, for some constant $b>0$ which depends only on c_{max} (see model definition). Thus, from (9), we have that, $\forall f\in\mathcal{F}$,

$$\begin{aligned}\Delta V[0] &\leq b|\mathcal{F}|K_{max}-\frac{\epsilon}{1+\epsilon}\frac{\lambda_f}{K_{max}}\sum_{n\in R(f)}Q^f_n[0],\\ \Delta V[1] &\leq b|\mathcal{F}|K_{max}-\frac{\epsilon}{1+\epsilon}\frac{\lambda_f}{K_{max}}\sum_{n\in R(f)}Q^f_n[1],\\ &\cdots\\ \Delta V[T-1] &\leq b|\mathcal{F}|K_{max}-\frac{\epsilon}{1+\epsilon}\frac{\lambda_f}{K_{max}}\sum_{n\in R(f)}Q^f_n[T-1].\end{aligned}$$

Taking expectation on both sides of those inequalities, summing and rearranging the terms yield

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{n\in R(f)}Q^f_n[t]\right]\\ &\leq\; \frac{1+\epsilon}{\epsilon}\frac{b|\mathcal{F}|K^2_{max}}{\lambda_f}+\frac{1+\epsilon}{\epsilon}\frac{K_{max}}{T\lambda_f}\left(V[0]-\mathbb{E}\left[V[T]\right]\right)\\ &\leq\; \frac{1+\epsilon}{\epsilon}\frac{b|\mathcal{F}|K^2_{max}}{\lambda_f}+\frac{1+\epsilon}{\epsilon}\frac{K_{max}V[0]}{T\lambda_f},\end{aligned}$$

where the last inequality is due to the fact that $V(\cdot)$ is non-negative. Since $V[0]$ is finite, we then obtain that, $\forall f\in\mathcal{F}$,

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{n\in R(f)}Q^f_n[t]\right] \;\leq\; \frac{1+\epsilon}{\epsilon}\frac{b}{\lambda_f}|\mathcal{F}|K^2_{max}.$$

The above bound along with the positive recurrence of $Q[t]$ gives the desired result.

Appendix B
Proof of Theorem 2

1) For inelastic traffic:

This result for inelastic traffic has been shown in [15] (Proposition 2), for a continuous-time model. The proof in our discrete-time setting is essentially same – we present it here for completeness.

Let $Q^0_j[t]$ denote the length of the queue maintained at node j for flow 0 at the beginning of time slot t. Since there is no interference between links, under the back-pressure algorithm, if a packet of flow 0 is transmitted over link j during time slot t, then necessarily $Q^0_j[t]>Q^0_{j+1}[t]$.

Suppose that $Q^0_j[k]\geq Q^0_{j+1}[k]-1$ at time k. Then we have that $Q^0_j[t]\geq Q^0_{j+1}[t]-1$ for all $t\geq k$. This is easily seen by induction on time: if the condition holds at t, then under back-pressure rule it must hold at $t+1$ as well. The Markov process is recurrent, in particular it reaches "empty" state (with all queues being zero) with probability 1. We conlude that in the stationary regime, $Q^0_j[\infty]\geq Q^0_{j+1}[\infty]-1$ holds for all j.

Next, note that in steady state,

$$\mathbb{P}\left(Q^0_j[\infty]>Q^0_{j+1}[\infty]\right)\geq\lambda_0, \qquad j=1,2,\ldots,N,$$

because the average rate of flow 0 from j to $j+1$ is exactly λ_0, and the condition $Q^0_j>Q^0_{j+1}$ is necessary for a packet to be passed from j to $j+1$. Therefore,

$$\begin{aligned}&\mathbb{E}\left[Q^0_j[\infty]-Q^0_{j+1}[\infty]\right]\\ &\geq\; 1\cdot\mathbb{P}\left(Q^0_j[\infty]>Q^0_{j+1}[\infty]\right)\\ &\quad+(-1)\cdot\mathbb{P}\left(Q^0_j[\infty]\leq Q^0_{j+1}[\infty]\right)\\ &=\; 2\mathbb{P}\left(Q^0_j[\infty]>Q^0_{j+1}[\infty]\right)-1\\ &\geq\; 2\lambda_0-1.\end{aligned}$$

Note that by convention, $Q^0_{N+1}[k]=0,\ \forall k\geq 0$. Therefore,

$$\begin{aligned}\mathbb{E}\left[Q^0_N[\infty]\right] &\geq (2\lambda_0-1)\\ \mathbb{E}\left[Q^0_{N-1}[\infty]\right] &\geq 2(2\lambda_0-1)\\ &\vdots\\ \mathbb{E}\left[Q^0_1[\infty]\right] &\geq N(2\lambda_0-1),\end{aligned}$$

and

$$\sum_{i=1}^{N}\mathbb{E}\left[Q^0_i[\infty]\right]\geq\frac{N(N+1)}{2}(2\lambda_0-1),$$

which is quadratic as long as $\lambda_0>1/2$.

2) For elastic traffic:

Let x_i and $U_i(\cdot)$ denote the rate and the utility function flow i, respectively. Then the network utility maximization problem

(1) for the linear network in Figure 1 becomes:

$$\begin{aligned} \max \quad & \sum_{i=0}^{N} U_i(x_i) \\ \text{s.t.} \quad & x_0 \ \leq \ \mu_{0,1}, \\ & \mu_{0,i} \ \leq \ \mu_{0,i+1}, \ i = 1, \ldots, N-1, \\ & x_i + \mu_{0,i} \ \leq \ c, \ i = 1, \ldots, N, \end{aligned}$$

where $\mu_{0,i}$ is the resource that link i allocates to serve flow 0.

If the utility is logarithmic (proportional fairness), i.e., $U_i(x) = \log(x)$, then one can easily compute the optimal rates and optimal queue lengths (which are the Lagrange multipliers) for the above optimization problem as follows:

$$\begin{aligned} & x_0^* = \mu_{0,1}^* = \ldots = \mu_{0,N}^* = \frac{c}{N+1}, \\ & x_1^* = \ldots = x_N^* = \frac{Nc}{N+1}, \\ & q_i^* = q_{0,i}^* - q_{0,i+1}^* = \frac{N+1}{Nc}, \ i = 1, \ldots, N, \end{aligned}$$

where q_i^* and $q_{0,i}^*$ are the optimal queue lengths maintained at node i for flow i and flow 0, respectively. Then, the end-to-end total queue backlog for flow 0 is

$$\sum_{i=1}^{N} q_{0,i}^* = \frac{N+1}{Nc} \sum_{i=1}^{N} i = \frac{(N+1)^2}{2c} = \Theta\left(N^2\right),$$

i.e., it grows quadratically in N.

For a more general class of utility functions which model a large class of fairness concepts [16],

$$U_i(x) = \frac{x^{1-\alpha}}{1-\alpha}, \quad \alpha > 0,$$

we still have similar results:

$$\begin{aligned} & x_0^* = \mu_{0,1}^* = \ldots = \mu_{0,N}^* = \Theta\left(N^{-1/\alpha}\right), \\ & x_1^* = \ldots = x_N^* = \Theta(1), \\ & q_i^* = q_{0,i}^* - q_{0,i+1}^* = \Theta(1), \ i = 1, \ldots, N, \end{aligned}$$

which again lead to $\sum_{i=1}^{N} q_{0,i}^* = \Theta\left(N^2\right)$.

Appendix C
Proof of Theorem 4

In this appendix, we provide details of the proof of Theorem 4. The proof uses the *fluid limit* technique [17]–[20], which is by now standard. For this reason, we focus on the details specific to our problem, while referring the reader to the above references for standard arguments.

A. Preliminaries

Recall the result from Theorem 3 that

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[x[t]] = x^*(\epsilon), \tag{10}$$

where $x^*(\epsilon)$ is within ϵ-boundary of the optimal solution x^* and ϵ can be made arbitrarily small by increasing M. To simplify the notations, from now on, we will drop ϵ in $x^*(\epsilon)$. In other words, we will use the notation x^* for the ϵ-approximate optimal solution.

From the above result, the following (strong law of large numbers) results can be established.

Lemma 1: Suppose a (non-random) initial state $\tilde{Q}[0]$ of shadow queues is fixed. For every flow $f \in \mathcal{F}$,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} a_f[t] \ = \ \beta x_f^* \qquad a.s., \tag{11}$$

i.e., the time average of real packet arrival rate converges to the optimal rates (for the elastic flows) scaled by β.
For every link $l \in \mathcal{L}$,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \pi_l^*[t] \ = \ \mu_l^* \qquad a.s. \tag{12}$$

for some μ^* such that $\sum_{f:l\in L(f)} x_f^* \leq \mu_l^*, \ \forall l \in \mathcal{L}$.

In other words, the average service rates "provided" by the shadow algorithm on all links are not smaller than the "nominal loads" of the link due to the optimal rates (for the elastic flows).

Proof: Let us prove (11). Consider any flow f. Let us consider the process $(\tilde{Q}[t], a_f[t]), t \geq 0$, where $\tilde{Q}[t]$ is the vector of all shadow queues at time t. This process is a countable irreducible ergodic Markov chain. (Ergodicity follows from that of the $\tilde{Q}[\cdot]$-component.) Pick any fixed state as a regeneration point, and denote by T_f and A_f the random duration of a regeneration cycle and the random number of flow f packets generated during a cycle, respectively. Since the process is ergodic, $\mathbb{E}[T_f] < \infty$. Therefore, by the key renewal theorem,

$$\frac{\mathbb{E}[A_f]}{\mathbb{E}[T_f]} = \lim_{t\to\infty} \mathbb{E}[a_f[t]] = \beta x_f^*,$$

which by the way implies $\mathbb{E}[A_f] < \infty$. Using the strong law of large numbers for the sequence of regeneration cycles, we obtain lemma statement when T increases along the sequence of regeneration points, which easily implies that it has to hold for just $T \to \infty$ as well. The proof of (12) is analogous. ■

To be consistent with [13], we introduce the concept of *packet class*. Each flow f consists of $|L(f)|$ packet classes; each class going through one link in the route of f. We let $\mathcal{S}$ denote the set of all packet classes. In other words, there is a bijection mapping a pair (f, l), $f \in \mathcal{F}$, $l \in L(f)$, to a packet class $s \in \mathcal{S}$. Clearly, $|\mathcal{S}| = \sum_{f\in\mathcal{F}} |L(f)|$.

For each flow $f \in \mathcal{F}$, let $\Phi(f)$ be the set of packet classes belonging to f. For each link $l \in \mathcal{L}$, let $C(l)$ be the set of packet classes going through l. Conversely, for each packet class $s \in \mathcal{S}$, let $f(s)$ be the corresponding flow (i.e., $s \in \Phi(f(s))$), and $l(s)$ be the corresponding link.

Let H denote the *constituency matrix* with size $|\mathcal{L}| \times |\mathcal{S}|$:

$$H_{l,s} \ = \ \begin{cases} 1 & \text{if } s \in C(l), \\ 0 & \text{otherwise.} \end{cases}$$

Also, let R be the *routing matrix* with size $|\mathcal{S}| \times |\mathcal{S}|$:

$$R_{s,u} \ = \ \begin{cases} 1 & \text{if } f(s) = f(u) \text{ and } u \text{ is the next hop} \\ & \qquad\qquad \text{of } s \text{ in the route of } f, \\ 0 & \text{otherwise.} \end{cases}$$

Next, let $E_s(t)$ denote the total number of *external* arrivals of packet class s up to time t. Thus,

$$E_s(t) = \begin{cases} \sum_{k=0}^{t-1} a_f[k] & \text{if } s \text{ is the first hop of } f(s), \\ 0 & \text{otherwise.} \end{cases}$$

Also, we define the arrival rates corresponding to packet classes:

$$\lambda_s = \begin{cases} \beta x_f^* & \text{if } s \text{ is the first hop of } f(s), \\ 0 & \text{otherwise.} \end{cases}$$

We then extend the definition of $E_s(t)$ to continuous time as follows: for each time $t \in \Re^+$, $E_s(t) := E_s(\lfloor t \rfloor)$. Hence, $E_s(t)$ is right continuous having left limits.

Recall that $\pi^*[t]$ is the outcome of the scheduling algorithm at time slot t. Now, for each $t \in \Re^+$, we let $M_l(t) := M_l(\lfloor t \rfloor) = \sum_{k=0}^{\lfloor t \rfloor - 1} \pi_l^*[k]$ denote the total amount of offered service (in terms of number of packets that *can be* transmitted) of link l up to time t.

Similarly, let us define $A_s(t) = A_s(\lfloor t \rfloor)$ as the total arrivals, and $D_s(t) = D_s(\lfloor t \rfloor)$ as the total departures, of packet class s up to time t. Thus,

$$A_s(t) = E_s(t) + \sum_{u \in \mathcal{S}} D_u(t) R_{u,s}. \tag{13}$$

Let $Q_s(t) = Q_s(\lfloor t \rfloor)$ be the number of packets of packet class s which are waiting to be served. Then,

$$Q_s(t) = Q_s(0) + A_s(t) - D_s(t). \tag{14}$$

Recall that $P_l(t) = P_l(\lfloor t \rfloor)$ is the length of FIFO queue, i.e., the current workload at link l at time t. Thus,

$$P_l(t) = \sum_{s \in C(l)} Q_s(t) = \sum_s H_{l,s} Q_s(t). \tag{15}$$

Now, we define $I_l(t)$ as the amount unused offered service, measured in number of packets, at link l during $[0, t]$. Then we have the following equations:

$$\sum_s H_{l,s} D_s(t) + I_l(t) = M_l(t), \tag{16}$$

and the fact that $I_l(t)$ can only increase when $P_l(t) < c_{max} \doteq \max_{\pi,l} c_l^\pi$, i.e., if $I_l(t_2) > I_l(t_1)$ then $P_l(t) < c_{max}$ for some $t \in [t_1, t_2]$.

We can rewrite the above equations (13)-(16) in vector form to get the following set of equations which describes the evolution of the system:

$$A(t) = E(t) + R^T D(t) \tag{17}$$
$$Q(t) = Q(0) + A(t) - D(t) \tag{18}$$
$$P(t) = HQ(t) \tag{19}$$
$$HD(t) + I(t) = M(t) \tag{20}$$
$$I_l(t) \text{ can only increase when } P_l(t) < c_{max},\ l \in \mathcal{L}. \tag{21}$$

Note that $E_s(t)$, $M_l(t)$, $A_s(t)$, $D_s(t)$, $Q_s(t)$, $P_l(t)$, and $I_l(t)$ are all right continuous having left limits. We also can and will use the convention that $A(0) = D(0) = I(0) = 0$.

Now, recall that the real queues are served in FIFO manner, which implies the following equation:

$$Q_s(0) + A_s(t) \in [D_s\left(t + \theta_{l(s)}(t)-\right), D_s\left(t + \theta_{l(s)}(t)\right)], \tag{22}$$

where $\theta_l(t)$ denotes the smallest number $y \geq 0$ such that $M_l(t+y) - M_l(t) \geq P_l(t)$. (In other words, $\theta_l(t)$ is an inverse of $M_l(t+\cdot) - M_l(t)$, taken at point $P_l(t)$.)

We now describe the *fluid model* of the system. (To be precise, this is a fluid model *assuming the initial state of shadow queues is fixed* – this will be enough for our purposes.) The set of fluid model equations is as follows:

$$\bar{A}(t) = \beta\lambda t + R^T \bar{D}(t) \tag{23}$$
$$\bar{Q}(t) = \bar{Q}(0) + \bar{A}(t) - \bar{D}(t) \tag{24}$$
$$\bar{P}(t) = H\bar{Q}(t) \tag{25}$$
$$H\bar{D}(t) + \bar{I}(t) = \mu^* t \tag{26}$$
$$\bar{I}_l(t) \text{ can only increase when } \bar{P}_l(t) = 0,\ l \in \mathcal{L} \tag{27}$$
$$\bar{D}_s\left(t + \frac{1}{\mu^*_{l(s)}} \bar{P}_{l(s)}(t)\right) = \bar{Q}_s(0) + \bar{A}_s(t), \tag{28}$$

where all involved functions are Lipschitz continuous and μ^* is defined in Lemma 1 as the set of achieved link rates. Equation (27) means that for each $t > 0$, whenever $\bar{P}_l(t) > 0$, there exists $\delta > 0$ such that $I_l(t+\delta) = I_l(t-\delta)$, i.e., $I_l(\cdot)$ is constant in $(t-\delta, t+\delta)$.

B. Definition of the Markov process

The state of the system at time t is formally defined as $(S(t), \tilde{Q}(t))$, where $\tilde{Q}(t)$ is the state of all shadow queues as defined earlier, and $S(t)$ is the state of all real queues, including the order of packet types in each queue. (Note that $(Q(t), \tilde{Q}(t))$ is uniquely determined by $(S(t), \tilde{Q}(t))$.) Given our model assumptions, this is an irreducible aperiodic Markov chain. Stability is ergodicity of this chain. Using the fact that we already know that the shadow queue process $\tilde{Q}(t)$ is ergodic, we will consider a different representation of the system process, which is also an irreducible aperiodic Markov chain, whose ergodicity is equivalent to that of $(S(t), \tilde{Q}(t))$. Namely, let us fix an arbitrary fixed state $\tilde{Q}_{**}$ of the shadow process, for example "empty" state. Consider this state as a regeneration point (of shadow process). A time interval between two consecutive "visits" of state $\tilde{Q}_{**}$ is a regeneration cycle. Suppose the system evolution process is constructed in the following way. If $\tilde{Q}(t)$ reaches state $\tilde{Q}_{**}$ at time t_0, the entire realization Ξ of the shadow process until the next regeneration point t_1 is randomly drawn (from the given fixed distribution of a regeneration cycle). So, given this realization Ξ, the evolution of process from t_0 to t_1 depends on the state $S(t_0)$ of real queues at t_0, and on (random) realization of the real arrival process in $(t_0, t_1]$ (which also depends on Ξ). The duration of Ξ we denote by $\tau(\Xi)$, and we know that $\mathbb{E}\tau(\Xi) < \infty$. An alternative definition of the process is

$$(S(t), \Xi(t), \xi(t)), \quad t \geq 0,$$

where $\Xi(t)$ in the regeneration-cycle realization the shadow process is "currently in" ($\Xi(t)$ only changes at regeneration

points), and $\xi(t)$ is the residual time of the current cycle. ($\xi(t)$ decreases by 1 in each time slot, until it reaches 0, at which point a new cycle realization is drawn and ξ is reset to new cycle duration.) Clearly, we always have $1 \le \xi(t) \le \tau(\Xi(t))$, and condition $\xi(t) = \tau(\Xi(t))$ indicates that t is a regeneration point.

Process $(S(\cdot), \Xi(\cdot), \xi(\cdot))$ is an irreducible aperiodic Markov chain. Since shadow process reaches state $\tilde{Q}_{**}$ from any other state within finite average time, the ergodicity of $(S(\cdot), \tilde{Q}(\cdot))$ is equivalent to that of $(S(\cdot), \Xi(\cdot), \xi(\cdot))$, and from now on we can focus on the latter.

C. Proof, using fluid limit technique

In the rest of the paper, we use the notation $\stackrel{P}{\to}$ to denote the convergence in probability; *u.o.c.* refers to uniform on compact sets convergence of deterministic vector-functions. We use $\|.\|$ to denote the L_1-norm of a vector; we will abuse this notation by writing $\|(S(t), \Xi(t), \xi(t))\|$ to mean $\|(Q(t)\| + \tau(\Xi(t))$, which is the total number of real packets plus the duration of the current regeneration cycle. Note that, since in our model the service rates on all links are uniformly bounded, the total number of all possible realizations of Ξ with a fixed finite duration $\tau(\Xi)$ is finite. This in particular implies that for any number r, the number of states with norm $\|(S, \Xi, \xi)\| \le r$ is finite.

The following lemma follows from more general results in [21] for discrete time countable Markov chains. (In the form (29), the stability condition was derived in [17] for *continuous time* countable Markov chains.)

Lemma 2: Suppose, there exists integer $T > 0$ such that the following holds. For any sequence of processes $(S^{(r)}(\cdot), \Xi^{(r)}(\cdot), \xi^{(r)}(\cdot))$ with the initial state norm $\|(S^{(r)}(0), \Xi^{(r)}(0), \xi^{(r)}(0))\| = r$ increasing to infinity,

$$\lim_{r\to\infty} \mathbb{E}\left[\frac{1}{r}\left\|(S^{(r)}(rT), \Xi^{(r)}(rT), \xi^{(r)}(rT))\right\|\right] = 0. \quad (29)$$

Then the Markov process $(S(\cdot), \Xi(\cdot), \xi(\cdot))$ is positive recurrent.

Using standard arguments (cf. [17]–[20]), along with the facts that $\mathbb{E}[\tau(\Xi)] < \infty$ and that all real packet exogenous input processes are uniformly stochastically bounded by a Poisson process with finite rate, it is easy to see that the family of random variables $\frac{1}{r}\|(S^{(r)}(rT), \Xi^{(r)}(rT), \xi^{(r)}(rT))\|$ is uniformly integrable (for any T). Also, by the definition of the process, for any r the first regeneration point is reached at the deterministic time $\xi^{(r)}(0)$ which is at most $\tau(\Xi^{(r)}(0)) \le \|(S^{(r)}(0), \Xi^{(r)}(0), \xi^{(r)}(0))\| = r$. After the first regeneration point is reached, the consequent regeneration cycle durations are i.i.d. (again by the process definition) with finite mean; in particular, using the functional strong law of large numbers for the sums of i.i.d. cycle durations, we see that, under the conditions of Lemma 2,

$$\frac{1}{r}\tau(\Xi^{(r)}(rT)) \stackrel{P}{\to} 0, \quad \forall T > 1.$$

Putting together all these observations, we see that the stability condition (29) in Lemma 2 can be weakened, namely we obtain the following fact.

Lemma 3: Suppose, there exists integer $T > 0$ such that the following holds. For any sequence of processes $(S^{(r)}(\cdot), \Xi^{(r)}(\cdot), \xi^{(r)}(\cdot))$, indexed by r increasing to infinity, with the initial state norm $\|(S^{(r)}(0), \Xi^{(r)}(0), \xi^{(r)}(0))\| \le r$, and with $\tau(\Xi^{(r)}(0)) = \xi^{(r)}(0)$ (i.e., time 0 being a regeneration point),

$$\frac{1}{r}\|(Q^{(r)}(rT)\| \stackrel{P}{\to} 0. \quad (30)$$

Then the Markov process $(S(\cdot), \Xi(\cdot), \xi(\cdot))$ is positive recurrent.

Thus, it will suffice to demonstrate that conditions of Lemma 3 hold. Let us define

$$X^{(r)}(t) \;:=\; (A^{(r)}(t), D(^{(r)}t), Q^{(r)}(t), I^{(r)}(t)), \quad (31)$$

for all real $t \ge 0$ using convention $X(t) = X(\lfloor t \rfloor)$. All component processes have been defined earlier; superscript (r) is simply the index of a process from the sequence defined in Lemma 3 statement.

Consider the corresponding sequence of scaled processes

$$X^r(t) = \frac{1}{r}X^{(r)}(rt), \;\; t \ge 0. \quad (32)$$

All processes $X^{(r)}(t)$ and $X^r(t)$ are in the Skorohod space of right-continuous functions having left limits. Then we have the following result:

Lemma 4: With probability 1 (i.e. for almost any outcome of the probability space), any subsequence of sequence $\{r\}$, has in turn another subsequence, along which

$$X^r(\cdot) \to \bar{X}(\cdot) = (\bar{A}(\cdot), \bar{D}(\cdot), \bar{Q}(\cdot), \bar{I}(\cdot)), \quad \text{u.o.c.}, \quad (33)$$

where $\bar{X}(\cdot)$ is a deterministic *fluid model solution* (namely a set of Lipschitz continuous functions, satisfying (23)-(28)), with $\|\bar{Q}(0)\| \le 1$.

Proof: From Lemma 1 we obtain the following functional strong law of large numbers properties of the exogenous arrival and link service processes. With probability 1,

$$\lim_{r\to\infty} \frac{1}{r}E^{(r)}(t) = \beta\lambda t \qquad \text{u.o.c.},$$
$$\lim_{r\to\infty} \frac{1}{r}M_l^{(r)}(t) = \mu_l^* t \qquad \text{u.o.c.}$$

For each outcome of the probability space, satisfying the above conditions, it remains to choose a converging subsequence and then essentially just to take the limit in (17)-(22). We omit further details, which are standard. ■

It remains to establish the following:

Lemma 5: There exist $T > 0$ such that for any fluid model solution $\bar{X}(\cdot)$ with $\|\bar{Q}(0)\| \le 1$,

$$\|\bar{Q}(t)\| = 0, \quad \forall t \ge T.$$

Lemma 5 is true by Bramson's result [13], because our equations (23)-(28) are same (up to a natural rescaling of variables) as the fluid model equations for a Kelly network studied in [13]. This completes the proof of Theorem 4.

ACKNOWLEDGMENTS

The work of the first two authors has been supported in part by DTRA Grant HDTRA1-08-1-0016, NSF Grant CNS 07-21286, and Army MURI 2008-01733.

REFERENCES

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1948, December 1992.

[2] X. Lin and N. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proceedings of IEEE Conference on Decision and Control*, vol. 2, Paradise Island, Bahamas, December 2004, pp. 1484–1489.

[3] A. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401–457, August 2005.

[4] ——, "Greedy primal-dual algorithm for dynamic resource allocation in complex networks," *Queueing Systems*, vol. 54, no. 3, pp. 203–220, 2006.

[5] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length based scheduling and congestion control," in *Proceedings of IEEE INFOCOM*, vol. 3, Miami, FL, March 2005, pp. 1794–1803.

[6] ——, "Joint congestion control, routing and MAC for stability and fairness in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, August 2006.

[7] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proceedings of IEEE INFOCOM*, vol. 3, Miami, FL, March 2005, pp. 1723–1734.

[8] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," in *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008, pp. 1103–1111.

[9] U. Akyol, M. Andrews, P. Gupta, J. Hobby, I. Saniee, and A. L. Stolyar, "Joint scheduling and congestion control in mobile ad-hoc networks," in *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008, pp. 619–627.

[10] L. Bui, R. Srikant, and A. Stolyar, "Optimal resource allocation for multicast sessions in multihop wireless networks," *Philosophical Transactions of the Royal Society, Series A*, vol. 366, no. 1872, pp. 2059–2074, June 2008.

[11] L. Ying, R. Srikant, and D. Towsley, "Cluster-based back-pressure routing algorithm," in *Proceedings of the IEEE INFOCOM*, Phoenix, AZ, April 2008, pp. 484–492.

[12] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key, "Horizon: Balancing TCP over multiple paths in wireless mesh network," in *Proceedings of the ACM MobiCom*, 2008, pp. 247–258.

[13] M. Bramson, "Convergence to equilibria for fluid models of FIFO queueing networks," *Queueing Systems*, vol. 22, no. 1-2, pp. 5–45, March 1996.

[14] R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, pp. 1969–1985, 1999.

[15] A. Stolyar, "Large number of queues in tandem: Scaling properties under back-pressure algorithm," *Queueing Systems*, vol. 67, no. 2, pp. 111–126, 2011.

[16] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, October 2000.

[17] S. Rybko and A. Stolyar, "Ergodicity of stochastic processes describing the operation of open queueing networks," *Problems of Information Transmission*, vol. 28, pp. 199–220, 1992.

[18] J. G. Dai, "On positive Harris recurrence for multiclass queueing networks: A unified approach via fluid limit models," *Annals of Applied Probability*, pp. 49–77, 1995.

[19] A. Stolyar, "On the stability of multiclass queueing networks: A relaxed sufficient condition via limiting fluid processes," *Markov Processes and Related Fields*, vol. 1, no. 4, pp. 491–512, 1995.

[20] J. Dai and S. Meyn, "Stability and convergence of moments for multiclass queueing networks via fluid limit models," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1889–1904, 1995.

[21] V. Malyshev and M. Menshikov, "Ergodicity, continuity, and analyticity of countable markov chains," *Transactions of Moscow Mathematical Society*, vol. 39, pp. 3–48, 1979.

Loc X. Bui received his B.Eng. degree in Electronics and Telecommunications from Posts and Telecommunications Institute of Technology at Ho Chi Minh City (PTIT-HCM) in 2003, and his M.S. and Ph.D. degrees in Electrical and Computer Engineering from University of Illinois at Urbana-Champaign in 2006 and 2008, respectively. From October 2008 to March 2010, he was with Airvana Inc., where he was a Software Engineer and then a Senior Sustaining Engineer. Since April 2010, he has joined Stanford University as a Postdoctoral Research Scholar in the Department of Management Science and Engineering. His research interests include communication networks, wireless communications, network control and optimization.

R. Srikant (S '90-M '91-SM '01-F '06) received his B.Tech. from the Indian Institute of Technology, Madras in 1985, his M.S. and Ph.D. from the University of Illinois in 1988 and 1991, respectively, all in Electrical Engineering. He was a Member of Technical Staff at AT&T Bell Laboratories from 1991 to 1995. He is currently with the University of Illinois at Urbana-Champaign, where he is the Fredric G. and Elizabeth H. Nearing Professor in the Department of Electrical and Computer Engineering, and a Research Professor in the Coordinated Science Lab. He was an associate editor of Automatica, the IEEE Transactions on Automatic Control, and the IEEE/ACM Transactions on Networking. He has also served on the editorial boards of special issues of the IEEE Journal on Selected Areas in Communications and IEEE Transactions on Information Theory. He was the chair of the 2002 IEEE Computer Communications Workshop in Santa Fe, NM and a program co-chair of IEEE INFOCOM, 2007. His research interests include communication networks, stochastic processes, queueing theory, information theory, and game theory.

Alexander Stolyar is a Distinguished Member of Technical Staff in the Industrial Mathematics and Operations Research Department of Bell Labs, Alcatel-Lucent, Murray Hill, New Jersey. He received Ph.D. in Mathematics from the Institute of Control Sciences, USSR Academy of Science, Moscow, in 1989. Before joining Bell Labs in 1998, he was with the Institute of Control Sciences (Moscow), Motorola (Arlington Heights, IL) and AT&T Labs-Research (Murray Hill, NJ). His research interests are in stochastic processes, queueing theory, and stochastic modeling of communication, especially wireless, systems. He is an associate editor of Operations Research; Queueing Systems - Theory and Applications; and Advances in Applied Probability.