Backpressure-based Packet-by-Packet Adaptive Routing in Communication Networks

Eleftheria Athanasopoulou, Loc Bui, Tianxiong Ji, R. Srikant, and Alexander Stolyar

***Abstract*—Backpressure-based adaptive routing algorithms where each packet is routed along a possibly different path have been extensively studied in the literature. However, such algorithms typically result in poor delay performance and involve high implementation complexity. In this paper, we develop a new adaptive routing algorithm built upon the widely-studied back-pressure algorithm. We decouple the routing and scheduling components of the algorithm by designing a probabilistic routing table which is used to route packets to per-destination queues. The scheduling decisions in the case of wireless networks are made using counters called shadow queues. The results are also extended to the case of networks which employ simple forms of network coding. In that case, our algorithm provides a low-complexity solution to optimally exploit the routing-coding tradeoff.**

I. Introduction

The back-pressure algorithm introduced in [1] has been widely studied in the literature. While the ideas behind scheduling using the weights suggested in that paper have been successful in practice in base stations and routers, the adaptive routing algorithm is rarely used. The main reason for this is that the routing algorithm can lead to poor delay performance due to routing loops. Additionally, the implementation of the back-pressure algorithm requires each node to maintain per-destination queues which can be burdensome for a wireline or wireless router. Motivated by these considerations, we re-examine the back-pressure routing algorithm in the paper and design a new algorithm which has much superior performance and low implementation complexity.

Prior work in this area [2] has recognized the importance of doing shortest-path routing to improve delay performance and modified the back-pressure algorithm to bias it towards taking shortest-hop routes. A part of our algorithm has similar motivating ideas. In addition to provably throughput-optimal routing which minimizes the number of hops taken by packets in the network, we decouple (to a certain degree) routing and scheduling in the network through the use of probabilistic routing tables and the so-called shadow queues. The min-hop routing idea was studied first in a conference paper [3] and shadow queues were introduced in [4] and [5], but the key step of partial decoupling the routing and scheduling which leads to both significant delay reduction and the use of per-next-hop queueing is original here. In [4], the authors introduced the shadow queue to solve a fixed routing problem. The min-hop routing idea is also studied in [6] but their solution requires even more queues than the original back-pressure algorithm. Compared to [4], the main purpose of this paper is to study if the shadow queue approach extends to the case of scheduling and routing. The first contribution is to come up with a formulation where the number of hops is minimized. It is interesting to contrast this contribution with [6]. The formulation in [6] has the same objective as ours but their solution involves per-hop queues, which dramatically increases the number of queues, even compared to the back-pressure algorithm. Our solution is significantly different: we use the same number of shadow queues as the back-pressure algorithm, but the number of real queues is very small (per-neighbor). The new idea here is to perform routing via probabilistic splitting, which allows the dramatic reduction in the number of real queues. Finally, an important observation in this paper, not found in [4], is that the partial "decoupling" of shadow back-pressure and real packet transmission allows us to activate more links than a regular back-pressure algorithm would. This idea appears to be essential to reduce delays in the routing case, as shown in the simulations.

We also consider networks where simple forms of network coding is allowed [7]. In such networks, a relay between two other nodes XORs packets and broadcast them to decrease the number of transmissions. There is a tradeoff between choosing long routes to possibly increase network coding opportunities (see the notion of reverse carpooling in [8]) and choosing short routes to reduce resource usage. Our adaptive routing algorithm can be modified to automatically realize this tradeoff with good delay performance. In addition, network coding requires each node to maintain more queues [9] and our routing solution at least reduces the number of queues to be maintained for routing purposes, thus partially mitigating the problem. An offline algorithm for optimally computing the routing-coding tradeoff was proposed in [10]. Our optimization formulation bears similarities to this work but our main focus is on designing low-delay on-line algorithms. Back-pressure solutions to network coding problems have also been studied in [11], [12], [13], but the adaptive routing-coding tradeoff solution that we propose here has not been studied previously.

We summarize our main results below.

- Using the concept of shadow queues, we partially de-

[1]This work was supported by MURI BAA 07-036.18, ARO MURI, DTRA Grant HDTRA1-08-1-0016, and NSF grants 07-21286, 05-19691 and 03-25673.

E. Athanasopoulou, T. Ji and R. Srikant are with Coordinated Science Laboratory and Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign. (email: {athanaso, tji2, rsrikant}@illinois.edu).

L. Bui is with School of Engineering, Tan Tao University. (email: locbui@ieee.org).

A. Stolyar is with Bell Labs, Alcatel-Lucent, NJ, USA. (email: stolyar@research.bell-labs.com).

couple routing and scheduling. A shadow network is used to update a probabilistic routing table which packets use upon arrival at a node. The same shadow network, with back-pressure algorithm, is used to activate transmissions between nodes; however, first, actual transmissions send packets from FIFO per-link queues and, second, potentially more links are activated, in addition to those activated by the shadow algorithm.

- The routing algorithm is designed to minimize the average number of hops used by packets in the network. This idea, along with the scheduling/routing decoupling, leads to delay reduction compared with the traditional back-pressure algorithm.
- Each node has to maintain counters, called shadow queues, per destination. This is very similar to the idea of maintaining a routing table per destination. But the real queues at each node are per-next-hop queues in the case of networks which do not employ network coding. When network coding is employed, per-previous-hop queues may also be necessary but this is a requirement imposed by network coding, not by our algorithm.
- The algorithm can be applied to wireline and wireless networks. Extensive simulations show dramatic improvement in delay performance compared to the back-pressure algorithm.

The rest of the paper is organized as follows. We present the network model in Section II. In Section III and IV, the traditional back-pressure algorithm and its modified version are introduced. We develop our adaptive routing and scheduling algorithm for wireline and wireless networks with and without network coding in Section V, VI and VII. In Section VIII, the simulation results are presented. We conclude our paper in Section IX.

II. The Network Model

We consider a multi-hop wireline or wireless network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of directed links. A directed link that can transmit packets from node n to node j is denoted by $(nj) \in \mathcal{L}$. We assume that time is slotted and define the link capacity c_{nj} to be the maximum number of packets that link (nj) can transmit in one time slot.

Let $\mathcal{F}$ be the set of flows that share the network. Each flow is associated with a source node and a destination node, but no route is specified between these nodes. This means that the route can be quite different for packets of the same flow. Let $b(f)$ and $e(f)$ be source and destination nodes, respectively, of flow f. Let x_f be the rate (packets/slot) at which packets are generated by flow f. If the demand on the network, i.e., the set of flow rates, can be satisfied by the available capacity, there must exist a routing algorithm and a scheduling algorithm such that the link rates lie in the capacity region. To precisely state this condition, we define μ_{nj}^d to be the rate allocated on link (nj) to packets destined for node d. Thus, the total rate allocated to all flows at link (nj) is given by $\mu_{nj} := \sum_{d \in \mathcal{N}} \mu_{nj}^d$. Clearly, for the network to be able to meet the traffic demand, we should have:

$$\{\mu_{nj}\}_{(nj)\in\mathcal{L}} \in \Lambda,$$

where Λ is the capacity region of the network for 1-hop traffic. The capacity region of the network for 1-hop traffic contains all sets of rates that are stabilizable by some kind of scheduling policy assuming all traffics are 1-hop traffic. As a special case, in the wireline network, the constraints are:

$$\mu_{nj} \le c_{nj}, \qquad \forall (nj).$$

As opposed to Λ, let Υ denote the capacity region of the multi-hop network, i.e., for any set of flows $\{x_f\}_{f\in\mathcal{F}} \in \Upsilon$, there exists some routing and scheduling algorithms that stabilize the network.

In addition, a flow conservation constraint must be satisfied at each node, i.e., the total rate at which traffic can possibly arrive at each node destined to d must be less than or equal to the total rate at which traffic can depart from the node destined to d :

$$\begin{aligned} \sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n, e(f)=d\}} + \sum_{l:(ln)\in\mathcal{L}} \mu_{ln}^d \\ \le \sum_{j:(nj)\in\mathcal{L}} \mu_{nj}^d, \end{aligned} \tag{1}$$

where I denotes the indicator function. Given a set of arrival rates $x = \{x_f\}_{f\in\mathcal{F}}$ that can be accommodated by the network, one version of the multi-commodity flow problem is to find the traffic splits μ_{nj}^d such that (1) is satisfied. However, finding the appropriate traffic split is computationally prohibitive and requires knowledge of the arrival rates. The back-pressure algorithm to be described next is an adaptive solution to the multi-commodity flow problem.

III. Throughput-Optimal Back-pressure Algorithm and Its Limitations

The back-pressure algorithm was first described in [1] in the context of wireless networks and independently discovered later in [14] as a low-complexity solution to certain multi-commodity flow problems. This algorithm combines the scheduling and routing functions together. While many variations of this basic algorithm have been studied, they primarily focus on maximizing throughput and do not consider QoS performance. Our algorithm uses some of these ideas as building blocks and therefore, we first describe the basic algorithm, its drawbacks and some prior solutions.

The algorithm maintains a queue for each destination at each node. Since the number of destinations can be as large as the number of nodes, this per-destination queueing requirement can be quite large for practical implementation in a network. At each link, the algorithm assigns a weight to each possible destination which is called *back-pressure.* Define the back-pressure at link (nj) for destination d at slot t to be

$$w_{nj}^d[t] = Q_{nd}[t] - Q_{jd}[t],$$

where $Q_{nd}[t]$ denotes the number of packets at node n destined for node d at the beginning of time slot t. Under this notation, $Q_{nn}[t] = 0, \forall t$. Assign a weight w_{nj} to each link (nj), where

w_{nj} is defined to be the maximum back-pressure over all possible destinations, i.e.,

$$w_{nj}[t] = \max_d w_{nj}^d[t].$$

Let d_{nj}^* be the destination which has the maximum weight on link (nj),

$$d_{nj}^*[t] = \arg\max_d \{w_{nj}^d[t]\}. \quad (2)$$

If there are ties in the weights, they can be broken arbitrarily. Packets belonging to destination $d_{nj}^*[t]$ are scheduled for transmission over the activated link (nj). A schedule is a set of links that can be activated simultaneously without interfering with each other. Let Γ denote the set of all schedules. The back-pressure algorithm finds an optimal schedule $\pi^*[t]$ which is derived from the optimization problem:

$$\pi^*[t] = \arg\max_{\pi\in\Gamma} \sum_{(nj)\in\pi} c_{nj} w_{nj}[t]. \quad (3)$$

Specially, if the capacity of every link has the same value, the chosen schedule maximizes the sum of weights in any schedule.

At time t, for each activated link $(nj) \in \pi^*[t]$ we remove c_{nj} packets from $Q_{nd_{nj}^*[t]}$ if possible, and transmit those packets to $Q_{jd_{nj}^*[t]}$. We assume that the departures occur first in a time slot, and external arrivals and packets transmitted over a link (nj) in a particular time slot are available to node j at the next time slot. Thus the evolution of the queue $Q_{nd}[t]$ is as follows:

$$\begin{aligned} Q_{nd}[t+1] &= Q_{nd}[t] - \sum_{j:(nj)\in\mathcal{L}} I_{\{d_{nj}^*[t]=d\}}\, \hat{\mu}_{nj}[t] \\ &\quad + \sum_{l:(ln)\in\mathcal{L}} I_{\{d_{ln}^*[t]=d\}}\, \hat{\mu}_{ln}[t] \\ &\quad + \sum_{f\in\mathcal{F}} I_{\{b(f)=n,e(f)=d\}}\, a_f[t], \end{aligned} \quad (4)$$

where $\hat{\mu}_{nj}[t]$ is the number of packets transmitted over link (nj) in time slot t and $a_f[t]$ is the number of packets generated by flow f at time t. It has been shown in [1] that the back-pressure algorithm maximizes the throughput of the network.

A key feature of the back-pressure algorithm is that packets may not be transferred over a link unless the back-pressure over a link is non-negative and the link is included in the picked schedule. This feature prevents further congesting nodes that are already congested, thus providing the adaptivity of the algorithm. Notice that because all links can be activated without interfering with each other in the wireline network, Γ is the set of all links. Thus the back-pressure algorithm can be localized at each node and operated in a distributed manner in the wireline network.

The back-pressure algorithm has several disadvantages that prohibit practical implementation:

- The back-pressure algorithm requires maintaining queues for each potential destination at each node. This queue management requirement could be a prohibitive overhead for a large network.
- The back-pressure algorithm is an adaptive routing algorithm which explores the network resources and adapts to different levels of traffic intensity. However it might also lead to high delays because it may choose long paths unnecessarily. High delays are also a result of maintaining a large number of queues at each node, and each of those queues being large. The queues can be large because, under back-pressure algorithm, average size of a per-destination queue at a node can grow with the distance from the node to the destination. Furthermore, large number of queues takes away statistical multiplexing advantage: since only one queue can be scheduled at a time, some of the allocated transmission capacity can be left unused if the scheduled queue is too short this can contribute to high latency as well.

In this paper, we address the high delay and queueing complexity issues. The computational complexity issue for wireless networks is not addressed here. We simply use the recently studied greedy maximal scheduling (GMS) algorithm. Here we call it the *largest-weight-first* algorithm, in short, LWF algorithm. LWF algorithm requires the same queue structure that the back-pressure algorithm uses. It also calculates the back-pressure at each link using the same way. The difference between these two algorithms only lies in the methods to pick a schedule. Let $\mathcal{S}$ denote the set of all links initially. Let $\mathcal{N}_b(l)$ be the set of links within the interference range of link l including l itself. At each time slot, the LWF algorithm picks a link l with the maximum weight first, and removes links within the interference range of link l from $\mathcal{S}$, i.e., $\mathcal{S} = \mathcal{S}\backslash\mathcal{N}_b(l)$; then it picks the link with the maximum weight in the updated set $\mathcal{S}$, and so forth. It should be noticed that LWF algorithm reduces the computational complexity with a price of the reduction of the network capacity region. The LWF algorithm where the weights are queue lengths (not back-pressures) has been extensively studied in [15], [16], [17], [18], [19]. While these studies indicate that there may be reduction in throughput due to LWF in certain special network topologies, it seems to perform well in practice and so we adopt it here for simulations.

In the rest of the paper, we present our main results which eliminate many of the problems associated with the back-pressure algorithm.

IV. Min-resource Routing Using Back-pressure Algorithm

As mentioned in Section III, the back-pressure algorithm explores all paths in the network and as a result may choose paths which are unnecessarily long which may even contain loops, thus leading to poor performance. We address this problem by introducing a cost function which measures the total amount of resources used by all flows in the network. Specially, we add up traffic loads on all links in the network and use this as our cost function. The goal then is to minimize this cost subject to network capacity constraints.

Given a set of packet arrival rates that lie within the capacity region, our goal is to find the routes for flows so that we use as few resources as possible in the network. Thus, we formulate

the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{(nj)\in\mathcal{L}} \mu_{nj} \qquad (5)\\ s.t. \quad & \sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}} + \sum_{(ln)\in\mathcal{L}} \mu^d_{ln} \le \sum_{(nj)\in\mathcal{L}} \mu^d_{nj}, \\ & \qquad \forall d\in\mathcal{N}, n\in\mathcal{N}, \\ & \{\mu_{nj}\}_{(nj)\in\mathcal{L}} \in \Lambda. \end{aligned}$$

We now show how a modification of the back-pressure algorithm can be used to solve this min-resource routing problem. (Note that similar approaches have been used in [20], [21], [22], [23], [24] to solve related resource allocation problems.)

Let $\{q_{nd}\}$ be the Lagrange multipliers corresponding to the flow conservation constraints in problem (5). Appending these constraints to the objective, we get

$$\begin{aligned} & \min_{\boldsymbol{\mu}\in\Lambda} \sum_{(nj)\in\mathcal{L}} \mu_{nj} + \sum_{n,d} q_{nd}\Big(\sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}} \\ & \qquad + \sum_{(ln)\in\mathcal{L}} \mu^d_{ln} - \sum_{(nj)\in\mathcal{L}} \mu^d_{nj}\Big) \qquad (6)\\ = \ & \min_{\boldsymbol{\mu}\in\Lambda}\Big(-\sum_{(nj)\in\mathcal{L}}\sum_{d} \mu^d_{nj}\big(q_{nd}-q_{jd}-1\big) \\ & \qquad -\sum_{n,d} q_{nd}\sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}}\Big). \end{aligned}$$

If the Lagrange multipliers are known, then the optimal $\boldsymbol{\mu}$ can be found by solving

$$\max_{\boldsymbol{\mu}\in\Lambda} \sum_{(nj)\in\mathcal{L}} \mu_{nj} w_{nj}$$

where $w_{nj} = \max_d (q_{nd} - q_{jd} - 1)$. The form of the constraints in (5) suggests the following update algorithm to compute q_{nd} :

$$\begin{aligned} q_{nd}[t+1] = \Big[q_{nd}[t] + \frac{1}{M}\Big(\sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}} \\ + \sum_{(ln)\in\mathcal{L}} \mu^d_{ln} - \sum_{(nj)\in\mathcal{L}} \mu^d_{nj}\Big)\Big]^+ \qquad (7) \end{aligned}$$

where $\frac{1}{M}$ is a step-size parameter. See [25] for details. Notice that $Mq_{nd}[t]$ looks very much like a queue update equation, except for the fact that arrivals into Q_{nd} from other links may be smaller than μ^d_{ln} when Q_{ld} does not have enough packets. This suggests the following algorithm.

Min-resource routing by back-pressure: At time slot t,

- Each node n maintains a separate queue of packets for each destination d; its length is denoted $Q_{nd}[t]$. Each link is assigned a weight

$$w_{nj}[t] = \max_d \left(\frac{1}{M}Q_{nd}[t] - \frac{1}{M}Q_{jd}[t] - 1\right), \qquad (8)$$

  where $M > 0$ is a parameter.
- Scheduling/routing rule:

$$\pi^*[t] \in \arg\max_{\pi\in\Gamma} \sum_{(nj)\in\pi} c_{nj} w_{nj}[t]. \qquad (9)$$

- For each activated link $(nj) \in \pi^*[t]$ we remove c_{nj} packets from $Q_{nd^*_{nj}[t]}$ if possible, and transmit those packets to $Q_{jd^*_{nj}[t]}$, where $d^*_{nj}[t]$ achieves the maximum in (8).

Note that the above algorithm does not change if we replace the weights in (8) by the following, re-scaled ones:

$$w_{nj}[t] = \max_d \left(Q_{nd}[t] - Q_{jd}[t] - M\right), \qquad (10)$$

and therefore, compared with the traditional back-pressure scheduling/routing, the only difference is that each link weight is equal to the maximum differential backlog *minus parameter* M. ($M = 0$ reverts the algorithm to the traditional one.) For simplicity, we call this algorithm *M-back-pressure algorithm*.

The performance of the stationary process which is "produced" by the algorithm with fixed parameter M is within $o(1)$ of the optimal as M goes to ∞ (analogous to the proofs in [21], [22]; see also the related proof in [23], [24]):

$$\left| \mathbb{E}\left[\sum_{(nj)\in\mathcal{L}} \mu_{nj}[\infty]\right] - \sum_{(nj)\in\mathcal{L}} \mu^*_{nj} \right| = o(1),$$

where μ^* is an optimal solution to (5).

Figure 1 illustrates how the M-back-pressure algorithm works in a simple wireline network. All links can be activate simultaneously without interfering with each other. Notice that the backlog difference of route 1 is 6 and the backlog difference of route 2 is 4. Because the backlog difference of route 2 is smaller than M, route 2 is blocked at current traffic load. The M-back-pressure algorithm will automatically choose route 1 which is shorter. Therefore, a proper M can avoid long routes in when the traffic is not close to capacity.

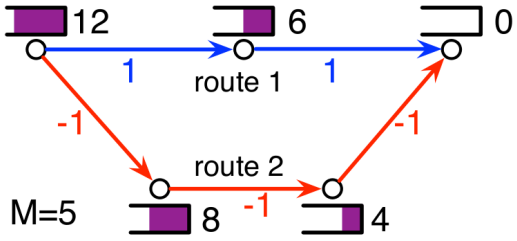


Fig. 1. Picture illustrating link weights under the M-backpressure algorithm.

Although M-back-pressure algorithm could reduce the delay by forcing flows to go through shorter routes, simulations indicate a significant problem with the basic algorithm presented above. A link can be scheduled only if the back-pressure of at least one destination is greater than or equal to M. Thus, at light to moderate traffic loads, the delays could be high since the back-pressure may not build up sufficiently fast. In order to overcome all these adverse issues, we develop a new routing algorithm in the following section. The solution also simplifies the queueing data structure to be maintained at each node.

V. PARN: Packet-by-Packet Adaptive Routing and Scheduling Algorithm For Networks

In this section, we present our adaptive routing and scheduling algorithm. We will call it PARN (Packet-by-Packet Adaptive Routing for Networks) for ease for repeated reference later. First, we introduce the queue structure that is used in PARN.

In the traditional back-pressure algorithm, each node n has to maintain a queue q_{nd} for each destination d. Let $|\mathcal{N}|$ and $|\mathcal{D}|$ denote the number of nodes and the number of destinations in the network, respectively. Each node maintains $|\mathcal{D}|$ queues. Generally, each pair of nodes can communicate along a path connecting them. Thus, the number of queues maintained at each node can be as high as one less than the number of nodes in the network, i.e., $|\mathcal{D}|$=$|\mathcal{N}|-1$.

Instead of keeping a queue for every destination, each node n maintains a queue q_{nj} for every neighbor j, which is called a *real queue*. Notice that real queues are per-neighbor queues. Let J_n denote the number of neighbors of node n, and let $J_{max} = \max_n J_n$. The number of queues at each node is no greater than J_{max}. Generally, J_{max} is much smaller than $|\mathcal{N}|$. Thus, the number of queues at each node is much smaller compared with the case using the traditional back-pressure algorithm.

In additional to real queues, each node n also maintains a counter, which is called *shadow queue,* p_{nd} for each destination d. Unlike the real queues, counters are much easier to maintain even if the number of counters at each node grows linearly with the size of the network. A back-pressure algorithm run on the shadow queues is used to decide which links to activate. The statistics of the link activation are further used to route packets to the per-next-hop neighbor queues mentioned earlier. The details are explained next.

A. Shadow Queue Algorithm – M-back-pressure Algorithm

The shadow queues are updated based on the movement of fictitious entities called shadow packets in the network. The movement of the fictitious packets can be thought of as an exchange of control messages for the purposes of routing and schedule. Just like real packets, shadow packets arrive from outside the network and eventually exit the network. The external shadow packet arrivals are general as follows: when an exogenous packet arrives at node n to the destination d, the shadow queue p_{nd} is incremented by 1, and is further incremented by 1 with probability ε in addition. Thus, if the arrival rate of a flow f is x_f, then the flow generates "shadow traffic" at a rate $x_f(1+\varepsilon)$. In words, the incoming shadow traffic in the network is $(1+\varepsilon)$ times of the incoming real traffic.

The back-pressure for destination d on link (nj) is taken to be

$$w^d_{nj}[t] = p_{nd}[t] - p_{jd}[t] - M,$$

where M is a properly chosen parameter. The choice of M will be discussed in the simulations section.

The evolution of the shadow queue $p_{nd}[t]$ is

$$\begin{aligned} p_{nd}[t+1] &= p_{nd}[t] - \sum_{j:(nj)\in\mathcal{L}} I_{\{d^*_{nj}[t]=d\}}\,\hat{\mu}_{nj}[t] \\ &\quad + \sum_{l:(ln)\in\mathcal{L}} I_{\{d^*_{ln}[t]=d\}}\,\hat{\mu}_{ln}[t] \\ &\quad + \sum_{f\in\mathcal{F}} I_{\{b(f)=n,e(f)=d\}}\,\hat{a}_f[t], \end{aligned} \tag{11}$$

where $\hat{\mu}_{nj}[t]$ is the number of shadow packets transmitted over link (nj) in time slot t, $d^*_{nj}[t]$ is the destination that has the maximum weight on link (nj), and $\hat{a}_f[t]$ is the number of shadow packets generated by flow f at time t. The number of shadow packets scheduled over the links at each time instant is determined by the back-pressure algorithm in equation (9).

From the above description, it should be clear that the shadow algorithm is the same as the traditional back-pressure algorithm, except that it operates on the shadow queueing system with an arrival rate slightly larger than the real external arrival rate of packets. Note the shadow queues do not involve any queueing data structure at each node; there are no packets to maintain in a FIFO order in each queue. The shadow queue is simply a counter which is incremented by 1 upon a shadow packet arrival and decremented by 1 upon a departure.

The back-pressure algorithm run on the shadow queues is used to activate the links. In other words, if $\pi^*_{nj} = 1$ in (9), then link (nj) is activated and packets are served from the real queue at the link in a first-in, first-out fashion. This is, of course, very different from the traditional back-pressure algorithm where a link is activated to serve packets to a particular destination. Thus, we have to develop a routing scheme that assigns packets arriving to a node to a particular next-hop neighbor so that the system remains stable. We design such an algorithm next.

B. Adaptive Routing Algorithms

Now we discuss how a packet is routed once it arrives at a node. Let us define a variable $\sigma^d_{nj}[t]$ to be the number of shadow packets "transferred" from node n to node j for destination d during time slot t by the shadow queue algorithm. Let us denote by $\bar{\sigma}^d_{nj}$ the expected value of $\sigma^d_{nj}[t]$, when the shadow queueing process is in a stationary regime; let $\hat{\sigma}^d_{nj}[t]$ denote an estimate of $\bar{\sigma}^d_{nj}$, calculated at time t. (In the simulations we use the exponential averaging, as specified in the next section.)

At each time slot, the following sequence of operations occurs at each node n. A packet arriving at node n for destination d is inserted in the real queue q_{nj} for next-hop neighbor j with probability

$$P^d_{nj}[t] = \frac{\hat{\sigma}^d_{nj}[t]}{\sum_{k:(nk)\in\mathcal{L}} \hat{\sigma}^d_{nk}[t]}. \tag{12}$$

Thus, the estimates $\hat{\sigma}^d_{nj}[t]$ are used to perform routing operations: in today's routers, based on the destination of a packet, a packet is routed to its next hop based on routing table entries. Instead, here, the $\bar{\sigma}$'s are used to probabilistically choose the next hop for a packet. Packets waiting at link (nj)

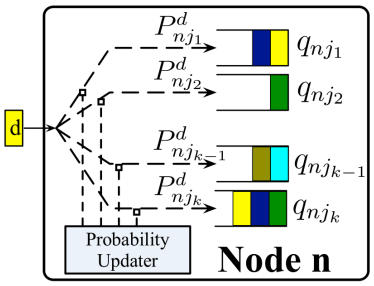


Fig. 2. Probabilistic splitting algorithm at Node n.

are transmitted over the link when that link is scheduled (See Figure 2).

The first question that one must ask about the above algorithm is whether it is stable if the packet arrival rates from flows are within the capacity region of the multi-hop network. This is a difficult question, in general. Since the shadow queues are positive recurrent, "good" estimates $\hat{\sigma}^d_{nj}[t]$ can be maintained by simple averaging (e.g. as specified in the next section), and therefore the probabilities in (12) will stay close to their "ideal" values

$$\bar{P}^d_{nj} = \frac{\bar{\sigma}^d_{nj}}{\sum_{k:(nk)\in\mathcal{L}} \bar{\sigma}^d_{nk}}.$$

The following theorem asserts that the real queues are stable under the additional assumption that the routing probabilities P^d_{nj} are fixed at their ideal values $\bar{P}^d_{nj}$ (as opposed to being updated via (12), which is what the actual algorithm does).

Theorem 1: Suppose the routing probabilities are fixed at $\bar{P}^d_{nj}$. Assume that there exists a delta such that $\{x_f(1+\epsilon+\delta)\}$ lies in Γ. Let $a_f[t]$ be the number of packets arriving from flow f at time slot t, with $E(a_f[t]) = x_f$ and $E(a_f[t]) < \infty$. Assume that the arrival process is independent across time slots and flows (this assumption can be considerably relaxed). Then, the Markov chain, jointly describing the evolution of shadow queues and real FIFO queues (whose state include the destination of the real packet in each position of each FIFO queue), is positive recurrent.

Proof: The key ideas behind the proof are outlined. The details are similar to the proof in [4] and are omitted.

- The average rate at which packets arrive to link (nj) is strictly smaller than the capacity allocated to the link by the shadow process if $\varepsilon > 0$. (This fact is verified in Appendix A.)
- It follows that the fluid limit of the real-queue process is same as that of the networks in [26]. Such fluid limit is stable [26], which implies the stability of our process as well.

■

VI. Implementation Details

The algorithm presented in the previous section ensures that the queue lengths are stable. In this section, we discuss a number of enhancements to the basic algorithm to improve performance.

A. Exponential Averaging

To compute $\hat{\sigma}^d_{nj}[t]$ we use the following iterative exponential averaging algorithm:

$$\hat{\sigma}^d_{nj}[t] = (1-\beta)\,\hat{\sigma}^d_{nj}[t-1] + \beta\,\sigma^d_{nj}[t], \tag{13}$$

where $0 < \beta < 1$.

B. Token Bucket Algorithm

Computing the average shadow rate $\hat{\sigma}^d_{nj}[t]$ and generating random numbers for routing packets may impose a computational overhead of routers which should be avoided if possible. Thus, as an alternative, we suggest the following simple algorithm. At each node n, for each next-hop neighbor j and each destination d, maintain a token bucket r^d_{nj}. Consider the shadow traffic as a guidance of the real traffic, with tokens *removed* as shadow packets traverse the link. In detail, the token bucket is decremented by $\sigma^d_{nj}[t]$ in each time slot, but cannot go below the lower bound 0:

$$r^d_{nj}[t] = \max\{r^d_{nj}[t-1] - \sigma^d_{nj}[t], 0\}.$$

When $r^d_{nj}[t-1] - \sigma^d_{nj}[t] < 0$, we say that $\sigma^d_{nj}[t] - r^d_{nj}[t-1]$ tokens (associated with bucket r^d_{nj}) are "wasted" in slot t. Upon a packet arrival at node n for destination d, find the token bucket $r^d_{nj^*}$ which has the smallest number of tokens (the minimization is over next-hop neighbors j), breaking ties arbitrarily, add the packet to the corresponding real queue q_{nj^*} and add one token to the corresponding bucket:

$$r^d_{nj^*}[t] = r^d_{nj^*}[t-1] + 1. \tag{14}$$

To explain how this algorithm works, denote by $\bar{\sigma}^d_{nj}$ the average value of $\sigma^d_{nj}[t]$ (in stationary regime), and by η^d_n the average rate at which real packets for destination d arrive at node n. Due to the fact that real traffic is injected by each source at the rate strictly less than the shadow traffic, we have

$$\eta^d_n < \sum_j \bar{\sigma}^d_{nj}. \tag{15}$$

For a single-node network, (15) just means that arrival rate is less than available capacity. More generally, it is an assumption that needs to be proved. However, here our goal is to provide an intuition behind the token bucket algorithm, so we simply assume (15). Condition (15) guarantees that the token processes are *stable* (that is, roughly, they cannot runaway to infinity) since the total arrival rate to the token buckets at a node is less than the total service rate and the arrivals employ a join-the-shortest-queue discipline. Moreover, since $r^d_{nj}[t]$ are random processes, the token buckets will "hit 0" in a non-zero fraction of time slots, except in some degenerate cases; this in turn means that the arrival rate of packets at the token bucket must be less than the token generation rate:

$$\eta^d_{nj} < \bar{\sigma}^d_{nj}, \tag{16}$$

where η^d_{nj} is the actual rate at which packets arriving at n and destined for d are routed along link (nj). Inequality (16) thus describes the idea of the algorithm.

Ideally, in addition to (16), we would like to have the ratios $\eta^d_{nj}/\bar{\sigma}^d_{nj}$ to be equal across all j, i.e., the real packet arrival

rates at the outgoing links of a node should be proportional to the shadow service rates. It is not difficult to see that if ε is very small, the proportion will be close to ideal. In general, the token-based algorithm does not guarantee that, that is why it is an approximation.

Also, to ensure implementation correctness, instead of (14), we use

$$r^d_{nj_*}[t] = \min\{r^d_{nj_*}[t-1] + 1, B\}, \tag{17}$$

i.e., the value of $r^d_{nj_*}[t]$ is not allowed to go above some relatively large value B, which is a parameter of the order of $O(1/\epsilon)$. Under "normal circumstances", $r^d_{nj_*}[t]$ "hitting" ceiling B is a rare event, occurring due to the process randomness. The main purpose of having the upper bound B is to detect serious anomalies when, for whatever reason, the condition (15) "breaks" for prolonged periods of time – such situation is detected when any $r^d_{nj_*}[t]$ hits the upper bound B frequently.

C. Extra Link Activation

Under the shadow back-pressure algorithm, only links with back-pressure greater than or equal to M can be activated. The stability theory ensures that this is sufficient to render the real queues. On the other hand, the delay performance can still be unacceptable. Recall that the parameter M was introduced to discourage the use of unnecessarily long paths. However, under light and moderate traffic loads, the shadow back-pressure at a link may be frequently less than M, and thus, packets at such links may have to wait a long time before they are processed. One way to remedy the situation is to activate additional links beyond those activated by the shadow back-pressure algorithm.

The basic idea is as follows: in each time slot, first run the shadow back-pressure algorithm. Then, add additional links to make the schedule maximal. If the extra activation procedure depends only on the state of shadow queues (but beyond that, can be random and/or arbitrarily complex), then the stability result of Theorem 1 still holds (with essentially same proof). Informally, the stability prevails, because the shadow algorithm alone provides sufficient average throughput on each link, and adding extra capacity "does not hurt"; thus, with such extra activation, a certain degree of "decoupling" between routing (totally controlled by shadow queues) and scheduling (also controlled by shadow queues, but not completely) is achieved.

For example, in the case of wireline networks, by the above arguments, all links can be activated all the time. The shadow routing algorithm ensures that the arrival rate at each link is less than its capacity. In this case the *complete* decoupling of routing and scheduling occurs.

In practice, activating extra links which have large queue backlogs leads to better performance than activating an arbitrary set of extra links. However, in this case, the extra activation procedure depends on the state of real queues which makes the issue of validity of an analog of Theorem 1 much more subtle. We believe that the argument in this subsection provides a good motivation for our algorithm, which is confirmed by simulations.

D. The Choice of the Parameter ε

From basic queueing theory, we expect the delay at each link to be inversely proportional to the mean capacity minus the arrival rate at the link. In a wireless network, the capacity at a link is determined by the shadow scheduling algorithm. This capacity is guaranteed to be at least equal to the shadow arrival rate. The arrival rate of real packets is of course smaller. Thus, the difference between the link capacity and arrival rate could be proportional to epsilon. Thus, epsilon should be sufficiently large to ensure small delays while it should be sufficiently small to ensure that the capacity region is not diminished significantly. In our simulations, we found that choosing $\varepsilon = 0.1$ provides a good tradeoff between delay and network throughput.

In the case of wireline networks, recall from the previous subsection that all links are activated. Therefore, the parameter epsilon plays no role here.

VII. Extension to The Network Coding Case

In this section, we extend our approach to consider networks where network coding is used to improve throughput. We consider a simple form of network coding illustrated in Figure 3. When i and j each have a packet to send to the other through an intermediate relay n, traditional transmission requires the following set of transmissions: send a packet a from i to n, then n to j, followed by j to n and n to i. Instead, using network coding, one can first send from i to n, then j to n, XOR the two packets and broadcast the XORed packet from n to both i and j. This form of network coding reduces the number of transmissions from four to three. However, the network coding can only improve throughput only if such coding opportunities are available in the network. Routing plays an important role in determining whether such opportunities exist. In this section, we design an algorithm to automatically find the right tradeoff between using possibly long routes to provide network coding opportunities and the delay incurred by using long routes.

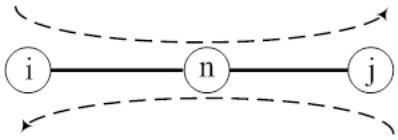


Fig. 3. Network coding opportunity.

A. System Model

We still consider the wireless network represented by the graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$. Let x_f be the rate (packets/slot) at which packets are generated by flow f. To facilitate network coding, each node must not only keep track of the destination of the packet, but also remember the node from which a packet was received. Let μ^d_{lnj} be the rate at which packets received from either node l or flow l, destined for node d, are scheduled over link (nj). Note that, for compactness of notation, we allow l in the definition of μ^d_{lnj} to denote either a flow or a node. We assume μ^d_{lnj} is zero when such a transmission is not feasible,

i.e., when n is not the source node or d is not the destination node of flow l, or if (ln) or (nj) is not in $\mathcal{L}$. At node n, the network coding scheme may generate a coded packet by "XORing" two packets received from previous-hop nodes l and j destined for the destination nodes d and d' respectively, and broadcast the coded packet to nodes j and l. Let $\mu_{n|jl}^{d,d'}$ denote the rate at which coded packets can be transferred from node n to nodes j and l destined for nodes d and d', respectively. Notice that, due to symmetry, the following equality holds $\mu_{n|jl}^{d,d'} = \mu_{n|lj}^{d',d}$. Assume $\mu_{n|jl}^{d,d'}$ to be zero if at least one of $(nl), (ln), (nj)$ and (jn) doesn't belong to $\mathcal{L}$. Note that $\mu_{lnj}^{d} = 0$ when $d = l$ or $d = n$, and $\mu_{n|jl}^{d,d'} = 0$ when $d = n$ or $d' = n$.

There are two kinds of transmissions in our network model: point-to-point transmissions and broadcast transmissions. The total point-to-point rate at which packets received externally or from a previous-hop node are scheduled on link (nj) and destined to d is denoted by

$$\mu_{nj,pp}^{d} = \sum_{l:l\in\mathcal{F}} \mu_{lnj}^{d} + \sum_{l:l\in\mathcal{N}} \mu_{lnj}^{d},$$

and the total broadcast rate at which packets scheduled on link (nj) destined to d is denoted by

$$\mu_{nj,broad}^{d} = \sum_{d'} \sum_{l:l\neq j} \mu_{n|jl}^{d,d'}.$$

The total point-to-point rate on link (nj) is denoted by

$$\mu_{nj,pp} = \sum_{d} \mu_{nj,pp}^{d}$$

and the total broadcast rate at which packets are broadcast from node n to nodes j and l is denoted by

$$\mu_{n|jl} = \sum_{d'} \sum_{d} \mu_{n|jl}^{d,d'}.$$

Let $\boldsymbol{\mu}$ be the set of rates including all point-to-point transmissions and broadcast transmissions, i.e.,

$$\boldsymbol{\mu} \quad = \quad \{\{\mu_{nj,pp}\}_{(nj)}, \{\mu_{n|jl}\}_{(n|jl)}\}.$$

The multi-hop traffic should also satisfy the flow conservation constraints.

Flow conservation constraints: For each node n, each neighbor j, and each destination d, we have

$$\mu_{nj,pp}^{d} + \mu_{nj,broad}^{d} \leq \sum_{k} \mu_{njk}^{d} + \sum_{d'} \sum_{k:k\neq n} \mu_{j|kn}^{d,d'}, \qquad (18)$$

where the left-hand side denotes the total incoming traffic rate at link nj destined to d, and the right-hand side denotes the total outgoing traffic rate from link nj destined to d. For each node n and each destination d, we have

$$\sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n, e(f)=d\}} \leq \sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{N}} \mu_{fnj}^{d}, \qquad (19)$$

where I denotes the indicator function.

B. Links and Schedules

We allow broadcast transmission in our network model. In order to define a schedule, we first define two kinds of "links:" the point-to-point link and the broadcast link. A point-to-point link (nj) is a link that supports point-to-point transmission, where $(nj) \in \mathcal{L}$; A broadcast link $(n|lj)$ is a "link" which contains links (nl) and (nj) and supports broadcast transmission. Let $\mathcal{B}$ denote the set of all broadcast links, thus $(n|lj) \in \mathcal{B}$. Let $\bar{\mathcal{L}}$ be the union of the set of the point-to-point links $\mathcal{L}$ and the set of the broadcast links $\mathcal{B}$, i.e., $\bar{\mathcal{L}} = \mathcal{L} \cup \mathcal{B}$.

We let Γ' denote the set of links that can be activated simultaneously. By abusing notation, Γ' can be thought of as a set of vectors where each vector is a list of 1's or 0's where a 1 corresponds to an active link and a 0 corresponds to an inactive link. Then, the capacity region of the network for 1-hop traffic is the convex hull of all schedules, i.e., $\Lambda' = co(\Gamma')$. Thus, $\mu \in \Lambda'$.

C. Queue Structure and Shadow Queue Algorithm

Each node n maintains a set of counters, which are called *shadow queues,* p_{lnd} for each previous hop l and each destination d, and p_{0nd} for external flows destined for d at node n. Each node n also maintains a real queue, denoted by q_{lnj}, for each previous hop l and each next-hop neighbor j, and q_{0nj} for external flows with their next hop j.

By solving the optimization problem with flow conservation constraints, we can work out the back-pressure algorithm for network coding case (see the brief description in Appendix B). More specifically, for each link $(nj) \in L$ in the network and for each destination d, define the *back-pressure* at every slot to be

$$\begin{aligned} &w_{nj}^{d}[t] = \max_{l:(ln)\in\mathcal{L} \text{ or } l=0} w_{lnj}^{d}[t] \\ &\text{where } w_{lnj}^{d}[t] = p_{lnd}[t] - p_{njd}[t] - M, \\ &\text{and } l_{nj}^{*}[t] = \arg\max_{l:(ln)\in\mathcal{L} \text{ or } l=0} w_{lnj}^{d}[t]. \end{aligned} \qquad (20)$$

For each broadcast at node n to nodes j and l destined for d and d', respectively, define the *back-pressure* at every slot to be

$$w_{n|jl}^{d,d'}[t] = w_{lnj}^{d}[t] + w_{jnl}^{d'}[t]. \qquad (21)$$

The weights associated with each point-to-point link $(nj) \in \mathcal{L}$ and each broadcast link $(n|jl)$ are defined as follows

$$\begin{aligned} &w_{nj}[t] = \max_{d}\{w_{nj}^{d}[t]\}, \\ &w_{n|jl}[t] = \max_{d,d'}\{w_{n|jl}^{d,d'}[t]\}, \\ &\text{with } d_{nj}^{*}[t] = \arg\max_{d}\{w_{nj}^{d}[t]\}, \\ &\{d,d'\}_{n|jl}^{*}[t] = \arg\max_{d,d'}\{w_{n|jl}^{d,d'}[t]\}. \end{aligned} \qquad (22)$$

The rate vector $\tilde{\boldsymbol{\mu}}^{*}[t]$ at each time slot is chosen to satisfy

$$\begin{aligned} \tilde{\boldsymbol{\mu}}^{*}[t] \in \arg\max_{\tilde{\boldsymbol{\mu}}\in\Gamma'} \Big\{ &\sum_{(nj)\in\mathcal{L}} \tilde{\mu}_{nj,pp} w_{nj}[t] \\ &+ \sum_{(n|jl)\in\mathcal{B}} \tilde{\mu}_{n|jl} w_{n|jl}[t] \Big\}. \end{aligned}$$

By running the shadow queue algorithm in network coding case, we get a set of activated links in $\bar{\mathcal{L}}$ at each slot.

Next we describe the evolution of the shadow queue lengths in the network. Notice that the shadow queues at each node n are distinguished by their previous hop l and their destination d, so p_{lnd} only accepts the packets from previous hop l with destination d. The similar rule should be followed when packets are drained from the shadow queue p_{lnd}. We assume the departures occur before arrivals at each slot, and the evolution of queues is given by

$$\begin{aligned} p_{lnd}[t+1] &= \Big[p_{lnd}[t] - \sum_{j\in\mathcal{N}} \tilde{\mu}^*_{nj,pp}[t] I_{\{l=l^*_{nj}, d=d^*_{nj}\}} \\ &- \sum_{d'\in\mathcal{N}} \sum_{j\in\mathcal{N}} \tilde{\mu}^*_{n|jl}[t] I_{\{\{d,d'\}=\{d,d'\}^*_{n|jl}\}}\Big]^+ \\ &+ \sum_{k\in\mathcal{N}} \hat{\mu}^d_{kln}[t] I_{\{k=l^*_{ln}, d=d^*_{ln}\}} \\ &+ \sum_{k\in\mathcal{N}} \sum_{d'\in\mathcal{N}} \hat{\mu}^{d,d'}_{l|nk}[t] I_{\{\{d,d'\}=\{d,d'\}^*_{l|nk}\}} \\ &+ \sum_{f\in\mathcal{F}} \hat{a}_f[t] I_{\{b(f)=n, e(f)=d, l=0\}}, \end{aligned} \tag{23}$$

where $\hat{\mu}^d_{kln}[t]$ is the actual number of shadow packets scheduled over link (ln) and destined for d from the shadow queue p_{kld} at slot t, $\hat{\mu}^{d,d'}_{l|nk}[t]$ is the actual number of coded shadow packets transfered from node l to nodes n and k destined for nodes d and d' at slot t, and $\hat{a}_f$ denotes the actual number of shadow packets from external flow f received at node n destined for d.

D. Implementation Details

The implementation details of the joint adaptive routing and coding algorithm are similar to the case with adaptive routing only, but the notation is more cumbersome. We briefly describe it here.

1) Probabilistic Splitting Algorithm: The probabilistic splitting algorithm chooses the next hop of the packet based on the probabilistic routing table. Let $P^d_{lnj}[t]$ be the probability of choosing node j as the next hop once a packet destined for d receives at node n from previous hop l or from external flows, i.e., $l=0$ at slot t. Assume that $P^d_{lnj}[t]=0$ if $(nj)\notin\mathcal{L}$. Obviously, $\sum_{j\in\mathcal{N}} P^d_{lnj}[t]=1$. Let $\sigma^d_{lnj}[t]$ denote the number of potential shadow packets "transferred" from node n to node j destined for d whose previous hop is l during time slot t. Notice that the packet comes from an external flow if $l=0$. Also notice that $\sigma^d_{lnj}[t]$ is contributed by shadow traffic point-to-point transmission as well as shadow traffic broadcast transmission, i.e.,

$$\begin{aligned} \sigma^d_{lnj}[t] &= \mu^*_{nj,pp}[t] I_{\{l=l^*_{nj}[t], d=d^*_{nj}[t]\}} \\ &+ \sum_{d'\in\mathcal{N}} \mu^*_{n|jl}[t] I_{\{\{d,d'\}=\{d,d'\}^*_{n|jl}[t]\}}. \end{aligned}$$

We keep track of the the average value of $\sigma^d_{lnj}[t]$ across time by using the following updating process:

$$\hat{\sigma}^d_{lnj}[t] = (1-\beta)\hat{\sigma}^d_{lnj}[t-1] + \beta\sigma^d_{lnj}[t], \tag{24}$$

where $0\le\beta\le 1$. The splitting probability $P^d_{lnj}[t]$ is expressed as follows:

$$P^d_{lnj}[t] = \frac{\hat{\sigma}^d_{lnj}[t]}{\sum_{k\in\mathcal{N}} \hat{\sigma}^d_{lnk}[t]}. \tag{25}$$

2) Token Bucket Algorithm: At each node n, for each previous-hop neighbor l, next-hop neighbor j and each destination d, we maintain a token bucket r^d_{lnj}. At each time slot t, the token bucket is decremented by $\sigma^d_{lnj}[t]$, but cannot go below the lower bound 0 :

$$r^d_{lnj}[t] = \max\{r^d_{lnj}[t-1] - \sigma^d_{lnj}[t], 0\}.$$

When $r^d_{lnj}[t-1] - \sigma^d_{lnj}[t] < 0$, we say $\sigma^d_{lnj}[t] - r^d_{lnj}[t-1]$ tokens (associated with bucket r^d_{lnj}) are "wasted" in slot t. Upon a packet arrival from previous hop l at node n for destination d at slot t, we find the token bucket r^d_{lnj*} which has the smallest number of tokens (the minimization is over next-hop neighbors j), breaking ties arbitrarily, add the packet to the corresponding real queue q_{lnj^*}, and add one token from the corresponding bucket:

$$r^d_{lnj^*}[t] = r^d_{lnj^*}[t] + 1.$$

E. Extra link Activation

Like the case without network coding, extra link activation can reduce delays significantly. As in the case without network coding, we add additional links to the schedule based on the queue lengths at each link. For extra link activation purposes, we only consider point-to-point links and not broadcast. Thus, we schedule additional point-to-point links by giving priority to those links with larger queue backlogs.

VIII. Simulations

We consider two types of networks in our simulations: wireline and wireless. Next, we describe the topologies and simulation parameters used in our simulations, and then present our simulation results.

A. Simulation Settings

1) Wireline Setting: The network shown in Figure 4 has 31 nodes and represents the GMPLS network topology of North America [27]. Each link is assume to be able to transmit one packet in each slot. We assume that the arrival process is a Poisson process with parameter λ, and we consider the arrivals that come within a slot are considered for service at the beginning of the next slot. Once a packet arrives from an external flow at a node n, the destination is decided by probability mass function $\hat{P}_{nd}, d=1,2,...N$, where $\hat{P}_{nd}$ is the probability that a packet is received externally at node n destined for d. Obviously, $\sum_{d:d\neq n}\hat{P}_{nd}=1$, and $\hat{P}_{nn}=0$. The probability $\hat{P}_{nd}$ is calculated by

$$\hat{P}_{nd} = \frac{J_d + J_n}{\sum_{k:k\neq n}(J_k + J_n)},$$

where J_n denotes the number of neighbors of node n. Thus, we use $\hat{P}_{nd}$ to split the incoming traffic to each destination based on the degrees of the source and the destination.

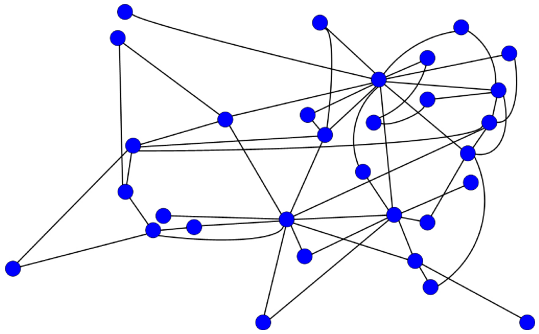


Fig. 4. Sprint GMPLS network topology of North America with 31 nodes.[27]

2) Wireless Setting: We generated a random network with 30 nodes which resulted in the topology in Figure 5. We used the following procedure to generate the random network: 30 nodes are placed uniformly at random in a unit square; then starting with a zero transmission range, the transmission range was increased till the network was connected. We assume that each link can transmit one packet per time slot. We assume a 2-hop interference model in our simulations. By a k-hop interference model, we mean a wireless network where a link activation silences all other links which are k hops from the activated link. The packet arrival processes are generated using the same method as in the wireline case. We simulate two cases given the network topology: the no coding case and the network coding case. In both wireline and wireless simulations, we chose β in (13) to be 0.02, and we use probabilistic splitting algorithm for simulations except Figure 12.

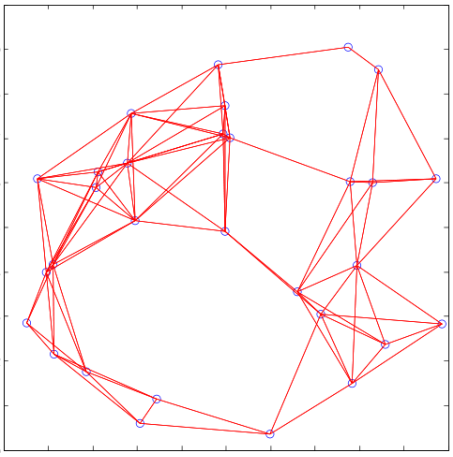


Fig. 5. Wireless network topology with 30 nodes.

B. Simulation Results

1) Wireline Networks: First, we compare the performance of three algorithms: the traditional back-pressure algorithm, the basic shadow queue routing/scheduling algorithm without the extra link activation enhancement and PARN. Without extra link activation, to ensure that the real arrival rate at each link is less than the link capacity provided by the shadow algorithm, we choose $\varepsilon = 0.02$. Figure 6 shows delay as a function of the arrival rate lambda for the three algorithms. As can be seen from the figure, simply using a value of $M > 0$ does not help to reduce delays without extra link activation. The reason is that, while $M > 0$ encourages the use of shortest paths, links with back-pressure less than M will not be scheduled and thus can contribute to additional delays. Because we exaggerate the shadow traffic by a factor of ε, the throughput region of the algorithm without extra link activation is smaller than the throughput region of the traditional back-pressure algorithm.

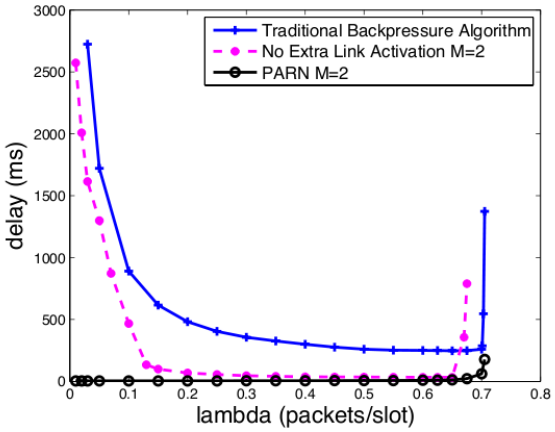


Fig. 6. The impact of the parameter M and extra link activation in Sprint GMPLS network topology.

We also compare the delay performance of PARN with that of the shortest path routing in Figure 7. For each pair of source and destination, we find a shortest path between them by using Dijkstra's algorithm. When the arrival rate $\lambda < 0.38$, the difference between the average packet delays of PARN and the shortest path routing is very small. This implies that PARN can obtain similar delay performance as the shortest path routing at light traffic. However, the shortest path routing can only achieve about 60% of the capacity region of the network.

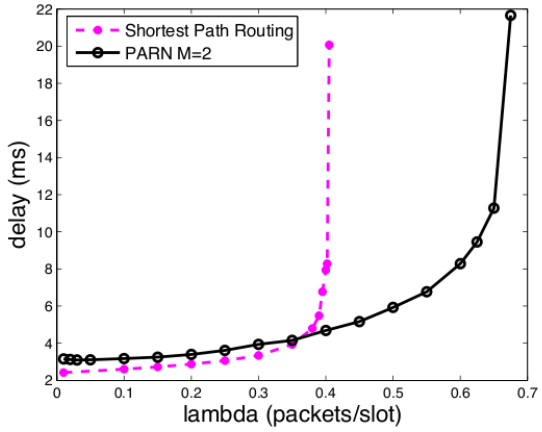


Fig. 7. The delay performance of PARN and shortest path routing.

Next, we study the impact of M on the performance on

PARN. Figure 8 shows the delay performance for various M with extra link activation in the wireline network. The delays for different values of M (except $M = 0$) are almost the same in the light traffic region. Once M is sufficiently larger than zero, extra link activation seems to play a bigger role, than the choice of the value of M, in reducing the average delays.

The wireline simulations show the usefulness of the PARN algorithm for adaptive routing. However, a wireline network does not capture the scheduling aspects inherent to wireless networks, which is studied next.

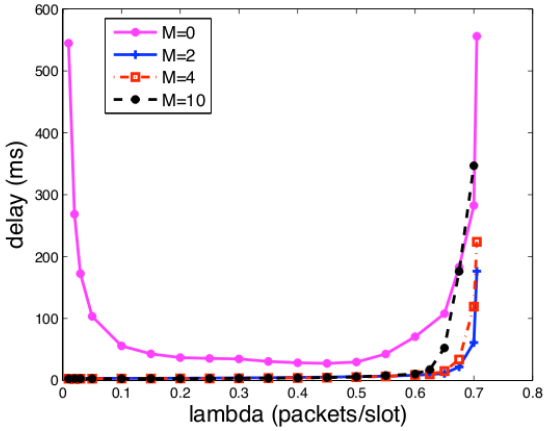


Fig. 8. Packet delay as a function of λ under PARN in Sprint GMPLS network topology.

2) Wireless Networks: In the case of wireless networks, even with extra link activation, to ensure stability even when the arrival rates are within the capacity region, we need $\varepsilon > 0$. We chose $\varepsilon = 0.1$ in our simulations due to reasons mentioned in Section VI.

In Figure 9, we study wireless networks without network coding. From the figure, we see that the delay performance is relatively insensitive to the choice of M as long as it is sufficiently greater than zero. However, M does play an important role because it suppresses the search of long paths when the traffic load is not high. Extra link activation can be used to decrease delays significantly for $M > 0$ especially in light traffic region.

In Figures 10 and 11, we show the corresponding results for the case where both adaptive routing and network coding are used. Comparing Figures 9 and 10, we see that, when used in conjunction with adaptive routing, network coding can increase the capacity region. We make the following observation regarding the case $M = 0$ in Figure 11: in this case, no attempt is made to optimize routing in the network. As a result, the delay performance is very bad compared to the cases with $M > 0$ (Figure 10). In other words, network coding alone does not increase capacity sufficiently to overcome the effects of back-pressure routing. On the other hand, PARN with $M > 0$ harnesses the power of network coding by selecting routes appropriately.

Next, we make the following observation about network coding. Comparing Figures 10 and 11, we noticed that at moderate to high loads (but when the load is within the capacity

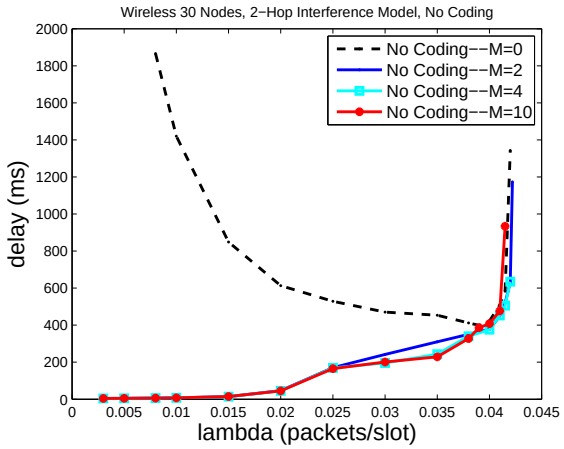


Fig. 9. Packet delay as a function of λ under PARN in the wireless network under 2-hop interference model without network coding.

region of the no coding case), network coding increases delays slightly. We believe that this is due to fact that packets are stored in multiple queues under network coding at each node: for each next-hop neighbor, a queue for each previous-hop neighbor must be maintained. This seems to result in slower convergence of the routing table.

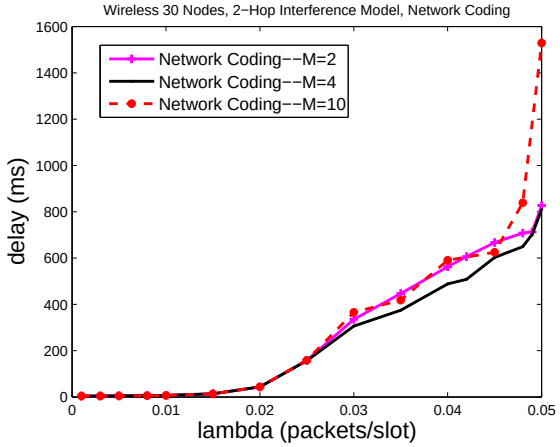


Fig. 10. Packet delay as a function of λ under PARN for $M > 0$ in the wireless network under 2-hop interference model with network coding.

Finally, we study the performance of the probabilistic splitting algorithm versus the token bucket algorithm. In our simulations, the token bucket algorithm runs significantly faster, by a factor of 2. The reason is that many more calculations are needed for the probabilistic splitting algorithm as compared to the token bucket algorithm. This may have some implications for practice. So, in Figure 12, we compare the delay performance of the two algorithms. As can be seen from the figure, the token bucket and probabilistic splitting algorithms result in similar performance. Therefore, in practice, the token bucket algorithm may be preferable.

IX. Conclusion

The back-pressure algorithm, while being throughput-optimal, is not useful in practice for adaptive routing since the

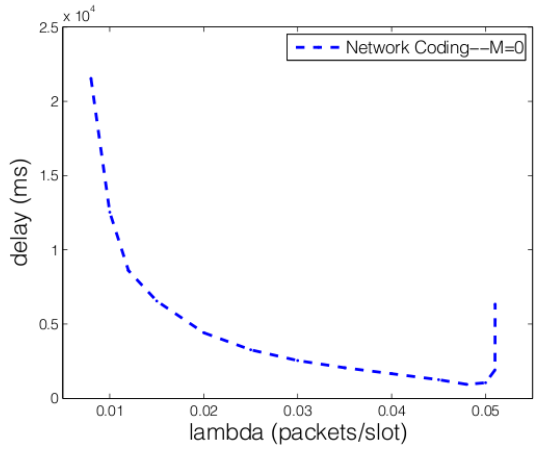


Fig. 11. Packet delay as a function of λ under PARN for $M = 0$ in the wireless network under 2-hop interference model with network coding.

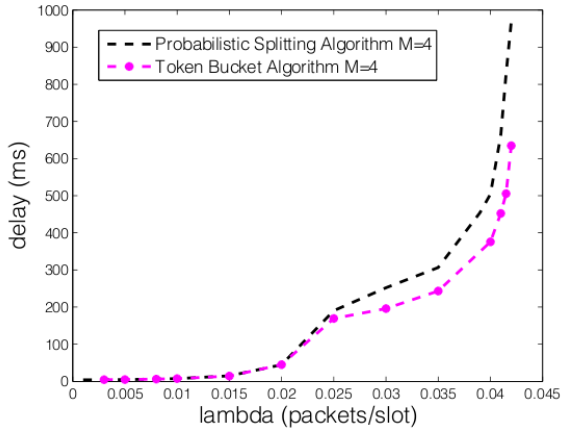


Fig. 12. Comparison of probabilistic splitting and token bucket algorithms under PARN in the wireless network under 2-hop interference model without network coding.

delay performance can be really bad. In this paper, we have presented an algorithm that routes packets on shortest hops when possible, and decouples routing and scheduling using a probabilistic splitting algorithm built on the concept of shadow queues introduced in [5], [3]. By maintaining a probabilistic routing table that changes slowly over time, real packets do not have to explore long paths to improve throughput, this functionality is performed by the shadow "packets." Our algorithm also allows extra link activation to reduce delays. The algorithm has also been shown to reduce the queueing complexity at each node and can be extended to optimally trade off between routing and network coding.

Appendix A
The Stability of the Network Under PARN

Our stability result uses the result in [26] and relies on the fact that the arrival rate on each link is less than the available capacity of the link.

We will now focus on the case of wireless networks without network coding.

All variables in this appendix are assumed to be average values in the stationary regime of the corresponding variables in the shadow process. Let $\bar{\sigma}^d_{nj}$ denote the mean shadow traffic rate at link (nj) destined to d. Let $\bar{\mu}_{nj}$ and $\alpha^d_n(1+\varepsilon)$ denote the mean service rate of link (nj) and the exogenous shadow traffic arrival rate destined to d at node n. Notice that ε comes from our strategy on shadow traffic. The flow conservation equation is as follows:

$$\alpha^d_n(1+\varepsilon) + \sum_{l:(ln)\in\mathcal{L}} \bar{\sigma}^d_{ln} = \sum_{j:(nj)\in\mathcal{L}} \bar{\sigma}^d_{nj}, \forall n, d \in \mathcal{N}. \quad (26)$$

The necessary condition on the stability of shadow queues are as follows:

$$\sum_{d\in\mathcal{N}} \bar{\sigma}^d_{nj} \le \bar{\mu}_{nj}. \quad (27)$$

Since we know that the shadow queues are stable under the shadow queue algorithm, the expression (27) should be satisfied.

Now we focus on the real traffic. Suppose the system has an equilibrium distribution and let λ^d_{nj} be the mean arrival rate of real traffic at link (nj) destined to d. The splitting probabilities are expressed as follows:

$$\bar{P}^d_{nj} = \frac{\bar{\sigma}^d_{nj}}{\sum_{k\in\mathcal{N}} \bar{\sigma}^d_{nk}}, \text{where} \quad d \neq n. \quad (28)$$

Thus, the mean arrival rates at a link satisfy traffic equation:

$$\lambda^d_{nj} = \alpha^d_n \bar{P}^d_{nj} + \sum_{l:(ln)\in\mathcal{L}} \lambda^d_{ln} \bar{P}^d_{nj}, \forall (nj) \in \mathcal{L}, d \in \mathcal{N}, \quad (29)$$

where $d \neq n$.

The traffic intensity at link (nj) is expressed as:

$$\rho_{nj} = \frac{1}{\bar{\mu}_{nj}} \sum_{d\in\mathcal{N}} \lambda^d_{nj}. \quad (30)$$

Now we will show $\rho_{nj} < 1$ for any link $(nj) \in \mathcal{L}$. Let $\lambda^d_{nj} = \bar{\sigma}^d_{nj}/(1+\varepsilon)$ for every $(nj) \in \mathcal{L}$, and substitute it into expression (29). It is easy to check that the candidate solution is valid by using expression (26). From (27), the traffic intensity at link (nj) is strictly less than 1 for any link $(nj) \in \mathcal{L}$:

$$\rho_{nj} = \frac{1}{\bar{\mu}_{nj}} \sum_{d\in\mathcal{N}} \lambda^d_{nj} = \frac{1}{(1+\varepsilon)\bar{\mu}_{nj}} \sum_{d\in\mathcal{N}} \bar{\sigma}^d_{nj} < 1. \quad (31)$$

Thus we have shown that the traffic intensity at each link is strictly less than 1.

The wireline network is a special case of a wireless network. Substitute the link capacity c_{nj} for $\bar{\mu}_{nj}$ and set ε to be zero, and stability follows directly.

The stability of wireless networks with network coding is similar to the case of wireless network with no coding.

Appendix B
The Back-pressure Algorithm in the Network Coding Case

Given a set of packet arrival rates that lie in the capacity region, our goal is to find routes for flows that use as

few resources as possible. Thus, we formulate the following optimization problem for the network coding case.

$$\begin{aligned} \min \quad & \sum_{(nj)\in\bar{\mathcal{L}}} \mu_{nj,pp} + \sum_{(n|jl)\in\bar{\mathcal{L}}} \mu_{(n|jl)} \qquad (32)\\ s.t. \quad & \mu^d_{nj,pp} + \mu^d_{nj,broad} \le \sum_k \mu^d_{njk} + \sum_{d'} \sum_{k:k\neq n} \mu^{d,d'}_{j|kn} \\ & \sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}} \le \sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{N}} \mu^d_{fnj} \end{aligned}$$

Let $\{q_{njd}\}$ and $\{q_{0nd}\}$ be the Lagrange multipliers corresponding to the flow conservation constraints in problem (32). Appending the constraints to the objective, we get

$$\begin{aligned} & \min_{\boldsymbol{\mu}\in\Lambda'} \sum_{(nj)\in\bar{\mathcal{L}}} \mu_{nj,pp} + \sum_{(n|jl)\in\bar{\mathcal{L}}} \mu_{n|jl} + \sum_d \sum_{(nj)\in\bar{\mathcal{L}}} q_{njd}\Big[\\ & \mu^d_{nj,pp} + \mu^d_{nj,broad} - \sum_k \mu^d_{njk} - \sum_{d'} \sum_{k:k\neq n} \mu^{d,d'}_{j|kn}\Big] \qquad (33)\\ & + \sum_{n,d} q_{0nd}\Big[\sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}} - \sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{N}} \mu^d_{fnj}\Big] \\ & = \min_{\boldsymbol{\mu}\in\Lambda'}\Big(- \sum_{(nj)\in\bar{\mathcal{L}}} \sum_{l:(ln)\in\bar{\mathcal{L}}} \sum_d \mu^d_{lnj}\big(q_{lnd} - q_{njd} - 1\big) \\ & - \sum_{(n|jl)\in\bar{\mathcal{L}}, j<l} \sum_{d,d'} \mu^{d,d'}_{n|jl}\big(q_{lnd} - q_{njd} + q_{jnd'} - q_{nld'} - 2\big) \\ & - \sum_{(nj)\in\bar{\mathcal{L}}} \sum_d \sum_{f\in\mathcal{F}} \mu^d_{fnj}\big(q_{0nd} - q_{njd} - 1\big) \\ & + \sum_{n,d} q_{0nd} \sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}}\Big). \end{aligned}$$

If the Lagrange multipliers are known, then the optimal $\boldsymbol{\mu}$ can be found by solving

$$\max_{\boldsymbol{\mu}\in\Lambda'} \sum_{(nj)\in\bar{\mathcal{L}}} \mu_{nj,pp} w_{nj} + \sum_{(n|jl)\in\bar{\mathcal{L}}, j<l} \mu_{n|jl} w_{n|jl} \qquad (34)$$

where

$$\begin{aligned} w_{nj} &= \max_d \{w^d_{nj}\},\\ w_{n|jl} &= \max_{d,d'} \{w^{d,d'}_{n|jl}\},\\ w^{d,d'}_{n|jl} &= w^d_{lnj} + w^{d'}_{jnl}\\ w^d_{nj} &= \max_{l:(ln)\in\mathcal{L} \text{ or } l=0} w^d_{lnj}\\ w^d_{lnj} &= q_{lnd} - q_{njd} - 1. \end{aligned}$$

Similar to the update algorithm of q_{nd} in (7), we can derive the update algorithm to compute q_{njd} :

$$\begin{aligned} q_{njd}[t+1] = \Big[& q_{njd}[t] + \frac{1}{M}\big(\mu^d_{nj,pp} + \mu^d_{nj,broad} \\ & - \sum_k \mu^d_{njk} - \sum_{d'} \sum_{k:k\neq n} \mu^{d,d'}_{j|kn}\big) \qquad (35)\\ & + \frac{1}{M}\big(\sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}} - \sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{N}} \mu^d_{fnj}\big)\Big]^+ \end{aligned}$$

By choosing $\frac{1}{M}$ to be the step-size parameter, Mq_{njd} looks very much like a queue update equation. Replacing Mq_{njd} by p_{njd}, we get (20)-(23). It can be shown using the results in [21], [22] that the stochastic version of the above equations are stable and that the average rates can approximate the solution to (32) arbitrarily closely.

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*, pp. 1936–1948, December 1992.

[2] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, January 2005.

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

[4] ——, "A novel architecture for delay reduction in the back-pressure scheduling algorithm," *IEEE/ACM Trans. Networking*, vol. 19, no. 6, pp. 1597–1609, December 2011.

[5] ——, "Optimal resource allocation for multicast flows in multihop wireless networks," *Philosophical Transactions of the Royal Society, Ser. A*, vol. 366, pp. 2059–2074, 2008.

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

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *ACM SIGCOMM Computer Communication Review*, vol. 36, 2006, pp. 243–254.

[8] M. Effros, T. Ho, and S. Kim, "A tiling approach to network code design for wireless networks," in *Information Theory Workshop*, 2006.

[9] H.Seferoglu, A.Markopoulou, and U.Kozat, "Network coding-aware rate control and scheduling in wireless networks," in *Special Session on "Network Coding for Multimedia Streaming", ICME*, Cancun, Mexico, June 2009.

[10] S. B. S. Sengupta, S. Rayanchu, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," in *Proc. IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[11] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *IEEE Transactions on Information Theory*, February 2009.

[12] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," in *Proceedings of the Workshop on Network Coding, Theory and Applications (NetCod)*, January 2007.

[13] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," in *Proc. IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[14] B. Awerbuch and T. Leighton, "A simple local-control approximation algorithm for multicommodity flow," in *Proc. 34th Annual Symposium on the Foundations of Computer Science*, 1993.

[15] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Advances in Applied Probability*, June 2006.

[16] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," in *Proc. IEEE INFOCOM*, 2008.

[17] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks - a partitioning approach," in *Proc. ACM Mobicom*, Sep. 2006.

[18] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficient of greedy maximal scheduling in wireless networks," in *Proc. ACM MobiHoc*, 2009.

[19] B. Li, C. Boyaci, and Y. Xia, "A refined performance characterization of longest-queue-first policy in wireless networks," in *Proc. ACM MobiHoc*, 2009.

[20] X. Lin and N. Shroff, "On the stability region of congestion control," in *Proceedings of the Allerton Conference on Communications, Control and Computing*, 2004.

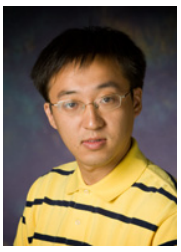
[21] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proceedings of IEEE INFOCOM*, 2005.

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

[23] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," in *Proceedings of IEEE INFOCOM*, 2005, revised version to appear in *IEEE/ACM Transactions on Networking*.
[24] ——, "Joint congestion control, routing and mac for stability and fairness in wireless networks," in *Proc. International Zurich Seminar on Communications*, 2006.
[25] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, 2006.
[26] M. Bramson, "Convergence to equilbria for fluid models of FIFO queueing networks," *Queueing Systems: Theory and Applications*, vol. 22, pp. 5–45, 1996.
[27] "Sprint IP network performance," available at https://www.sprint.net/performance/.

Eleftheria Athanasopoulou (M'02) received her Diploma degree in Electrical and Computer Engineering from the University of Patras in 2000 and her M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2002 and 2007, respectively. She was then a post-doctoral research associate at the University of Illinois at Urbana-Champaign and the Coordinated Science Lab. Her research interests include wireline and wireless communication networks, stochastic models, discrete event systems, and failure diagnosis of dynamic systems and networks.

Loc X. Bui received the B.Eng. degree in Electronics and Telecommunications from the Posts and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam, in 2003, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the 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 Senior Software Engineer. and then a Senior Sustaining Engineer. From April 2010 to September 2011, he was a Postdoctoral Scholar in the Department of Management Science and Engineering, Stanford University. From October 2011 to January 2012, he was a Visiting Fellow in the Department of Electrical Engineering, Technion - Israel Institute of Technology. He is currently a Lecturer of Electrical Engineering in the School of Engineering, Tan Tao University. His research interests include communication networks, wireless communications, game theory, and machine learning.



Tianxiong Ji (M'07) received his B.Eng. and M.S. degrees in Electrical Engineering from Tsinghua University, Beijing, China in 2005 and 2007, respectively and his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2012. He has been working at Google Inc. since December 2011. His research interests include wireless networks, queuing theory, data center networks, and wireless communication.

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 and service systems. He is an associate editor of Operations Research; Queueing Systems - Theory and Applications; and Advances in Applied Probability.