Open access • Proceedings Article • DOI:10.1109/INFCOM.2009.5062086

On Combining Shortest-Path and Back-Pressure Routing Over Multihop Wireless Networks — Source link

Lei Ying, Sanjay Shakkottai, Reddy Akula

Institutions: Iowa State University, University of Texas at Austin

Published on: 19 Apr 2009 - International Conference on Computer Communications

Topics: Equal-cost multi-path routing, Static routing, Geographic routing, Multipath routing and DSRFLOW

On Combining Shortest-Path and Back-Pressure Routing Over Multihop Wireless Networks

Lei Ying
ECE Department
Iowa State University
Email: leiying@iastate.edu

Sanjay Shakkottai and Aneesh Reddy
ECE Department
The University of Texas at Austin
Email: {shakkott, areddy}@ece.utexas.edu

***Abstract*—Back-pressure based algorithms based on the algorithm by Tassiulas and Ephremides have recently received much attention for jointly routing and scheduling over multihop wireless networks. However a significant weakness of this approach has been in routing, because the traditional back-pressure algorithm explores and exploits all feasible paths between each source and destination. While this extensive exploration is essential in order to maintain stability when the network is heavily loaded, under light or moderate loads, packets may be sent over unnecessarily long routes and the algorithm could be very inefficient in terms of end-to-end delay and routing convergence times.**

This paper proposes new routing/scheduling back-pressure algorithms that not only guarantees network stability (throughput optimality), but also adaptively selects a set of optimal routes based on *shortest-path information* in order to minimize average path-lengths between each source and destination pair. Our results indicate that under the traditional back-pressure algorithm, the end-to-end packet delay first decreases and then increases as a function of the network load (arrival rate). This surprising low-load behavior is explained due to the fact that the traditional back-pressure algorithm exploits all paths (including very long ones) even when the traffic load is light. On the other-hand, the proposed algorithm adaptively selects a set of routes according to the traffic load so that long paths are used only when necessary, thus resulting in much smaller end-to-end packet delays as compared to the traditional back-pressure algorithm.

I. Introduction

Due to the scarcity of wireless bandwidth resources, it is important to efficiently utilize resources to support high-throughput, high-quality communications over multi-hop wireless networks. In this context, good routing and scheduling algorithms are needed to dynamically allocate wireless resources to maximize the network throughput region. To address this, throughput-optimal[1] routing and scheduling, first developed in the seminal work of [1], has been extensively studied [2], [3], [4], [5], [6], [7], [8], [9], [12], [13], [14], [15]. We refer to [10], [11] for a comprehensive survey. While these algorithms maximize the network throughput region, additional issues need to be considered for practical deployment.

With the significant increase of real-time traffic (an article by Ellacoya [16] published in 2007 suggests that video-streaming accounts for 36% of today's HTTP traffic), end-to-end delay becomes very important in network algorithm design. The traditional back-pressure algorithms stabilize the network by exploiting all possible paths between source-destination pairs (thus load balancing over the entire network). While this might be needed in a heavily loaded network, this seems unnecessary in a light or moderate load regime. Exploring all paths is in fact detrimental – it leads to packets traversing excessively long paths between sources and destinations leading to large end-to-end packet delays.

This paper proposes a new routing/scheduling back-pressure algorithm that minimizes the path-lengths between sources and destinations while simultaneously being overall throughput-optimal. The proposed algorithm results in much smaller end-to-end packet delay as compared to the traditional back-pressure algorithm. The main contributions of this paper are summarized in the following subsection.

A. Main Contributions

We define a flow using its source and destination. Let f denote a flow in network, and $A_f[t]$ denote the number of packets generated by flow f at time t. We first consider the case where each flow associates with a hop constraint H_f. The routing and scheduling algorithm needs to guarantee that the packets from flow f are delivered no more than H_f hops. Note that this hop constraint is closely related to the end-to-end propagation delay. For this problem, we propose a shortest-path-aided back-pressure algorithm which exploits the shortest-path information to guarantee the hop constraint and is throughput optimal, i.e., if there exists a routing/scheduling algorithm that can support the traffic with the given hop constraints, then the shortest-path-aided back-pressure can support the traffic as well.

We then consider a case where no per-flow hop constraint is imposed. The objective is to minimize the average number of hops per packet delivery (or the average path-lengths between sources and destinations). Mathematically, given a traffic load $\{A_f[t]\}$, the objective is

$$\min \sum_{f\in\mathcal{F}, N-1\geq h>0} hA_{f,h},$$

where $A_{f,h}$ is the fraction of flow f transmitted over paths with h hops, and $\sum_h A_{f,h} = \mathbf{E}[A_f[t]]$. This objective has two interpretations:

[1]A routing/scheduling algorithm is throughput-optimal if it can stabilize any traffic that can be stabilized by any other routing/scheduling algorithm.

- First, $\sum_{f,h} hA_{f,h}$ can be thought of as the number of transmissions needed to support traffic $\mathbf{A}[t]$ (transmitting a packet over an h-hop path requires h transmissions). Thus, minimizing $\sum_{f,h} hA_{f,h}$ can be regarded as minimizing the network resource used to support the traffic demand;
- Second, note that the number of hops is closely related to the end-to-end delay, so $\sum_h hA_{f,h}$ is related to the average end-to-end delay of flow f. Thus minimizing $\sum_{f,h} hA_{f,h}$ can potentially be used as a surrogate for minimizing the average end-to-end delay over all flows in the network (the difference being that the MAC delays is ignored in the hop-count metric).

To solve this problem, we propose a joint traffic-control and shortest-path-aided back-pressure algorithm that not only guarantees the network stability (throughput-optimal), but also adaptively selects the optimal routes according to the traffic demand. When the traffic is light, the algorithm only uses shortest paths; when the traffic increases, more paths are exploited to support the traffic. Our simulations show that the joint traffic-control and shortest-path-aided back-pressure algorithm leads to a much smaller end-to-end delay compared to the traditional back-pressure algorithm (3 vs 1000 when the traffic load is light and 200 vs 400 when the traffic load is high).

B. Related Work

Throughput-optimal routing/scheduling was first proposed in [1], and then have been studied for varied networks including cellular networks [17], cooperative relay networks [12], [13], and multi-hop wireless networks [6], [4], [5]. Low-complexity implementations have been proposed in [18], [19], [20], [21], [22], [14], [23], [24]. Joint scheduling/routing/power control has been developed in [5], [9]. Throughput-optimal routing/scheduling for multicast flows has been considered in [25]. The idea of using the shortest path information to enhance the performance of back-pressure algorithm has been studied in [26]. The difference is that the proposed algorithm *provably* minimizes the average path-lengths whereas the enhanced algorithm in [26] uses the shortest path information in a heuristic manner. An alternate algorithm that deals with minimizing the number of hops has been recently independently obtained in [30]. The objective function in [30] is the same as in this paper, however the proposed algorithms are different.

II. An Illustrative Example

As we discussed in the introduction, the back-pressure algorithm exploits all feasible paths, which is critical to maintain stability when the network is heavily loaded. However, when the traffic load is light, packets may be sent over unnecessary long paths and the algorithm could be very inefficient.

In this section, we use an example to demonstrate the weakness of the back-pressure algorithm, and the significant end-to-end delay reduction that results under the proposed algorithm (the algorithm will be described in Section V).

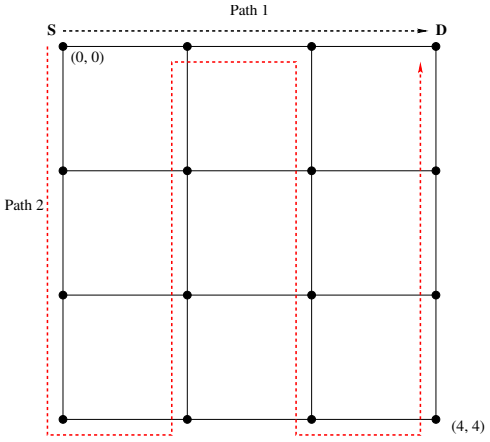

Fig. 1. A grid network example

Consider a 4×4 grid network as in shown Figure 1. Assume that the channel capacity is one data unit per time slot for all channels. When one link is on, no adjacent link can be on simultaneously. We also impose half-duplex constraint so that a node cannot transmit and receive at the same time. At the beginning of each time slot, each node generates a packet with probability λ. The destination of this packet is randomly and uniformly selected from all nodes in the network. (A detailed description of our simulation settings will be presented in Section VI).

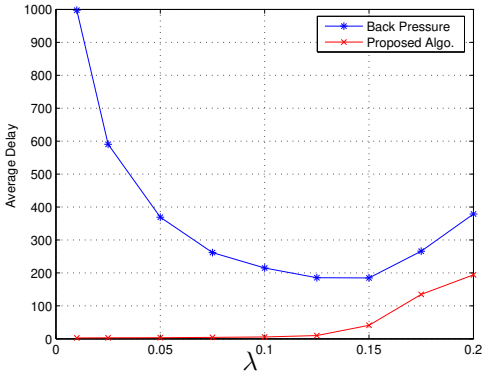

Fig. 2. Back-pressure via our joint traffic-splitting and shortest-path-aided back-pressure

The end-to-end delay of a packet is defined to be the time interval from when the packet enters the source to when the packet reaches the destination (this includes the MAC delay at intermediate nodes). In Figure 2, we plot the average end-to-end delay under the back-pressure algorithm and the proposed algorithm with different values of λ. From Figure 2, we have two observations:

(1) Under the back-pressure algorithm, surprisingly, the delay first decreases and then increases with arrival rate λ. The second part is easy to understand: the queues build up when the traffic load increases, which increases the queuing delays. The first part is because the back-

pressure algorithm uses all paths even when the traffic load is light. For example, with a very small λ, using path 1 only is sufficient to support the flow from node S to node D. However, under the back-pressure algorithm, long paths (e.g., path 2 that has 15 hops) and paths with loops are also used. Furthermore, the lighter is the traffic load, more loops are involved in the route. Hence the end-to-end delay is large when λ is small.

(2) In the proposed algorithm, the set of routes used are intelligently selected according to the traffic load so that long paths are used only when necessary. We can see that under the proposed algorithm, not only is the delay significantly reduced (3 *v.s* 1000), but also the delay monotonically increases with the traffic load.

We would like to emphasize that under the proposed algorithm, the delay improvement is achieved without losing the throughput-optimality. The proposed algorithm is still throughput-optimal, but yields much smaller end-to-end delays as compared to the traditional back-pressure algorithm.

III. Basic Model

Network model: Consider a network represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of directed links. We assume that $|\mathcal{N}| = N$ and $|\mathcal{L}| = L$.

Denote by (m, n) the link from node m to node n. Let $\boldsymbol{\mu} = \{\mu_{(m,n)}\}$ denote a link-rate vector (over link (m, n), the transmission rate is $\mu_{(m,n)}$). A link-rate vector $\boldsymbol{\mu}$ is said to be *admissible* if the link-rates specified by $\boldsymbol{\mu}$ can be achieved *simultaneously*. Define Γ to be the set of all admissible link-rate vectors. It is easy to see that Γ depends on the choice of interference model and might not be a convex set. Furthermore, Γ is time-varying if channels are time-varying. To simplify our notations, we assume time-invariant channels in this paper. However, our results can be extended to time-varying channels in a straightforward manner. Furthermore, we assume that there exists $\mu_{\max}$ such that $\mu_{(m,n)} \leq \mu_{\max}$ for all $(m, n) \in \mathcal{L}$ and all admissible $\boldsymbol{\mu}$. Next, we define a link vector $\boldsymbol{\mu}$ to be *obtainable* if $\boldsymbol{\mu} \in \mathcal{CH}(\Gamma)$, where $\mathcal{CH}(\Gamma)$ denotes the convex hull of Γ. Note that an *admissible* rate-vector is a set of rates at which the links can transmit simultaneously; while an *obtainable* rate-vector is a set of rates that can be achieved including using time-sharing.

Traffic model: For network traffic, we let f denote a flow, $s(f)$ denote the source of the flow, and $d(f)$ the destination of the flow. We use $\mathcal{F}$ to denote the set of all flows in the network. Assume that time is discretized, and let $A_f[t]$ ($f \in \mathcal{F}$) denote the number of packets injected by flow f at time t. We assume $\{A_f[t]\}$ are bounded random variables, and i.i.d. across time-slots and flows. We also define $A_f = \mathbf{E}[A_f[t]]$.

IV. Throughput-Optimal Routing/Scheduling with Hop Constraints

In this section, we consider the case where each flow is associated with a hop constraint H_f. Packets of flow f need to be delivered within H_f hops. We propose a shortest-path-aided back-pressure algorithm, which is throughput-optimal under hop-constraints. The algorithm is also a building block for the algorithm to be proposed in Section V, which smoothly integrates the back-pressure and the shortest-path routing.

Next, we characterize the network throughput region under hop-constraints.

A. Network Throughput Region under Hop-constraints

Given traffic $\mathbf{A}[t] = \{A_f[t]\}_{f\in\mathcal{F}}$ and hop-constraint $\mathbf{H} = \{H_f\}_{f\in\mathcal{F}}$, we say that $(\mathbf{A}[t], \mathbf{H}) \in \Lambda_{\mathcal{G}}$ if there exists $\left\{\hat{\mu}_{(m,n)}^{(n,d,h)} \geq 0\right\}$ such that the following conditions hold:

(i) For any three-tuple (n, d, h) such that $n \neq d$ and $N - 1 \geq h > 0$, we have

$$A_f 1_{\substack{s(f)=n,\, d(f)=d \\ H_f = h}} + \sum_{m:(m,n)\in\mathcal{L}} \hat{\mu}_{(m,n)}^{(m,d,h+1)} = \sum_{i:(n,i)\in\mathcal{L}} \hat{\mu}_{(n,i)}^{(n,d,h)}. \quad (1)$$

(ii) If $h - 1 < H_{n\to d}^{\min}$, then

$$\hat{\mu}_{(m,n)}^{(m,d,h)} = 0, \quad (2)$$

where $H_{n\to d}^{\min}$ is the minimum number of hops required from node n to node d.

(iii)

$$\left\{\hat{\mu}_{(m,n)}\right\}_{(m,n)\in\mathcal{L}} \in \mathcal{CH}(\Gamma), \quad (3)$$

where

$$\hat{\mu}_{(m,n)} = \sum_{\{(m,d,h):d\in\mathcal{D}, N-1\geq h>0\}} \hat{\mu}_{(m,n)}^{(m,d,h)},$$

and $\mathcal{D}$ is the set of all destinations.

We can regard $\hat{\mu}_{(m,n)}^{(m,d,h)}$ as the average transmission-rate over link (m, n) used to transmit those packets that are destined to node d and delivered with exactly h more hops (including the hop from m to n). Then, the conditions above can be explained as follows:

(a) Condition (i) is a flow conservation constraint, which states that the number of incoming packets to node n with hop-constraint h is equal to the number of outgoing packets from node n with hop-constraint $h-1$. Note that the hop-constraint is reduced by one after a packet is sent out by node n because it takes one hop to transmit the packet from node n to one of its neighbors. We only consider hop-constraints up to $N - 1$ hops because the longest loop-free route has no more than $N - 1$ hops, and considering only loop-free routes does not change the network throughput region.

(b) Condition (ii) states that a packet should not be transmitted from node m to node n if node n cannot deliver the packet within the required number of hops.

(c) Condition (iii) is the capacity constraint, which states that the rate-vector $\hat{\boldsymbol{\mu}}$ should be obtainable.

We say traffic $(\mathbf{A}[t], \mathbf{H})$ can be stabilized if *there exists some routing/scheduling algorithm under which the mean of the number of packets queued in the network is bounded.*

From discussions (a)-(c), it is easy to see that if $(\mathbf{A}[t], \mathbf{H})$ can be stabilized, then there must exist $\hat{\boldsymbol{\mu}}$ satisfying conditions (i)-(iii). *Thus, $\Lambda_{\mathcal{G}}$ is named as the the throughput region of network $\mathcal{G}$.*

Next, we introduce our queue management scheme.

B. Queue Management

Recall $H^{\min}_{m\to d}$ is the minimum number of hops required from node m to node d (or the length of the shortest path from node m to node d). Note that $H^{\min}_{m\to d}$ can be computed in a distributed fashion using algorithms such as the Bellman-Ford algorithm. Thus, we assume that node m knows $H^{\min}_{m\to d}$ for all destinations $d \in \mathcal{D}$, and $H^{\min}_{n\to d}$ for n such that $(m,n) \in \mathcal{L}$.

We assume node m maintains a separate queue, named queue $\{m, d, h\}$, for the packets required to be delivered to node d within h hops. For destination d, node m maintains queues for $h = H^{\min}_{m\to d}, \ldots, N-1$, where $N-1$ is a universal upper bound on the number of hops along loop-free paths.

As an example, consider the directed network shown in Figure 3, and assume that $\mathcal{D} = \{4\}$ (i.e., there is only one destination). Each non-destination node maintains up to three queues (because for this topology, there are no loop free paths longer than three hops). Node 1 has queues corresponding to $h = 1, 2, 3$ respectively. Node two does not have a direct path to node 4 (i.e., $H^{\min}_{2\to 4} = 2$), hence, it maintains only two queues corresponding to $h = 2, 3$ (and implicitly, we set $Q_{\{2,4,1\}} = \infty$ to ensure that no packets enter $Q_{\{2,4,1\}}$ from other nodes). Node 3 maintains three separate queues corresponding to $h = 1, 2, 3$. This is in spite of the observation that there is only one feasible route from node 3 to node 4. We maintain these additional queues because the global network topology is not known by individual nodes (in the algorithm, we will later see that the "extra" queues will build up sufficient back-pressure so that the rate of packet arrivals into these queues goes to zero). Finally, all queues at the destination for packets meant to itself are set to zero (i.e., $Q_{\{4,4,h\}} = 0$). In Figure 3, queues into which packets potentially arrive are marked in solid lines and the "virtual" queues which are fixed at $\{0, \infty\}$ are in dotted lines.

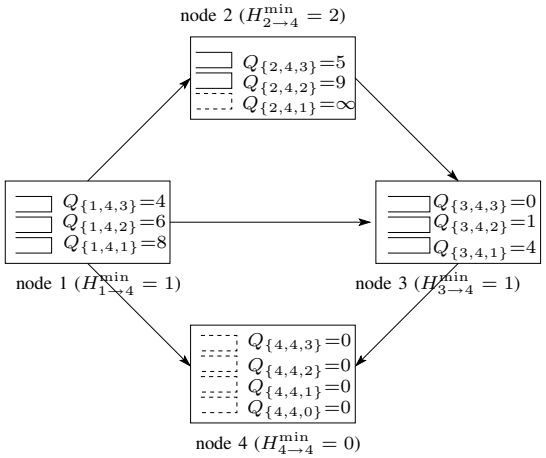

Fig. 3. Illustration of queue-management and computation of back-pressure

C. Queue Dynamics

Let $Q_{\{m,d,h\}}[t]$ denote the queue length at time slot t, and $\mu^{\{m,d,h\}}_{(m,n)}[t]$ denote the service rate for queue $\{m, d, h\}$ over link (m, n) at time t. *For packets transmitted over link (m, n), we require that the packets from queue $\{m, d, h\}$ are transferred to queue $\{n, d, h-1\}$.* For example, packets from queue $\{2, 4, 3\}$ can be transferred to queue $\{3, 4, 2\}$, but not to queue $\{3, 4, 1\}$.

The dynamics of queue $\{n, d, h\}$ $(n \neq d)$ is as follows:

$$Q_{\{n,d,h\}}[t+1] = Q_{\{n,d,h\}}[t] + A_f[t]1_{s(f)=n,d(f)=d,H_f=h} + \sum_{m:(m,n)\in\mathcal{L}} \nu^{\{m,d,h+1\}}_{(m,n)}[t] - \sum_{i:(n,i)\in\mathcal{L}} \nu^{\{n,d,h\}}_{(n,i)}[t],$$

where $\nu^{\{n,d,h\}}_{(n,i)}$ is the actual number of packets transferred from queue $\{n, d, h\}$ to queue $\{i, d, h-1\}$, and is smaller than $\mu^{\{n,d,h\}}_{(n,i)}[t]$ when there is no enough packet in queue $\{n, d, h\}$. Define $u^{\{m,d,h\}}_{(m,n)}[t]$ to be the unused service, we have

$$\nu^{\{m,d,h\}}_{(m,n)}[t] = \mu^{\{m,d,h\}}_{(m,n)}[t] - u^{\{m,d,h\}}_{(m,n)}[t].$$

We also define $Q_{\{n,n,h\}} = 0$ for all h, i.e., packets delivered are removed from the network.

In the next subsection, we propose a shortest-path-aided back-pressure algorithm that stabilizes the network given any $(\mathbf{A}[t], \mathbf{H}) \in \Lambda_{\mathcal{G}}$.

D. Shortest-path-aided Back-pressure Algorithm

Recall that we have per-hop queues for each destination, which is different from the back-pressure algorithm in [1]. Thus we first define the back-pressure of link (m, n) under our queue management scheme. We define $P^{\{m,d,h\}}_{(m,n)}[t]$ (the back-pressure of queue $\{m, d, h\}$ over link (m, n)) as follows:

- $P^{\{m,d,h\}}_{(m,n)}[t] = Q_{\{m,d,h\}}[t] - Q_{\{n,d,h-1\}}[t]$ if $H^{\min}_{n\to d} \leq h-1$;
- $P^{\{m,d,h\}}_{(m,n)}[t] = -\infty$ if $H^{\min}_{n\to d} > h-1$ (note that queue $\{n, d, h-1\}$ does not exist if $H^{\min}_{n\to d} > h-1$).

The back-pressure of link (m, n) is defined to be

$$P_{(m,n)}[t] = \max\left\{\max_{d\in\mathcal{D}, N-1\geq h\geq H^{\min}_{m\to d}} P^{\{m,d,h\}}_{(m,n)}[t], 0\right\}.$$

Considering the example shown in Figure 3, it can be verified that $P_{(1,2)} = 0$, $P_{(1,3)} = Q_{\{1,4,3\}} - Q_{\{3,4,2\}} = 3$, $P_{(1,4)} = Q_{\{1,4,1\}} - Q_{\{4,4,0\}} = 8$, $P_{(2,3)} = Q_{\{2,4,2\}} - Q_{\{3,4,1\}} = 5$, and $P_{(3,4)} = Q_{\{3,4,1\}} - Q_{\{4,4,0\}} = 4$.

Shortest-path-aided Back-Pressure Algorithm: Consider time slot t.

Step 0: The packets injected by flow f are deposited into queue $\{s(f), d(f), H_f\}$ maintained at node $s(f)$.

Step 1: The network first computes $\boldsymbol{\mu}^*[t]$ that solves the following optimization problem:

$$\boldsymbol{\mu}^*[t] = \arg\max_{\boldsymbol{\mu}\in\Gamma} \sum_{(m,n)\in\mathcal{L}} \mu_{(m,n)} P_{(m,n)}[t], \tag{4}$$

Step 2: Consider link (m,n). If $\mu^*_{(m,n)}[t] > 0$ and $P_{(m,n)} > 0$, node m selects a queue $\{m,d,h\}$ such that

$$Q_{\{m,d,h\}}[t] - Q_{\{n,d,h-1\}}[t] = P_{(m,n)}[t],$$

and transfers packets from queue $\{m,d,h\}$ to queue $\{n,d,h-1\}$ at rate $\mu^*_{(m,n)}[t]$.

We again consider the example in Figure 3. Assume the node exclusive interference model where adjacent links cannot be active at the same time. Furthermore, assume that link capacity is equal to one for all links. Then, given the queue-states shown in the figure, we can easily verify that $\mu^*_{(1,4)}[t] = \mu^*_{(2,3)}[t] = 1$ and $\mu^*_{(1,2)}[t] = \mu^*_{(1,3)}[t] = \mu^*_{(3,4)}[t] = 0$. Node 1 transmits one packet from queue $\{1,4,1\}$ to its destination (node 4), and node 2 transmits one packet from queue $\{2,4,2\}$ to queue $\{3,4,1\}$ at node 3.

Remark 1: Note that the optimization problem defined by equation (4) is a centralized problem. There has been a lot of recent work on distributed solutions, e.g., [18], [19], [20], [21], [22], which compute near optimal solutions with polynomial or even constant complexity. These distributed algorithms can be used in step 2 of the proposed algorithm in this paper. Distributed implementation, however, is not the focus of this paper.

Remark 2: From the definition of the back-pressure and the optimization (4), we can see that the packets in queue $\{m,d,h\}$ can be transmitted to its neighbor n only if $H^{\min}_{n\to d} \leq h-1$. Also packets of flow f are first queued at queue $\{s(f), d(f), H_f\}$. Based on the facts above, it can be easily verified that if a packet is received by its destination $d(f)$, then $0 = H^{\min}_{d(f)\to d(f)} \leq H_f - g$, where g is the number of hops the packet has been transmitted over. Thus, we can conclude that every delivered packet is delivered within the required number of hops under the shortest-path-aided back-pressure algorithm.

Theorem 1: Given traffic $\mathbf{A}[t]$ and hop constraint $\mathbf{H}$ such that $((1+\epsilon)\mathbf{A}[t], \mathbf{H}) \in \Lambda_{\mathcal{G}}$, the network is stochastically stable under the shortest-path-aided back-pressure algorithm; and packets delivered are routed over paths that satisfy corresponding hop constraints.

Proof: The second part of the theorem has been explained in Remark 2. To prove the first part, we define a Lyapunov function

$$V[t] = \sum_{\{n,d,h\}} \left(Q_{\{n,d,h\}}[t]\right)^2.$$

It can be shown that there exists $Q_{\max} > 0$ such that if $Q_{\{n,d,h\}}[t] > Q_{\max}$ for some queue $\{n,d,h\}$, then

$$\begin{aligned}\mathbf{E}[V[t+1] - V[t] | \mathbf{Q}[t]] &< -\delta + \\ &+ \sum_{(m,n)\in\mathcal{L}} \sum_{d,h} \hat{\mu}^{\{m,d,h\}}_{(m,n)} \left(Q_{\{m,d,h+1\}}[t] - Q_{\{n,d,h\}}[t]\right) \\ &- \sum_{(m,n)\in\mathcal{L}} \mu^*_{(m,n)}[t] P_{(m,n)}[t],\end{aligned}$$

where $\left\{\hat{\mu}^{\{m,d,h\}}_{(m,n)}\right\} = \hat{\boldsymbol{\mu}}$ is the rate-vector satisfying condition (i)-(iii) for given traffic $((1+\epsilon)\mathbf{A}[t], \mathbf{H})$ ($\hat{\boldsymbol{\mu}}$ exists because $((1+\epsilon)\mathbf{A}[t], \mathbf{H}) \in \Lambda_{\mathcal{G}}$), and $\{\mu^*_{(m,n)}[t]\} = \boldsymbol{\mu}^*[t]$ is the optimal solution of (4) given $\mathbf{Q}[t]$.

We also can prove that

$$\begin{aligned}&\sum_{(m,n)\in\mathcal{L}} \sum_{d,h} \hat{\mu}^{\{m,d,h\}}_{(m,n)} \left(Q_{\{m,d,h+1\}}[t] - Q_{\{n,d,h\}}[t]\right) \\ &\leq \sum_{(m,n)\in\mathcal{L}} \mu^*_{(m,n)}[t] P_{(m,n)}[t],\end{aligned}$$

which implies that $\mathbf{E}[V[t+1] - V[t] | \mathbf{Q}[t]] < -\delta$ if $Q_{\{n,d,h\}}[t] > Q_{\max}$ for some $\{n,d,h\}$. This part of the theorem follows from Foster's Criterion [28]. We skip the proof details due to space constraints. Interested readers can find the details in [29]. ■

V. Throughput-Optimal and Hop-Optimal Routing/Scheduling

In the previous section, we proposed the shortest-path-aided back-pressure algorithm that is throughput-optimal and supports per-flow hop-constraint.

In this section, we consider the scenario where no hop constraint is imposed. Recall that $N-1$ is an upper bound on the number of hops of loop-free paths. Define $\bar{\mathbf{H}}$ such that $\bar{H}[f] = N-1$ for all $f \in \mathcal{F}$. Then, we can assume that a flow is always associated with hop-constraint $\bar{\mathbf{H}}$, i.e., all loop-free paths are allowed. Note that considering only loop-free paths does not change the network throughput region. Thus we say $\mathbf{A}[t]$ is within the network throughput region if $(\mathbf{A}[t], \bar{\mathbf{H}}) \in \Lambda_{\mathcal{G}}$, which is also written as $\mathbf{A}[t] \in \Lambda_{\mathcal{G}}$.

It is well-known that the back-pressure algorithm can stabilize any $\mathbf{A}[t]$ that is in the network throughput region. However, the back-pressure algorithm exploits all feasible paths, which leads to undesirable delay performance as shown in Section II. Intuitively, we should only use short-paths when the traffic load is low, and start to exploit longer-paths as the traffic load increases. We note that the number of hops used to deliver a packet is an important parameter in two senses: *(i)* the number of hops is related to the wireless resource used to deliver the packet; *(ii)* the number of hops is also related to the end-to-end delay. Motivated by these observations, we will design an algorithm that is not only throughput-optimal, but also minimizes the average number of hops used to deliver a packet. The motivation is the hope that *such an algorithm will not only minimize the number of transmissions required to support the traffic, but also reduce the average end-to-end transmission delay.* (As we will later see from simulations, minimizing hop-count does seem to result in smaller end-to-end delays).

A. Hop Minimization

Given traffic $\mathbf{A}[t] \in \Lambda_{\mathcal{G}}$, we let $\mathcal{S}_{\mathbf{A}[t]}$ denote the set of routing/scheduling policies that stabilize the network. We further define $A_{f,h,\mathcal{P}}[\infty]$ to be the fraction of flow f that is delivered with exact h hops under policy $\mathcal{P}$, which is well defined when the network is stochastically stable. Our

objective is to find a policy $\mathcal{P}^*$ such that

$$\mathcal{P}^* = \arg \min_{\mathcal{P} \in \mathcal{S}_{\mathbf{A}[t]}} \sum_{f \in \mathcal{F}} \sum_{N-1 \geq h > 0} h A_{f,h,\mathcal{P}}[\infty]. \tag{5}$$

Note that each stabilizing policy $\mathcal{P}$ yields an obtainable rate vector $\hat{\boldsymbol{\mu}} = \left\{\hat{\mu}_{(m,n)}^{(m,d,h)}\right\}$. Thus, problem (5) is equivalent to the following optimization problem:

$$\min \sum_{f \in \mathcal{F}} \sum_{N-1 \geq h > 0} K h A_{f,h} \tag{6}$$

$$\text{s.t.: } \sum_{f \in \mathcal{F}} A_{f,h} 1_{\substack{s(f)=n \\ d(f)=d}} + \sum_{m:(m,n) \in \mathcal{L}} \hat{\mu}_{(m,n)}^{(m,d,h+1)} \leq \sum_{i:(n,i) \in \mathcal{L}} \hat{\mu}_{(n,i)}^{(n,d,h)}, \ \forall (n,d,h) \text{ such that } n \neq d; \tag{7}$$

$$\hat{\mu}_{(m,n)}^{(m,d,h)} = 0, \text{ if } h - 1 < H_{n \to d}^{\min}; \tag{8}$$

$$\left\{ \sum_{d \in \mathcal{D}, N-1 \geq h > 0} \hat{\mu}_{(m,n)}^{(m,d,h)} \right\}_{(m,n) \in \mathcal{L}} \in \mathcal{CH}(\Gamma); \tag{9}$$

$$\hat{\mu}_{(m,n)}^{(m,d,h)} \geq 0 \tag{10}$$

$$\sum_{N-1 \geq h > 0} A_{f,h} = \mathbf{E}[A_f[t]], ; \tag{11}$$

$$A_{f,h} \geq 0. \tag{12}$$

Note that K is a positive constant, and the optimal solution is the same for all $K > 0$.

To understand problem (6), we can think that we split flow f into $N-1$ flows – $(f_1, \ldots, f_{N-1})$, allocate $\frac{A_{f,h}}{\mathbf{E}[A_f[t]]}$ fraction of flow f to flow f_h, and impose hop constraint h to flow f_h. Then the average number of hops per packet delivery of flow f is

$$\sum_{N-1 \geq h > 0} h A_{f,h}.$$

Thus, problem (6) is to find a splitting that is supportable and also minimizes the number of hops used to support the traffic.

B. Dual Decomposition

To solve optimization problem (6), we define $\beta_{n,d,h}$ to be the Lagrange multiplier associated with (7). Then we can obtain a partial Lagrange dual function as follows:

$$L(\boldsymbol{\beta}) = \min_{\{A_{f,h}\}, \hat{\boldsymbol{\mu}} \in \mathcal{CH}(\Gamma)} \left(\sum_{f \in \mathcal{F}, N \geq h > 0} K h A_{f,h} + \sum_{\{n,d,h\}} \beta_{n,d,h} \left(A_{\text{in}(\{n,d,h\})} + \hat{\mu}_{\text{in}(\{n,d,h+1\})} - \hat{\mu}_{\text{out}(\{n,d,h\})} \right) \right)$$

subject to: (8) - (12),

where

$$\hat{\mu}_{\text{out}(\{n,d,h\})} = \sum_{i:(n,i) \in \mathcal{L}} \hat{\mu}_{(n,i)}^{\{n,d,h\}},$$

$$\hat{\mu}_{\text{in}(\{n,d,h+1\})} = \sum_{m:(m,n) \in \mathcal{L}} \hat{\mu}_{(m,n)}^{\{m,d,h+1\}},$$

and

$$A_{\text{in}(\{n,d,h\})} = \sum_{f \in \mathcal{F}} A_{f,h} 1_{s(f)=n, d(f)=d}.$$

According to the Slater's condition [27], the strong duality holds. Thus, there exist $(\boldsymbol{\beta}^*, \boldsymbol{\mu}^*, \mathbf{A}^*)$ such that $(\mathbf{A}^*, \boldsymbol{\mu}^*)$ is the optimal solution to problem (6), and

$$\begin{aligned} &(\mathbf{A}^*, \boldsymbol{\mu}^*) \\ &= \arg \min_{\mathbf{A}} \sum_{f \in \mathcal{F}, h > 0} \left(K h A_{f,h} + \beta^*_{s(f),d(f),h} A_{f,h} \right) \\ &- \arg \max_{\hat{\boldsymbol{\mu}} \in \mathcal{CH}(\Gamma)} \sum_{\{n,d,h\}} \beta^*_{n,d,h} \left(\hat{\mu}_{\text{out}(\{n,d,h\})} - \hat{\mu}_{\text{in}(\{n,d,h+1\})} \right). \end{aligned}$$

From the equality above, we can thus conclude that there exist $(\boldsymbol{\beta}^*, \boldsymbol{\mu}^*, \mathbf{A}^*)$ such that the following equations hold:

$$\begin{aligned} &\{A^*_{f,h}\}_{N-1 \geq h > 0} \in \\ &\arg \min_{A_{f,h}} \sum_{N-1 \geq h > 0} \left(K h A_{f,h} + \beta^*_{s(f),d(f),h} A_{f,h} \right) \end{aligned} \tag{13}$$

$$\text{subject to:} (11) - (12);$$

$$\boldsymbol{\mu}^* \in \arg \max_{\hat{\boldsymbol{\mu}} \in \mathcal{CH}(\Gamma)} \sum_{n,d,h} \beta^*_{n,d,h} \left(\hat{\mu}_{\text{in}(\{n,d,h+1\})} - \hat{\mu}_{\text{out}(\{n,d,h\})} \right) \tag{14}$$

$$\text{subject to:} (8) - (10);$$

$$\beta^*_{n,d,h} \left(\mu^*_{\text{out}(\{n,d,h\})} - A^*_{\text{in}(\{n,d,h\})} - \mu^*_{\text{in}(\{n,d,h+1\})} \right) = 0 \tag{15}$$

where equality (15) holds according to the definition of Lagrange multipliers.

C. Joint Traffic-Splitting And Shortest-Path-Aided Back-Pressure Algorithm

Now motivated by (13) and (14), we propose a joint traffic-splitting and shortest-path-aided back-pressure algorithm.

First note that

$$\sum_{n,d,h} \beta^*_{n,d,h} \left(\hat{\mu}_{\text{in}(\{n,d,h\})} - \hat{\mu}_{\text{out}(\{n,d,h\})} \right)$$

is linear in terms of $\hat{\boldsymbol{\mu}}$. Thus, we have

$$\begin{aligned} &\max_{\hat{\boldsymbol{\mu}} \in \mathcal{CH}(\Gamma)} \sum_{n,d,h} \beta^*_{n,d,h} \left(\hat{\mu}_{\text{in}(\{n,d,h\})} - \hat{\mu}_{\text{out}(\{n,d,h\})} \right) \\ &= \max_{\boldsymbol{\mu} \in \Gamma} \sum_{n,d,h} \beta^*_{n,d,h} \left(\mu_{\text{in}(\{n,d,h\})} - \mu_{\text{out}(\{n,d,h\})} \right). \end{aligned}$$

Note that the Lagrange multiplier $\beta_{(n,d,h)}$ is related to queue length $Q_{\{n,d,h\}}$, and (7)-(10) are the same as conditions (i)-(iii) defined in Section IV-A, so equality (14) motivates us to use the shortest-path-aided back-pressure defined by (4).

Furthermore, equality (13) motivates us to propose a traffic-splitting scheme such that, at time slot t, the $A_f[t]$ arrivals of flow f are deposited in queue h that minimizes

$$K h + Q_{\{s(f),d(f),h\}}[t].$$

Joint Traffic-splitting and Shortest-path-aided Back-Pressure Algorithm:

Traffic Splitting: At time t, external arrivals of flow f are deposited into queue $\{s(f), d(f), h_f^*[t]\}$, where $h_f^*[t]$ is the smallest integer of the following set:

$$\left\{\tilde{h} : \tilde{h} \in \arg \min_{N-1 \geq h > 0} \left(Kh + Q_{\{s(f),d(f),h\}}[t]\right)\right\} \quad (16)$$

Routing/Scheduling: The shortest-path-aided back-pressure algorithm without step 0.

We first show that the algorithm above is throughput-optimal.

Theorem 2: Given $\mathbf{A}[t]$ such that $(1+\epsilon)\mathbf{A}[t] \in \Lambda_{\mathcal{G}}$, the network is stochastically stable under joint traffic-splitting and shortest-path-aided back-pressure algorithm.

Proof: It can be easily verified that $\{\mathbf{Q}[t]\}_t$ is a Markov chain. We define a Lyapunov function

$$V[t] = \sum_{\{n,d,h\}} \left(Q_{\{n,d,h\}}[t]\right)^2$$

and prove that there exists $Q_{\max}$ such that if $Q_{\{n,d,h\}}[t] > Q_{\max}$ for some $\{n, d, h\}$, then

$$\mathbf{E}[V[t+1] - V[t] | \mathbf{Q}[t]] < -\delta, \quad (17)$$

which implies the positive recurrence of the Markov chain. We skip the proof details due to space constraints. Interested readers can find the details in [29]. ∎

Now given $\mathbf{A}[t]$ such that $(1+\epsilon)\mathbf{A}[t] \in \Lambda_{\mathcal{G}}$, we define

$$A_{f,h,K}[\infty] = \lim_{t\to\infty} \mathbf{E}[A_{f,h}[t]]$$

under the joint traffic control and shortest-path-aided back-pressure algorithm with parameter K. Note that $A_{f,h,K}[\infty]$ is well-defined because the network is stable according to Theorem 2.

Next we prove that the algorithm asymptotically solves the optimization problem (6) as $K \to \infty$.

Theorem 3: Given $\mathbf{A}[t]$ such that $(1+\epsilon)\mathbf{A}[t] \in \Lambda_{\mathcal{G}}$, under the joint traffic-allocation and shortest-path-aided back-pressure, we have

$$\lim_{K\to\infty} \sum_{f\in\mathcal{F}, N-1\geq h>0} hA_{f,h,K}[\infty] = \sum_{f\in\mathcal{F}, N-1\geq h>0} hA_{f,h}^*, \quad (18)$$

where and $\{A_{f,h}^*\}$ is the optimal solution to problem (6).

Proof: Based on Theorem 2, we can first show that there exists $M_3 > 0$ such that

$$\lim_{t\to\infty} \sum_{f\in\mathcal{F}} \sum_{N-1\geq h>0} h\mathbf{E}[A_{f,h,K}[t]] \leq \sum_{f\in\mathcal{F}} \sum_{N-1\geq h>0} hA_{f,h}^* + \frac{M_3}{K}. \quad (19)$$

Furthermore, it is easy to see that

$$\sum_{f\in\mathcal{F}} \sum_{N-1\geq h>0} h\mathbf{E}[A_{f,h,K}[t]] \geq \sum_{f\in\mathcal{F}} \sum_{N-1\geq h>0} hA_{f,h}^*$$

holds for any t and K. Thus the theorem holds. We skip the proof details due to space constraints. Interested readers can find the details in [29]. ∎

Remark 3: According to Theorem 3, we should choose a large K to minimize the average-number of hops per packet delivery. However, we notice that with a large K, packets are assigned to queue $\{s(f), d(f), h\}$ only when queue $\{s(f), d(f), h-1\}$ has a large backlog, which could lead to a large queueing delay (i.e., large MAC delay). Thus, there is a tradeoff choosing the value of K (to trade-off between reducing hop-count and queueing delay). In Section VI, we will study the impact of K on the packet delay performance using simulations.

VI. Simulations

In this section, we use simulations to compare the network performance under the joint traffic-splitting and shortest-path-aided back-pressure algorithm and the traditional back-pressure algorithm. We use the term *the joint algorithm* to refer to the joint traffic-splitting and shortest-path-aided back-pressure algorithm.

A. Simulation Setup

We consider a 4×4 grid network with 16 nodes and 48 links as shown in Figure 1. At the beginning of each time slot, each node generates a packet with probability λ. The destination of the generated packet is randomly, uniformly chosen from all the nodes in the network. Thus, there is a flow between each pair of nodes, i.e., there are $16 \times 15 = 240$ flows in the network. The mean arrival rate of each flow is $\lambda/15$.

We assume each link can serve one packet at each time slot. No two adjacent links can transmit at the same time, that is, we impose the half-duplex constraint.

In the following simulations, we choose different values of λ (node traffic generation rate) and K (the parameter used in the joint algorithm). For each (λ, K), we execute the simulation for 5000 iterations. We then compute the average end-to-end delay and average number of hops per packet delivery.

B. Average number of hops per packet delivery

We first computed the average number of hops over all flows. We considered the back-pressure algorithm, and the joint algorithm with $K = 0.01, 0.1, 1,$ and 10.

From Figure 4, we have the following observations:

- When λ is small, the average number of hops per packet delivery under the joint algorithm is much smaller than the one under the back-pressure algorithm. This is because the back-pressure exploits all feasible paths even when the traffic load is light.
- When λ is large (the network is critically loaded), the average number of hops under the joint algorithm is similar to the one under the back-pressure algorithm. This is because the back-pressure algorithm is optimal when the network is critically loaded.

Figure 5 is the "zoomed-in" picture of Figure 4, and shows the average number of hops under the joint algorithm with

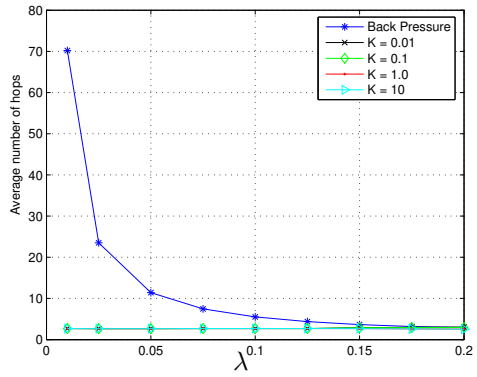

Fig. 4. Back-pressure algorithm vs the joint algorithm

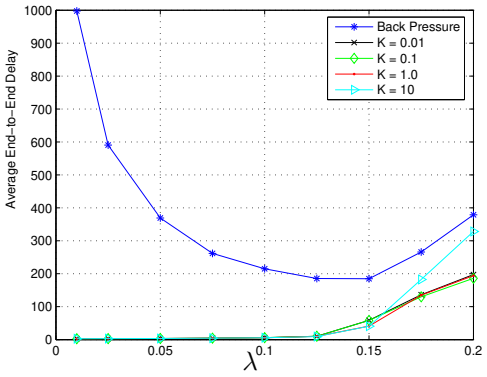

Fig. 6. Back-pressure vs our joint algorithm

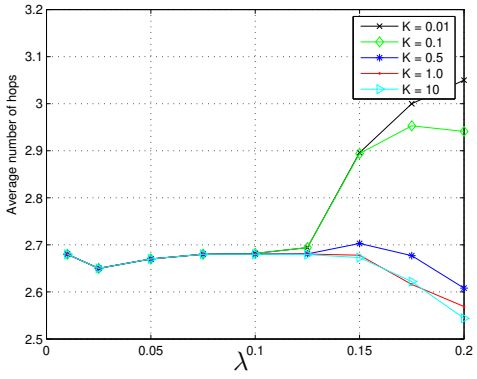

Fig. 5. Performance of the joint algorithm with different values of K

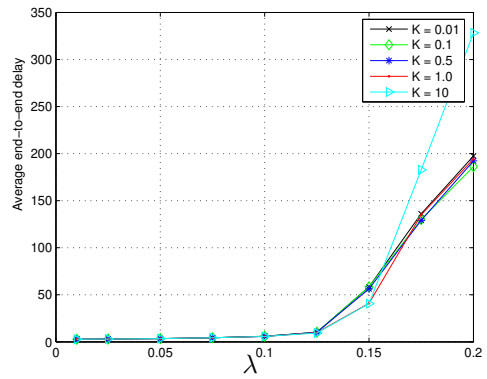

Fig. 7. Performance of the joint algorithm with different values of K

different values of K. We can see that larger K yields smaller average number hops. This is because the average path-lengths are asymptotically minimized when $K \to \infty$, which is proved in Theorem 3.

C. Average end-to-end delay

We also computed the average end-to-end delay over all flows. Similar to the average number of hops, in Figure 6, we can see that the back pressure performs very poorly with small λ. This can be attributed to the excessive looping in the route of each packet and can roughly be interpreted as a random walk on the two-dimensional network.

Furthermore, with large λ, the simulation also shows a significant improvement under the joint algorithm. This is different from the behavior of the average number of hops observed in Figure 4, where the joint algorithm and back-pressure algorithm have the similar performance when λ is large. From the simulations, we observed that the reason seems to be that the joint algorithm has a smaller variance in the path-lengths traversed by packets thus resulting in smaller queueing delays compared to the traditional back-pressure algorithm (even though the average path-lengths are similar).

Figure 7 is the "zoomed-in" picture of Figure 6, which only shows the average end-to-end delay under the joint algorithm with different values of K. We can see that larger K here does not result in smaller end-to-end delay, which is different from the behavior of the average number of hops shown in Figure 5. Note that under the joint algorithm, the source of flow f sends packets into queue $\{s(f), d(f), h+1\}$ only if $Q_{\{n,d,h+1\}} > Q_{\{n,d,h\}} + K$, so K is the barrier to prevent long paths from being used before short paths are saturated. Thus, large K implies that long paths are exploited only after the queues for short paths build up, which leads to larger queueing delay. This observation indicates that K should be properly chosen in order to minimize both the average number of hops and the average end-to-end delay. For the network studied in our simulations, we found that $K = 1$ is a good value to use.

D. End-to-end Delay Distribution

Figure 8 shows the end-to-end distribution of flow $(1, 16)$. We can see that the joint algorithm has much steeper slopes compared to the back-pressure algorithm, which again indicates that the joint algorithm has a much better delay performance compared to the back-pressure algorithm.

VII. Discussion

A. Minimum-Weight-Aided Back-Pressure

In Section IV and V, the scheduling/routing algorithms we developed use the shortest-path-information in finding the

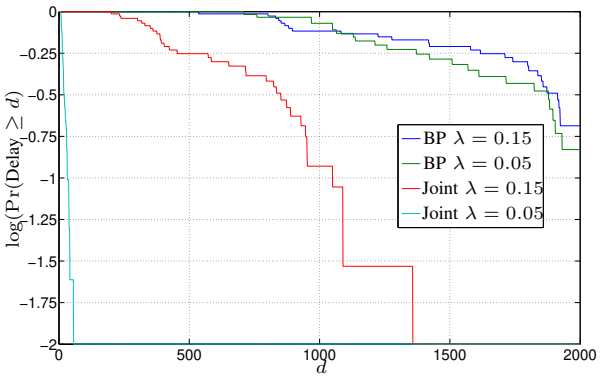

Fig. 8. End-to-end delay distribution of flow $(1, 16)$

next hop. The length of a path is defined to be the number of hops along the path. Instead of counting the number of hops, we can assign different weights to different links. The weight can be the propagation time of the link, the geographic distance between two nodes, etc. Then, letting $W^{\min}_{n\to d}$ denote the minimum aggregated weight from node n to node d, we can use this information to replace $H^{\min}_{n\to d}$ to have algorithms that support other quality of service constraints.

B. Queue Complexity

Note that given N nodes in the network, the number of hops of a loop-free path is most $N-1$. Thus at each node, we need to maintain at most $N(N-1)$ queues. A cluster-back-pressure algorithm [15] has been proposed to reduce the number of queues at each node. By using the cluster technique proposed in [15], we can reduce the queue complexity of the proposed algorithm in this paper.

VIII. Conclusion

In this paper, we have proposed new routing/scheduling algorithms that integrate the back-pressure algorithm and shortest-path routing. Using simulations, we have demonstrated a significant end-to-end delay performance improvement using the proposed algorithm.

Acknowledgment: The authors gratefully acknowledge the useful discussions with Prof. R. Srikant, University of Illinois at Urbana-Champaign. This work was partially supported by NSF Grants CNS-0347400, CNS-0519535, CNS-0721380, and CNS-0831756, the Darpa ITMANET program and the DTRA grant HDTRA1-08-1-0016.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 4, pp. 1936–1948, December 1992.

[2] ——, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inform. Theory*, vol. 39, pp. 466–478, March 1993.

[3] X. Lin and N. Shroff, "Joint rate control and scheduling in multihop wireless networks," Paradise Island, Bahamas, December 2004.

[4] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," in *Proc. IEEE Infocom.*, 2005.

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

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

[7] M. J. Neely, "Optimal backpressure routing for wireless networks with multi-receiver diversity," in *Proc. Conf. on Information Sciences and Systems (CISS)*, 2006, pp. 18–25.

[8] A. Eryilmaz and R. Srikant, "Joint congestion control, routing and mac for stability and fairness in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1514–1524, August 2006.

[9] M. Neely, "Energy optimal control for time varying wireless networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 2915–2934, July 2006.

[10] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, 2006.

[11] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*, 2006, foundations and Trends in Networking.

[12] E. Yeh and R. Berry, "Throughput optimal control of wireless networks with two-hop cooperative relaying," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, June 2007.

[13] ——, "Throughput optimal control of cooperative relay networks," *IEEE Transactions on Information Theory: Special Issue on Models, Theory, and Codes for Relaying and Cooperation in Communication Networks*, vol. 53, no. 10, pp. 3827–3833, October 2007.

[14] K. Jung and D. Shah, "Low delay scheduilng algorithms," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2007.

[15] L. Ying, R. Srikant, and D. Towsley, "Cluster-based back-pressure routing algorithm," in *Proc. IEEE Infocom.*, 2008.

[16] Web article, available at http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf.

[17] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "CDMA data QoS scheduling on the forward link with variable channel conditions," *Bell Labs Tech. Memo*, April 2000.

[18] X. Lin and S. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," in *Proc. Conf. on Decision and Control*, 2006.

[19] X. Wu and R. Srikant, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," in *Proc. IEEE Infocom.*, 2006.

[20] A. Eryilmaz, A. Ozdaglar, and E. Modiano, "Polynomial complexity algorithms for full utilization of multi-hop wireless networks," in *Proc. IEEE Infocom.*, 2007.

[21] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," in *Proc. IEEE Infocom.*, 2007.

[22] S. Sanghavi, L. Bui, and R. Srikant, "Distributed link scheduling with constant overhead," in *Proc. Ann. ACM SIGMETRICS Conf.*, San Diego, CA, June 2007.

[23] L. Lin, X. Lin, and N. Shroff, "Low-complexity and distributed energy minimization in multi-hop wireless networks," in *Proc. IEEE Infocom.*, 2007.

[24] 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.*, Phoenix, Arizona, April 2008.

[25] L. Bui, R. Srikant, and A. L. Stolyar, "Optimal resource allocation for multicast flows in multihop wireless networks," *the Philosophical Transactions of the Royal Society*, 2008.

[26] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," Ph.D. dissertation, Massachusetts Institute of Technology, November 2003.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge Unversity Press, 2004.

[28] S. Asmussen, *Applied Probability and Queues*. New York: Springer-Verlag, 2003.

[29] L. Ying, S. Shakkottai, and A. Reddy, "On Combining Shortest-Path and Back-Pressure Routing Over Multihop Wireless Networks," *Technical Report,* 2008. Available at http://www.ece.iastate.edu/∼leiying/Publications/YinShaRed08.pdf

[30] L. Bui, R. Srikant, and A. Stolyar, "Novel Architectures and Algorithms for Delay Reduction in Back-Pressure Scheduling and Routing," to appear in *Proc. IEEE Infocom Mini-Conference*, 2009.