Separation of Routing and Scheduling in Backpressure-Based Wireless Networks

Hulya Seferoglu, *Member, IEEE*, and Eytan Modiano, *Fellow, IEEE*

***Abstract*—Backpressure routing and scheduling, with its throughput-optimal operation guarantee, is a promising technique to improve throughput in wireless multihop networks. Although backpressure is conceptually viewed as layered, the decisions of routing and scheduling are made jointly, which imposes several challenges in practice. In this work, we present Diff-Max, an approach that separates routing and scheduling and has three strengths: 1) Diff-Max improves throughput significantly; 2) the separation of routing and scheduling makes practical implementation easier by minimizing cross-layer operations; i.e., routing is implemented in the network layer and scheduling is implemented in the link layer; and 3) the separation of routing and scheduling leads to modularity; i.e., routing and scheduling are independent modules in Diff-Max, and one can continue to operate even if the other does not. Our approach is grounded in a network utility maximization (NUM) formulation and its solution. Based on the structure of Diff-Max, we propose two practical schemes: Diff-subMax and wDiff-subMax. We demonstrate the benefits of our schemes through simulation in ns-2.**

***Index Terms*—Backpressure routing and scheduling, network utility maximization, wireless networks.**

I. Introduction

BACKPRESSURE routing and scheduling has emerged from the pioneering work in [1] and [2], which showed that, in wireless networks, one can stabilize queues for any feasible traffic by making routing and scheduling decisions based on queue backlog differences. Moreover, it has been shown that backpressure can be combined with flow control to provide utility-optimal operation guarantee [3].

The strengths of these techniques have recently increased the interest in practical implementation of backpressure in wireless networks, some of which are summarized in Section VI. However, the practical implementation of backpressure imposes several challenges mainly due to the joint nature of the routing and scheduling, which is the focus of this paper.

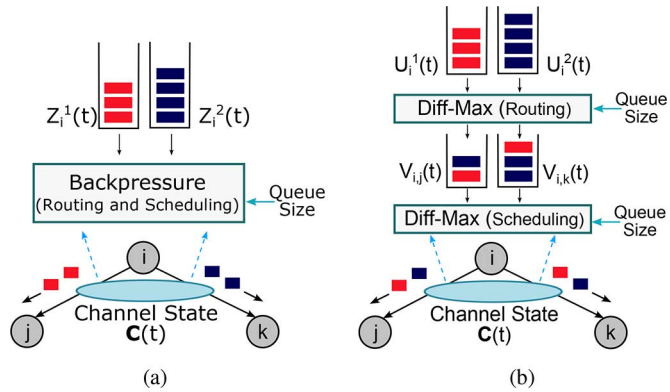


Fig. 1. Example topology consisting of three nodes i, j, k and two flows 1, 2. Note that this small topology is a zoomed part of a large multihop wireless network. The source and destination nodes of flows 1 and 2 are not shown in this example, i.e., nodes i, j, k are intermediate nodes that route and schedule flows 1 and 2. (a) Backpressure: Node i constructs per-flow queues Z_i^1 and Z_i^2 and determines the queue backlog differences at time t; $D_{i,j}^s(t) = Z_i^s(t) - Z_j^s(t)$, $D_{i,k}^s(t) = Z_i^s(t) - Z_k^s(t)$, where $s \in \{1, 2\}$. Based on the differences as well as the channel state of the network, $\boldsymbol{C}(t)$, backpressure makes joint routing and scheduling decisions. (b) Diff-Max: Node i constructs per-flow queues U_i^1 and U_i^2 in the network layer and per-link queue sizes $V_{i,j}$ and $V_{i,k}$ in the link layer, and makes routing decision based on the queue backlog differences at time t; $\tilde{D}_{i,j}^s(t) = U_i^s(t) - U_j^s(t) - V_{i,j}(t)$, $\tilde{D}_{i,k}^s(t) = U_i^s(t) - U_k^s(t) - V_{i,k}(t)$, where $s \in \{1, 2\}$. Separately, node i makes the scheduling decision based on $V_{i,j}(t)$, $V_{i,k}(t)$, and $\boldsymbol{C}(t)$.

In the backpressure framework, each node constructs per-flow queues. Based on the per-flow queue backlog differences, and by taking into account the state of the network, each node makes routing and scheduling decisions (note that scheduling algorithm is also called max-weight [4]). Although the backpressure framework is conceptually viewed as layered, the decisions of routing and scheduling are made jointly. To better illustrate this point, let us discuss the following example.

Example 1: Let us consider Fig. 1(a) for backpressure operation. At time t, node i makes routing and scheduling decisions for flows 1 and 2 based on the per-flow queue sizes $Z_i^1(t)$, $Z_i^2(t)$, as well as the queue sizes of the other nodes, i.e., j and k in this example, and using the channel state of the network $\boldsymbol{C}(t)$. In particular, backpressure determines a packet (and its flow) that should be transmitted over link $i - j$ by $s^* = \arg\max\{D_{i,j}^1(t), D_{i,j}^2(t)\}$ such that $s^* \in \{1, 2\}$. The decision mechanism is the same for link $i - k$. The scheduling algorithm also determines the link activation policy. In particular, the maximum backlog difference over each link is calculated as; $D_{i,j}^*(t) = D_{i,j}^{s^*}(t)$ and $D_{i,k}^*(t) = D_{i,k}^{s^*}(t)$. Based on $D_{i,j}^*(t), D_{i,k}^*(t)$ and $\boldsymbol{C}(t)$, the scheduling algorithm determines the link that should be activated. Note that the decisions of

Manuscript received November 06, 2013; revised August 09, 2014 and March 22, 2015; accepted April 19, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Chong. Date of publication June 12, 2015; date of current version June 14, 2016. This work was supported by the NSF under Grants CNS-0915988 and CNS-1217048, the ONR under Grant N00014-12-1-0064, and the ARO Muri under Grant No. W911NF-08-1-0238. The preliminary results of this paper were presented in part at the IEEE Conference on Computer Communications (INFOCOM), Turin, Italy, April 14–19, 2013.

H. Seferoglu was with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA. She is now with the Electrical and Computer Engineering Department at University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: hulya@uic.edu).

E. Modiano is with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: modiano@mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2436217

routing and scheduling are made jointly in backpressure, which imposes several challenges in practice. We elaborate on them next. □

Routing algorithms are traditionally designed in the network layer, while the scheduling algorithms are implemented in the link layer. However, the joint routing and scheduling nature of backpressure imposes challenges for practical implementation. To deal with these challenges, [5] implements backpressure at the link layer, and [6] proposes updates to the MAC layer. This approach is practically difficult due to device memory limitations and strict limitations imposed by device firmware and drivers not to change the link-layer functionalities. The second approach is to implement backpressure in the network layer, e.g., [7]–[9], which requires joint operation of the network and link layers so that backpressure implemented in the network layer operates gracefully with the link-layer functionalities. Thus, the network and link layers should work together synchronously, which may not be practical for many off-the-shelf devices.

Existing networks are designed in layers, in which protocols and algorithms are modular and operate independently at each layer of the protocol stack. E.g., routing algorithms at the network layer should work in a harmony with different types of scheduling algorithms in the link layer. However, the joint nature of backpressure stresses joint operation and hurts modularity, which is especially important in contemporary wireless networks, which may vary from a few node networks to ones with hundreds of nodes. It is natural to expect that different types of networks, according to their size as well as software and hardware limitations, may choose to employ backpressure partially or fully. E.g., some networks may be able to employ both routing and scheduling algorithms, while others may only employ routing. Therefore, the algorithms of backpressure, i.e., routing and scheduling should be modular.

In this paper, we are interested in a framework in which the routing and scheduling are separated. We seek to find such a scheme where the routing operates in the network layer and the scheduling is implemented in the link layer. The key ingredients of our framework, which we call Diff-Max,[1] are: 1) per-flow queues at the network layer and making routing decisions based on their differences; 2) per-link queues at the link layer and making scheduling decisions based on their size.

Example 1—Continued: Let us consider Fig. 1(b) for Diff-Max operation. 1) Routing: At time t, node i makes routing decision for flows 1 and 2 based on queue backlogs $\tilde{D}^s_{i,j}(t)$ and $\tilde{D}^s_{i,k}(t)$. This decision is made at the network layer and the routed packets are inserted into the link-layer queues. Note that in backpressure, routed packets are scheduled jointly, i.e., when a packet is routed, it should be transmitted if the corresponding links are activated. Hence, both algorithms should make the decision jointly in backpressure. However, in Diff-Max, a packet may be routed at time t, and scheduled and transmitted at a later time $t+T$ where $T>0$. 2) Scheduling: At the link layer, links are activated and packets are transmitted based on per-link queue sizes $V_{i,j}$, $V_{i,k}$, and $\boldsymbol{C}(t)$. The details of Diff-Max are provided in Section III. □

Our approach is grounded in a network utility maximization (NUM) framework [10]. The solution decomposes into several parts with an intuitive interpretation, such as routing, scheduling, and flow control. The structure of the NUM solution provides insight into the design of our scheme, Diff-Max. By separating routing and scheduling, Diff-Max makes the practical implementation easier and minimizes cross-layer operations. The following are the key contributions of this work.

- We propose a new system model and NUM framework to separate routing and flow scheduling. Our solution to the NUM problem separates routing and scheduling such that routing is implemented at the network layer and scheduling is at the link layer. Based on the structure of the NUM solution, we propose Diff-Max. We show that the deterministic version of Diff-Max optimizes utility, and we conjecture that its stochastic version satisfies stability and utility optimality.
- We extend Diff-Max to employ routing and intranode scheduling, but disable internode scheduling. We call the new framework Diff-subMax, which reduces computational complexity and overhead significantly, and provides high throughput improvements in practice. Namely, Diff-subMax only needs information from one-hop away neighbors to make its routing and scheduling decisions. Furthermore, we show that the deterministic version of Diff-subMax provides utility optimality for the networks with predetermined internode scheduling.
- We propose a window-based routing scheme, wDiff-subMax, which implements routing but disables the scheduling. wDiff-subMax is a heuristic developed based on Diff-Max and Diff-subMax, and it is designed for the scenarios in which the implementation of the scheduling in the link layer is impossible (or not desirable) e.g., due to device restrictions. wDiff-subMax makes the routing decisions on the fly and reduces overhead.
- We evaluate our schemes in a multihop setting and consider their interaction with transport, network, and link layers. We implement our schemes in a simulator, ns-2 [11], and show that they significantly improve throughput as compared to adaptive routing schemes such as Ad hoc On-Demand Distance Vector (AODV) [12] and Destination-Sequenced Distance-Vector Routing (DSDV) [13].

The structure of the rest of the paper is as follows. Section II gives an overview of the system model. Section III presents the Diff-Max formulation and design. Section IV presents the development and implementation details of Diff-Max schemes. Section V presents simulation results. Section VI presents related work. Section VII concludes the paper.

II. System Overview

In this section, we provide an overview of the system model for separation of routing and scheduling. We also provide background on the backpressure framework so that we can make a connection and comparison between our scheme and backpressure throughout the rest of the paper.

[1]Note that *Diff* means that the routing is based on the queue *diff*erences, and *Max* refers to the fact that the scheduling is based on the *max*imum of the (weighted) link-layer queues. Finally, the hyphen in Diff-Max is to mention the separation of the routing and scheduling.

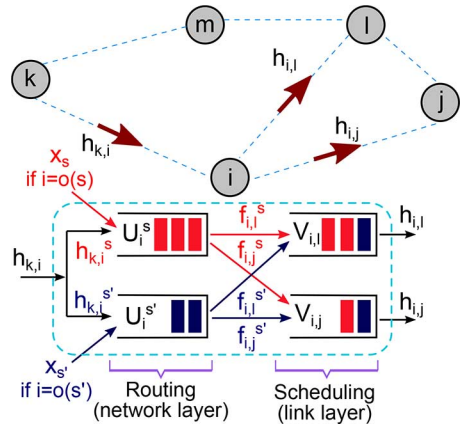


Fig. 2. Wireless mesh network. The queues at the network and link layers as well as the interaction among the queues inside node i are shown in detail. U_i^s and $U_i^{s'}$ are the network-layer queues for flows s and s', and $V_{i,j}$ and $V_{i,l}$ are the per-link queues for the links $i-j$, $i-l$. The routing algorithm operates in the network layer; the scheduling is implemented in the link layer.

A. Separation of Routing and Scheduling

We consider multihop wireless networks in which packets from a source traverse potentially multiple wireless hops before being received at their destination. In this setup, each wireless node is able to perform routing, scheduling, and flow control. In this section, we provide an overview of this setup and highlight some of its key characteristics. Fig. 2 shows the key parts of our system model in an example topology.

Setup: We consider a wireless network which consists of N nodes and L edges, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of edges. We consider in our formulation and analysis that time is slotted, and t refers to the beginning of slot t.

Sources and Flows: Let $\mathcal{S}$ be the set of unicast flows between source–destination pairs in the network. Each flow $s \in \mathcal{S}$ arrives from an application layer to a transport layer with rate $A_s(t)$, $\forall s \in \mathcal{S}$ at time-slot t. The arrivals are i.i.d. over the slots, and their expected values are $\lambda_s = E[A_s(t)]$, and $E[A_s(t)^2]$ are finite. The transport layer stores the arriving packets in reservoirs (i.e., transport-layer per-flow queues), and controls the flow. In particular, each source s is associated with rate x_s and a utility function $g_s(x_s)$, which we assume to be a strictly concave function of x_s. The transport layer determines $x_s(t)$ at time-slot t according to the utility function $g_s(x)$, and $x_s(t)$ packets are transmitted from the transport-layer reservoir to the network layer at slot t.

Queue Structures: At node $i \in \mathcal{N}$, there are network- and link-layer queues. The network-layer queues are per-flow queues; i.e., U_i^s is the queue at node $i \in \mathcal{N}$ that only stores the packets from flow $s \in \mathcal{S}$. The link-layer queues are per-link queues; i.e., at each node $i \in \mathcal{N}$, a link-layer queue $V_{i,j}$ is constructed for a neighbor node $j \in \mathcal{N}$ (Fig. 2).[2]

Flow Rates: Our model optimizes the flow rates among different nodes as well as the flow rates in a node among different layers: transport, network, and link layers.

[2]Note that in some devices, there might be only one queue (per-node queue) for data transmission instead of per-link queues in the link layer. Developing a model with per-node queues is challenging due to coupling among actions and states, so it is an open problem.

The transport layer determines $x_s(t)$ at time t and passes $x_s(t)$ packets to the network layer. These packets are inserted in the network-layer queue U_i^s (assuming that node i is the source node of flow s).

The flow rate from the network layer to the link-layer queues is $f_{i,j}^s(t)$. In particular, $f_{i,j}^s(t)$ is the flow rate of the packets, belonging to flow s, from the network-layer queue; U_i^s to the link-layer queue; $V_{i,j}$ at node i. Note that the optimization of flow rate $f_{i,j}^s(t)$ is the routing decision since it basically determines how many packets from flow s should be forwarded/routed to node j.

The link transmission rate from i to j is $h_{i,j}(t)$. Note that $h_{i,j}(t)$ bounds per-flow data rates; i.e., $h_{i,j}(t) \geq \sum_{s\in\mathcal{S}} h_{i,j}^s(t)$. E.g., $h_{k,i}(t) \geq h_{k,i}^s(t) + h_{k,i}^{s'}(t)$ in Fig. 2, where $h_{k,i}^s(t)$ is the flow rate of flow s over link $k-i$. Note that the optimization of link transmission rate $h_{i,j}(t)$ corresponds to the scheduling decisions since it determines which packets from which link-layer queues should be transmitted as well as whether a link is activated.

At every time-slot t, U_i^s changes according to the following dynamics:

$$U_i^s(t+1) \leq \max\left[U_i^s(t) - \sum_{j\in\mathcal{N}} f_{i,j}^s(t), 0\right] + \sum_{j\in\mathcal{N}} h_{j,i}^s(t) + x_s(t)1_{[i=o(s)]} \quad (1)$$

where $o(s)$ is the source node of flow s and $1_{[i=o(s)]}$ is an indicator function, which is 1 if $i = o(s)$, and 0, otherwise. Note that (1) is an inequality because the actual amount of flow rate of flow s over link $j-i$ may be lower than $h_{j,i}^s(t)$ as there may not be enough packets from flow s in the link-layer queue at node j.

At every time-slot t, $V_{i,j}$ changes according to the following queue dynamics:

$$V_{i,j}(t+1) \leq \max[V_{i,j}(t) - h_{i,j}(t), 0] + \sum_{s\in\mathcal{S}} f_{i,j}^s(t). \quad (2)$$

Note that (2) is an inequality as the number of packets in $U_i^s(t)$ may be lower than $f_{i,j}^s(t)$.

Channel Model: At slot t, $\boldsymbol{C}(t)$ is the channel state vector, where $\boldsymbol{C}(t) = \{C_1(t), \ldots, C_l(t), \ldots, C_L(t)\}$, where l represents the edges such that $l = (i,j)$, $(i,j) \in \mathcal{L}$ and $i \neq j$. We assume that $C_l(t)$ is the state of link l at time t and takes values from the set $\{\text{ON}, \text{OFF}\}$ according to a probability distribution which is i.i.d. over time-slots. If $C_l(t) = ON$, packets can be transmitted with rate R_l. Otherwise (i.e., if $C_l(t) = \text{OFF}$), packets cannot be transmitted successfully.

Let $\Gamma_{\boldsymbol{C}(t)}$ denote the set of the link transmission rates feasible at time-slot t for channel state $\boldsymbol{C}(t)$ accounting for interference among the wireless links. In particular, at every time-slot t, the link transmission vector $\boldsymbol{h}(t) = \{h_1(t), \ldots, h_l(t), \ldots h_L(t)\}$ should be constrained such that $\boldsymbol{h}(t) \in \Gamma_{\boldsymbol{C}(t)}$.

Stability Region: Let (λ_s) be the vector of arrival rates $\forall s \in \mathcal{S}$. The network stability region Λ is defined as the closure of all arrival rate vectors that can be stably transmitted in the network, considering all possible routing and scheduling policies [1]–[3]. Λ is fixed and depends only on channel statistics and interference.

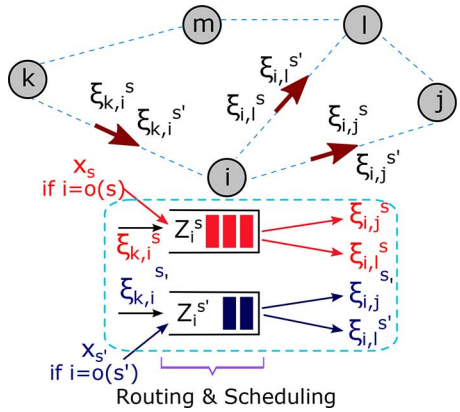


Fig. 3. Backpressure system model in a wireless mesh network. Per-flow queues inside node i are shown in detail. Z_i^s and $Z_i^{s'}$ are the per-flow queues for flows s and s', respectively. The backpressure routing and scheduling algorithm operates jointly over the per-flow queues.

B. Background on Backpressure

In this section, we provide background on backpressure so that we can make a connection and comparison between our scheme and backpressure throughout the rest of the paper. We consider a similar system model as Section II-A. Fig. 3 shows the key parts of backpressure system model in an example topology.

At node $i \in \mathcal{N}$, there are per-flow queues. The per-flow queue Z_i^s is the queue at node $i \in \mathcal{N}$ that only stores the packets from flow $s \in \mathcal{S}$. The flow rate $x_s(t)$ is determined at time t, and the corresponding number of packets are inserted in the network-layer queue Z_i^s (assuming that node i is the source node of flow s). The flow rate from node i to node j for flow s is $\xi_{i,j}^s$.

Note that the optimization of flow rate $\xi_{i,j}^s(t)$ is both the routing and scheduling decision since it basically determines how many packets from flow s should be forwarded/routed over which links. Thus, at every time-slot t, Z_i^s changes according to the following dynamics:

$$Z_i^s(t+1) \leq \max\left[Z_i^s(t) - \sum_{j\in\mathcal{N}} \xi_{i,j}^s(t), 0\right] + \sum_{j\in\mathcal{N}} \xi_{j,i}^s(t) + x_s(t)1_{[i=o(s)]}. \quad (3)$$

The backpressure scheme operates on per-flow queues Z_i^s and makes routing and scheduling decisions based on the following algorithm.

Backpressure:

- *Routing & Scheduling:* At each time-slot t, the rate $\xi_{i,j}^s(t)$ is determined by

$$\max_{\boldsymbol{\xi}} \sum_{j\in\mathcal{N}_i}\sum_{s\in\mathcal{S}} \xi_{i,j}^s(t)\left(Z_i^s(t) - Z_j^s(t)\right)$$
$$\text{s.t.} \quad \boldsymbol{\xi}(t) \in \Gamma_{\boldsymbol{C}(t)}. \quad (4)$$

Backpressure routing and scheduling algorithm in (4) stabilizes the network, and average queue backlog sizes are bounded [1], [2]. Moreover, it has been shown that backpressure can be combined with flow control to provide utility-optimal operation guarantee [3].

III. Diff-Max: Formulation and Design

A. Network Utility Maximization

In this section, we formulate and design Diff-Max. Our first step is the NUM formulation of the problem and its solution. This approach sheds light into the structure of the Diff-Max algorithms.[3]

1) Formulation: Our objective is to maximize the total utility by optimally choosing the flow rates x_s, as well as the amount of data traffic that should be routed to each neighbor node; i.e., $f_{i,j}^s$, and the link transmission rates; i.e., $h_{i,j}$

$$\max_{\boldsymbol{x},\boldsymbol{f},\boldsymbol{h}} \sum_{s\in\mathcal{S}} g_s(x_s)$$
$$\text{s.t.} \sum_{j\in\mathcal{N}} f_{i,j}^s - \sum_{j\in\mathcal{N}} h_{j,i}^s = x_s 1_{[i=o(s)]} \quad \forall i \in \mathcal{N}, s \in \mathcal{S}$$
$$\sum_{s\in\mathcal{S}} f_{i,j}^s \leq h_{i,j} \quad \forall (i,j) \in \mathcal{L}$$
$$f_{i,j}^s = h_{i,j}^s \quad \forall s \in \mathcal{S}, (i,j) \in \mathcal{L}$$
$$\boldsymbol{h} \in \tilde{\Gamma}. \quad (5)$$

The first constraint is the flow conservation constraint at the network layer: At every node i and for each flow s, the sum of the total incoming traffic, i.e., $\sum_{j\in\mathcal{N}} h_{j,i}^s$ and exogenous traffic, i.e., x_s should be equal to the total outgoing traffic from the network layer, i.e., $\sum_{j\in\mathcal{N}} f_{i,j}^s$. The second constraint is the flow conservation constraint at the link layer; the link transmission rate, i.e., $h_{i,j}$ should be larger than the incoming traffic; i.e., $\sum_{s\in\mathcal{S}} f_{i,j}^s$. Note that this constraint is an inequality because the link transmission rate can be larger than the actual data traffic. The third constraint gives the relationship between the network- and link-layer per-flow data rates, and the last constraint requires that the vector of link transmission rates, $\boldsymbol{h} = \{h_1, \ldots, h_l, \ldots h_L\}$, should be the element of the available link rates $\tilde{\Gamma}$. Note that $\tilde{\Gamma}$ is different than $\Gamma_{\boldsymbol{C}(t)}$ in the sense that $\tilde{\Gamma}$ represents long-term average rates rather than instantaneous rates.

The first two constraints are key to our work because they determine the incoming and outgoing flow relationships at the network and link layers, respectively. This approach separates routing and scheduling and assigns the routing to the network layer and scheduling to the link layer. Note that if these constraints are combined in such a way that incoming rate from a node and exogenous traffic should be smaller than the outgoing traffic for each flow, we obtain the backpressure solution [14], [15].

2) Solution: Lagrangian relaxation of the first constraint gives the following Lagrange function:

$$L(\boldsymbol{x},\boldsymbol{f},\boldsymbol{h},\boldsymbol{u},\boldsymbol{v}) = \sum_{s\in\mathcal{S}} g_s(x_s) + \sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}} u_i^s\left(\sum_{j\in\mathcal{N}} f_{i,j}^s - \sum_{j\in\mathcal{N}} h_{j,i}^s - x_s 1_{[i=o(s)]}\right) - \sum_{(i,j)\in\mathcal{L}} v_{i,j}\left(\sum_{s\in\mathcal{S}} f_{i,j}^s - h_{i,j}\right) \quad (6)$$

[3]NUM optimizes the average values of the parameters (i.e., flow rates) that are defined in Section II. By abuse of notation, we use a variable, e.g., ϕ as the average value $\phi(t)$ in our NUM formulation if both ϕ and $\phi(t)$ refers to the same parameter.

where u_i^s and $v_{i,j}$ are the Lagrange multipliers. The Lagrange function can be rewritten as

$$L(\boldsymbol{x},\boldsymbol{f},\boldsymbol{h},\boldsymbol{u},\boldsymbol{v}) = \sum_{s\in\mathcal{S}}\left(g_s(x_s) - u^s_{o(s)}x_s\right) + \sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{N}} u_i^s f_{i,j}^s - \sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{N}} u_j^s h_{i,j}^s - \sum_{(i,j)\in\mathcal{L}}\sum_{s\in\mathcal{S}} v_{i,j} f_{i,j}^s + \sum_{(i,j)\in\mathcal{L}} v_{i,j} h_{i,j} \quad (7)$$

Equation (7) can be decomposed into several intuitive subproblems such as flow control, routing, and scheduling. First, we solve the Lagrangian with respect to x_s

$$x_s = (g_s')^{-1}\left(u^s_{o(s)}\right) \quad (8)$$

where $(g_s')^{-1}$ is the inverse function of the derivative of g_s. This part of the solution can be interpreted as the flow control. Second, we solve the Lagrangian for $f_{i,j}^s$ and $h_{i,j}^s$. The following part of the solution can be interpreted as the routing:

$$\max_{\boldsymbol{f}} \sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{N}}\left(u_i^s f_{i,j}^s - u_j^s h_{i,j}^s\right) - \sum_{(i,j)\in\mathcal{L}}\sum_{s\in\mathcal{S}} v_{i,j} f_{i,j}^s$$
$$\text{s.t. } f_{i,j}^s = h_{i,j}^s \quad \forall i\in\mathcal{N}, j\in\mathcal{N}, s\in\mathcal{S} \quad (9)$$

The above problem is equivalent to

$$\max_{\boldsymbol{f}} \sum_{(i,j)\in\mathcal{L}}\sum_{s\in\mathcal{S}} f_{i,j}^s\left(u_i^s - u_j^s - v_{i,j}\right). \quad (10)$$

Third, we solve the Lagrangian for $h_{i,j}$. The following part of the solution can be interpreted as the scheduling:

$$\max_{\boldsymbol{h}} \sum_{(i,j)\in\mathcal{L}} v_{i,j} h_{i,j}$$
$$\text{s.t. } \boldsymbol{h}\in\tilde{\Gamma}. \quad (11)$$

The decomposed parts of the Lagrangian, i.e., (8), (10), and (11), and the Lagrange multipliers u_i^s and $v_{i,j}$ can be solved iteratively via a gradient descent algorithm. The convergence properties of this solution to the utility optimal operating point are provided in the Appendix. Next, we design Diff-Max based on the structure of the NUM solution.

B. Diff-Max

Now, we provide a stochastic control strategy including routing, scheduling, and flow control. The strategy, i.e., Diff-Max, which mimics the NUM solution, combines separated routing and scheduling together with the flow control.

Diff-Max:

- *Routing:* Node i determines $f_{i,j}^s(t)$ according to

$$\max_{\boldsymbol{f}} \sum_{j\in\mathcal{N}_i}\sum_{s\in\mathcal{S}} f_{i,j}^s(t)\left(\tilde{U}_i^s(t) - \tilde{U}_j^s(t) - V_{i,j}(t)\right)$$
$$\text{s.t. } \sum_{j\in\mathcal{N}_i}\sum_{s\in\mathcal{S}} f_{i,j}^s(t) \le F_i^{\max} \quad (12)$$

  where $F_i^{\max}$ is constant larger than the maximum outgoing rate from node i, $\mathcal{N}_i$ is the set of node i's neighbors, and $\tilde{U}_i^s(t)$ is the network-layer virtual queue.
  According to (12), $f_{i,j}^s(t)$ packets are removed from $\tilde{U}_i^s(t)$ and inserted to the link-layer queue $V_{i,j}(t)$. This routing algorithm mimics (10) and has the following interpretation. Packets from flow s can be transmitted to the next-hop node j as long as the network-layer queue in the next hop (node j) is small, which means that node j is able to route the packets, and the link-layer queue at the current node (node i) is small, which means that the congestion over link $i-j$ is relatively small. If the number of packets in $U_i^s(t)$ is smaller than the routing variable calculated by (12), the packets are transmitted to the link-layer queues beginning from the largest $U_i^s(t) - U_j^s(t) - V_{i,j}(t)$.
  The routing algorithm in (12) uses per-link queues as well as per-flow queues, which is the main difference of (12) as compared to the backpressure routing. The backpressure routing only uses per-flow queues and does not take into account the state of the link-layer queues, which do not exist in the standard backpressure formulation.
- *Scheduling:* At each time-slot t, the link rate $h_{i,j}(t)$ is determined by

$$\max_{\boldsymbol{h}} \sum_{(i,j)\in\mathcal{L}} V_{i,j}(t) h_{i,j}(t)$$
$$\text{s.t. } \boldsymbol{h}(t)\in\Gamma_{\boldsymbol{C}(t)} \quad \forall (i,j)\in\mathcal{L} \quad (13)$$

  where (13) mimics (11) and has the following interpretation. The link $i-j$ with the largest queue backlog $V_{i,j}$, taking into account the channel state vector, should be activated, and a packet(s) from the corresponding queue, i.e., $V_{i,j}$, should be transmitted. Note that the scheduling in (13) is known to be a difficult problem [10], [14]. Therefore, in Section IV, we propose suboptimal, low-complexity scheduling algorithms that interact well with the routing algorithm in (12).
  The scheduling algorithm in (13) differs from backpressure in the sense that it is completely independent from the routing. In particular, (13) makes the scheduling decision based on the per-link queues $V_{i,j}$ and the channel state $\boldsymbol{C}(t)$, while backpressure uses maximum queue backlog differences dictated by the routing algorithm. As it is seen, the routing and scheduling are operating jointly in backpressure, while in Diff-Max, these algorithms are separated.
- *Flow Control:* At every time-slot t, the flow/rate controller at the transport layer of node i determines the number of packets that should be passed from the transport layer to the network layer according to

$$\max_{\boldsymbol{x}} \sum_{[s\in\mathcal{S}|i=o(s)]} \left[M g_s(x_s(t)) - U_i^s(t) x_s(t)\right]$$
$$\text{s.t. } \sum_{[s\in\mathcal{S}|i=o(s)]} x_s(t) \le R_i^{\max} \quad (14)$$

  where $R_i^{\max}$ is a constant larger than the maximum outgoing rate from node i, and M is a finite constant, $M>0$. The flow control in our solution mimics (8) as well as the flow control algorithm proposed in [3].

IV. Implementation Details

We propose practical implementations of Diff-Max (Fig. 4) as well as Diff-subMax, which combines the routing algorithm

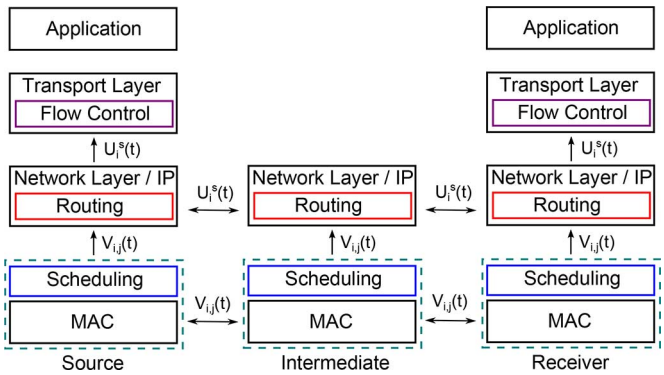


Fig. 4. Diff-Max operations at endpoints and intermediate nodes.

with a suboptimal scheduling, and wDiff-subMax, which makes routing decision based on a window-based algorithm.

A. Diff-Max

1) Flow Control: The flow control algorithm, implemented at the transport layer at the end nodes (see Fig. 4), determines the rate of each flow. We implement our flow control algorithm as an extension of UDP in the ns-2 simulator.

The flow control algorithm, at the source node i, divides time into epochs (virtual slots) such as $t_i^1, t_i^2, \ldots, t_i^k, \ldots$, where t_i^k is the beginning of the kth epoch. Let us assume that $t_i^{k+1} = t_i^k + T_i$ where T_i is the epoch duration.

At time t_i^k, the flow control algorithm determines the rate according to (14). We consider $g_s(x_s(t)) = \log(x_s(t))$ (note that any other concave utility function can be used). After $x_s(t_i^k)$ is determined, a corresponding number of packets are passed to the network layer and inserted to the network-layer queue U_i^s. Packets that are not forwarded to the network layer are stored in a reservoir at the transport layer and transmitted in later slots. At the receiver node, the transport protocol receives packets from the lower layers and passes them to the application.

2) Routing: The routing algorithm, implemented at the network layer of each node (see Fig. 4), determines routing policy, i.e., the next hop(s) to which packets are forwarded.

The first part of our routing algorithm is the neighbor discovery and queue size information exchange. Each node i transmits a message containing the size of its network-layer queues, U_i^s. These messages are in general piggybacked to data packets. The nodes in the network operates in the promiscuous mode. Therefore, each node, let us say node j, overhears a packet from node i even if node i transmits the packet to another node, let us say node k. Node j reads the queue size information from the data packet that it receives or overhears (thanks to operating on the promiscuous mode). The queue size information is recorded for future routing decisions. Note that when a node hears from another node through direct or promiscuous mode, it classifies it as its neighbor. The neighbor nodes of node i form a set $\mathcal{N}_i$. As we mentioned, queue size information is piggybacked to data packets. However, if there is no data packet for transmission, the node creates a packet to carry queue size information and broadcasts it.

Algorithm 1: The routing algorithm at node i at slot $t_i^{'k}$.

1: **for** $\forall j \in \mathcal{N}_i, \forall s \in \mathcal{S}$ **do**
2: Read the network-layer queue size information of neighbors: $U_j^s(t_i^{'k})$
3: Read the link-layer queue size information: $V_{i,j}(t_i^{'k})$
4: $f_{i,j}^s(t_i^{'k}) = 0$
5: $\{j^*, s^*\} = \arg\max_{[j\in\mathcal{N}_i, s\in\mathcal{S}]}\{U_i^s(t) - U_j^s(t) - V_{i,j}(t)\}$
6: $f_{i,j^*}^{s^*}(t_i^{'k}) = F_i^{\max}$
7: Remove $f_{i,j^*}^{s^*}(t_i^{'k})$ packets from $U_i^{s^*}$
8: Pass $f_{i,j^*}^{s^*}(t_i^{'k})$ packets to the link layer and insert them in V_{i,j^*}

The second part of our routing algorithm is the actual routing decision. Similar to the flow control algorithm, the routing algorithm divides time into epochs, such as $t_i^{'1}, t_i^{'2}, \ldots, t_i^{'k}, \ldots$, where $t_i^{'k}$ is the beginning of the kth epoch at node i. Let us assume that $t_i^{'k+1} = t_i^{'k} + T_i'$ where T_i' is the epoch duration. Note that we use $t_i^{'k}$ and T_i' instead of t_i^k and T_i because these two time epochs do not need to be the same nor synchronized.

At time $t_i^{'k}$, the routing algorithm checks $U_i^s(t_i^{'k}) - U_j^s(t_i^{'k}) - V_{i,j}(t_i^{'k})$ for each flow s. Note that $U_j^s(t_i^{'k})$ is not the instantaneous value of U_j^s at time $t_i^{'k}$, but the latest value of U_j^s heard by node i before $t_i^{'k}$. Note also that $V_{i,j}(t_i^{'k})$ is the per-link queue at node i, and this information should be passed to the network layer for routing decision. According to (12), $f_{i,j}^s(t_i^{'k})$ is determined $\forall j \in \mathcal{N}_i, \forall s \in \mathcal{S}$, and $f_{i,j}^s(t_i^{'k})$ packets are removed from U_i^s and inserted to the link-layer queue $V_{i,j}$ at node i. Note that the link layer transmits packets from $V_{i,j}$ only to node j, hence the routing decision is completed. The routing algorithm is summarized in Algorithm 1. Note that Algorithm 1 considers that there are enough packets in U_i^s for transmission. If not, the algorithm lists all the links $j \in \mathcal{N}_i$ in decreasing order, according to the weight, $U_i^s(t_i^{'k}) - U_j^s(t_i^{'k}) - V_{i,j}(t_i^{'k})$. Then, it begins to route packets beginning from the link that has the largest weight.

3) Scheduling: The scheduling algorithm in (13) assumes that time is slotted. Although there are time-slotted system implementations, and also recent work on backpressure implementation over time-slotted wireless networks [9], IEEE 802.11 MAC, an asynchronous medium access protocol without timeslots, is the most widely used MAC protocol in current wireless networks. Therefore, we implement our scheduling algorithm in (13) on top of 802.11 MAC (see Fig. 4) with the following updates.

The scheduling algorithm constructs per-link queues at the link layer. Node i knows its own link-layer queues, $V_{i,j}$, and estimates the loss probability and link rates. Let us consider that $\bar{p}_l$ and $\bar{R}_l$ are the estimated values of p_l and R_l, respectively. $\bar{p}_l$ is calculated as one minus the ratio of correctly transmitted packets over all transmitted packets in a time window over link l.[4] $\bar{R}_l$ is calculated as the average of the recent (in a

[4]Note that we do not use instantaneous channel states $C_l(t)$ in our implementation since it is not practical to get this information. Even if one can estimate $C_l(t)$ using physical-layer learning techniques, $C_l(t)$ should be estimated $\forall l \in \mathcal{L}$, which is not practical in current wireless networks.

Algorithm 2: Diff-Max scheduling algorithm at node i.

1: **if** V_l, $\bar{p}_l$, or $\bar{R}_l$ is updated such that $l \in \mathcal{L}$ **then**
2: Determine q^* such that $q^* = \arg\max_{[\forall q]}\{\sum_{l\in\mathcal{L}} V_l(1 - \bar{p}_l)\bar{R}_l\pi_q^l\}$
3: **if** $\exists(i,j)$ such that $\pi_{q^*}^{(i,j)} = 1, \forall j \in \mathcal{N}_i$ **then**
4: Reduce 802.11 MAC contention window size and access the medium
5: Transmit a packet from $V_{i,j}$ according to FIFO rule
6: **else**
7: Tell 802.11 MAC that there are no packets in the queues available for transmission

window of time) link rates over link l. $V_{i,j}$, $\bar{p}_{i,j}$, and $\bar{R}_{i,j}$ are piggybacked to the data packets and exchanged among nodes. Note that this information should be exchanged among all nodes in the network since each node is required to make its own decision based on global information. Also, each node knows the general topology and interfering links.

The scheduling algorithm that we implemented mimics (13). Each node i knows per-link queues, i.e., V_l, estimated loss probabilities, i.e., $\bar{p}_l$, and link rates, i.e., $\bar{R}_l$, for $l \in \mathcal{L}$ as well all maximal independent sets, which consist of links that are not interfering. Let us assume that there are Q maximal independent sets. For the qth maximal independent set such that $q = 1, \ldots, Q$, the policy vector is; $\boldsymbol{\pi}_q = \{\pi_q^1, \ldots, \pi_q^l, \ldots, \pi_q^L\}$, where $\pi_q^l = 1$ if link l is in the qth maximal set, and $\pi_q^l = 0$, otherwise. Our scheduling algorithm selects q^*th maximal independent set such that $q^* = \arg\max_{[\forall q]}\{\sum_{l\in\mathcal{L}} V_l(1-\bar{p}_l)\bar{R}_l\pi_q^l\}$. Node i calculates q^* whenever one of the parameters, V_l, $\bar{p}_l$, $\bar{R}_l$ changes. If, according to q^*, node i decides that it should activate one of its links, then it reduces the contention window size of 802.11 MAC so that node i can access the medium quickly and transmit a packet. If node i should not transmit, then the scheduling algorithm tells 802.11 MAC that there are no packets in the queues available for transmission. Note that in order to complement Diff-Max scheduling, the 802.11 protocol has to be slightly modified. The scheduling algorithm is summarized in Algorithm 2.

Note that Algorithm 2 is a hard problem because it is reduced to maximum independent set problem, [10], [14]. Furthermore, it introduces significant amount of overhead; each node needs to know every other node's queue sizes and link loss rates. Due to the complexity of the problem and overhead, we implement this algorithm for small topologies over ns-2 for the purpose of comparing its performance to suboptimal scheduling algorithms, which we describe next.

B. Diff-Submax

Diff-subMax is a low-complexity and low-overhead counterpart of Diff-Max. The flow control and the routing parts of Diff-subMax is exactly the same as in Diff-Max. The only different part is the scheduling algorithm, which uses 802.11 MAC protocol without any changes. When a transmission opportunity arises according to the underlying 802.11 MAC at time t, then the scheduling algorithm of node i calculates weights for all outgoing links to its neighbors. Let us consider link $i-j$ at time t. The weight is $\omega_{i,j}(t) = V_{i,j}(t)(1-\bar{p}_{i,j})\bar{R}_{i,j}$. Based on the weights, the link is chosen as $l^* = \arg\max_{[j\in\mathcal{N}_i]}\omega_{i,j}(t)$. This decision means that a packet from the link-layer queue V_{l^*} is chosen according to FIFO rule and transmitted. Note that this scheduling algorithm only performs intranode scheduling, i.e., it determines from which link-layer queue packets should be transmitted, but it does not determine which node should transmit, which is handled by 802.11 MAC.

Diff-subMax has several nice features. We show in the Appendix that the deterministic version of Diff-subMax provides utility optimality for the networks with predetermined internode scheduling such as CSMA/CA. Furthermore, Diff-subMax reduces the complexity of the algorithm and overhead significantly. In particular, each node i calculates and compares weights $\omega_{i,j}(t)$ for each neighbor node. Therefore, the complexity is linear with the number of (neighbor) nodes. The overhead is also significantly reduced; each node needs to know the queue size only of its one-hop away neighbors.

C. wDiff-subMax

wDiff-subMax is a heuristic designed as an extension of Diff-subMax for the scenarios that link-layer operations and data exchange (between the network and link layers) are not possible due to Wi-Fi firmware or driver restrictions or may not be desired. Therefore, wDiff-subMax does not employ any scheduling mechanism, but only the routing and flow control. The flow control algorithm is the same as in Diff-Max. Yet, the routing algorithm is updated as explained next.

Per-flow queues as well as per-link queues are required in (12) to make the routing decision. If per-link queues are not available at the network layer, the routing decision may not be efficient as there may be (uncontrolled) congestion in the link-layer queues. In order to make the routing decisions efficiently, we propose a heuristic called wDiff-subMax. The main idea behind wDiff-subMax is to react to link-layer congestion, while still implementing (12). To achieve this, wDiff-subMax employs acknowledgement (ACK) mechanism and uses an additive increase/multiplicative decrease (AIMD) algorithm.

wDiff-subMax labels each packet with a timestamp at the network layer. When a packet is received by the next hop, an ACK packet, echoing the timestamp of the packet, is transmitted back to the previous hop. The network layer of the previous hop receives ACKs and determines round-trip time (RTT) for each packet. $\mathrm{RTT}_{i,j}^s(t_i^{'k})$ is the average round-trip time of the ACKs received in the last slot (i.e., at slot $t_i^{'k-1}$), and $\mathrm{RTT}_{i,j}^s$ is the average round-trip time of the packets.

wDiff-subMax keeps a windows size $W_{i,j}^s(t_i^{'k})$ for link $i-j$ and flow s at slot $t_i^{'k}$. At each slot $t_i^{'k}$, the routing parameter $f_{i,j}^s(t_i^{'k})$ is set to $W_{i,j}^s(t_i^{'k})$ and $f_{i,j}^s(t_i^{'k})$ packets are passed to the link layer. wDiff-subMax determines the window size according to AIMD as explained next.

If $U_i^s(t_i^{'k}) - U_j^s(t_i^{'k}) > 0$ and $\mathrm{RTT}_{i,j}^s(t_i^{'k}) < RTT_{i,j}^s$, then $W_{i,j}^s(t_i^{'k})$ is increased by 1. Note that, in this case, per-slot RTT is smaller than the average RTT, which means that congestion level in the link layer is low, and there is a positive queue backlog difference between the two nodes. Thus, more

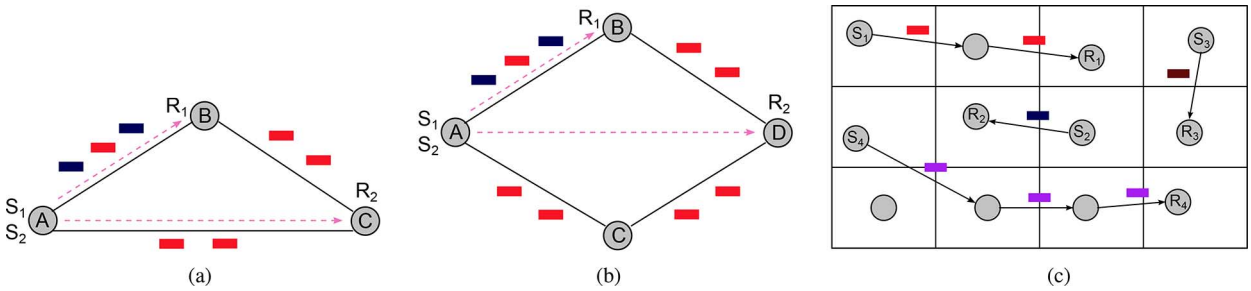


Fig. 5. Topologies used in simulations. (a) Triangle topology. There are two flows between sources S_1, S_2 and receivers R_1, R_2, i.e., from node A to B (S_1-R_1) and from node A to C (S_2-R_2). (b) Diamond topology. There are two flows between sources S_1, S_2 and receivers R_1, R_2, i.e., from node A to B (S_1-R_1) and from node A to D (S_2-R_2). (c) Grid topology. Twelve nodes are randomly placed over a 4×3 grid. An example node distribution and possible flows are illustrated in the figure.

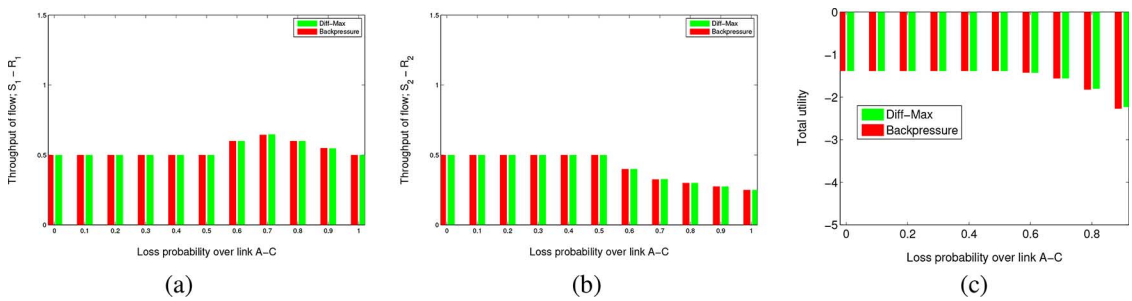


Fig. 6. Numerical results for the triangle topology shown in Fig. 5(a). The loss is over link $A - C$. (a) Throughput of $S_1 - R_1$ flow. (b) Throughput of $S_2 - R_2$ flow. (c) Total utility.

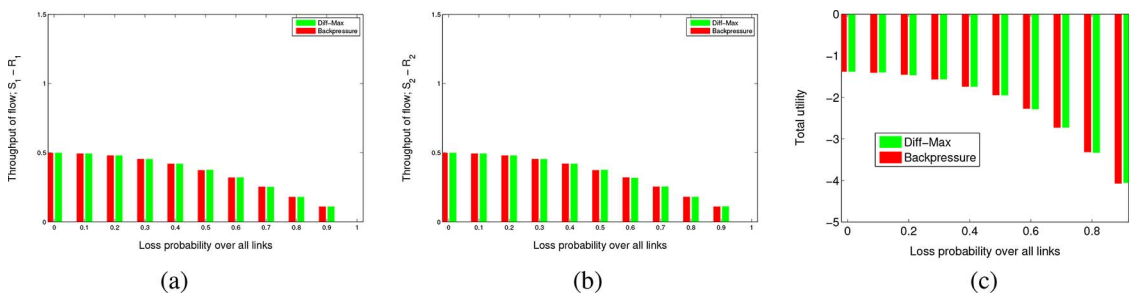


Fig. 7. Numerical results for the triangle topology shown in Fig. 5(a). The loss is over all links. (a) Throughput of $S_1 - R_1$ flow. (b) Throughput of $S_2 - R_2$ flow. (c) Total utility.

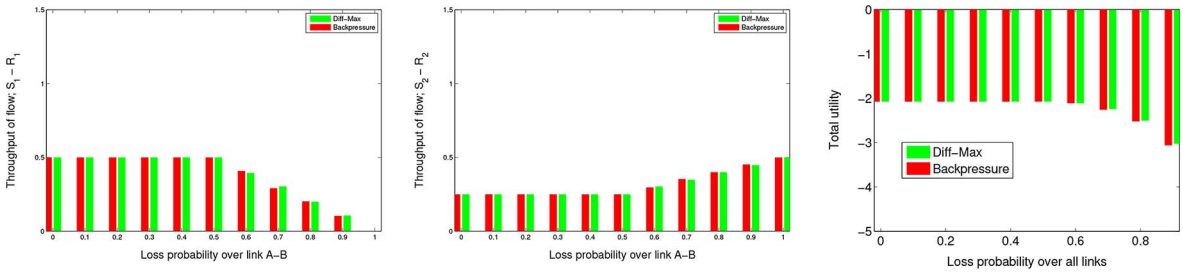
packets can be transmitted over this link, so $W^s_{i,j}(t_i^{'k})$ is increased. On the other hand, if $U^s_i(t_i^{'k}) - U^s_j(t_i^{'k}) > 0$ and $\mathrm{RTT}^s_{i,j}(t_i^{'k}) > \mathrm{RTT}^s_{i,j}$, then $W^s_{i,j}(t_i^{'k})$ is decreased by 1 since the link-layer congestion level is high, and less packets should be transmitted over this link. If none of the packets in the last slot is ACKed, this means that congestion is very high, and packets are dropped. In this case, $W^s_{i,j}(t_i^{'k})$ is halved so that the number of packets over link $i-j$ could be reduced sharply. After $W^s_{i,j}(t_i^{'k})$ is determined, $f^s_{i,j}(t_i^{'k})$ is set to $W^s_{i,j}(t_i^{'k})$ and $f^s_{i,j}(t_i^{'k})$ packets are passed to the link layer. Note that wDiff-subMax, similar to Diff-subMax, reduces computational complexity and overhead significantly as compared to Diff-Max.

V. Performance Evaluation

A. Numerical Simulations

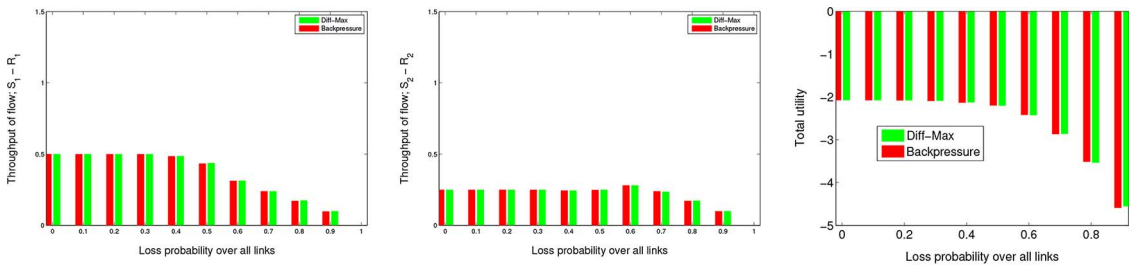
We simulate Diff-Max as well as the standard backpressure in an idealized time-slotted system. In particular, we consider triangle and diamond topologies shown in Fig. 5(a) and (b). In both topologies, there are two flows, $S_1 - R_1$ and $S_2 - R_2$, and all nodes are capable of forwarding packets to their neighbors. The simulation duration is 10 000 slots, and each simulation is repeated for 10 seeds. Each slot is in the ON or OFF state according to an i.i.d. random process with a given loss probability. The utility function in these simulations is log utility, i.e., $g_s(x_s(t)) = \log(x_s(t))$.

Fig. 6 shows the throughput and total utility (aggregated over per-flow utilities) versus the loss probability for the triangle topology when the link $A - C$ is lossy. As can be seen, the throughput and the total utility of Diff-Max is equal to that of backpressure. Similar results are observed in Fig. 6 for the same setup when all links are lossy. Note that the total utilities in Figs. 6 and 7 are negative as we employ log utility in these simulations. As can be seen, the throughput and utility of Diff-Max is equal to that of backpressure. The same results are shown for the diamond topology in Figs. 8 and 9. These results show that Diff-Max achieves the same throughput and utility as backpressure.



(a) (b) (c)

Fig. 8. Numerical results for the diamond topology shown in Fig. 5(b). The loss is over link $A-B$. (a) Throughput of S_1-R_1 flow. (b) Throughput of S_2-R_2 flow. (c) Total utility.



(a) (b) (c)

Fig. 9. Numerical results for the diamond topology shown in Fig. 5(b). The loss is over all links. (a) Throughput of S_1-R_1 flow. (b) Throughput of S_2-R_2 flow. (c) Total utility.

B. NS-2 Simulations

In this section, we simulate our schemes Diff-Max, Diff-subMax, and wDiff-subMax using the ns-2 simulator [11]. The simulation results show that our schemes significantly improve throughput, utility, and delay performance as compared to AODV [12] and DSDV [13] routing schemes. Next, we present the simulator setup and results in detail.

1) Setup: We considered three topologies: the diamond topology shown in Fig. 5(b), a grid topology shown in Fig. 5(c), and a random topology. In the diamond topology, the nodes are placed over a 500×500-m^2 terrain. Two flows are transmitted from node A to nodes B and D. In the grid topology, 4×3 cells are placed over an 800×600-m^2 terrain; 12 nodes are randomly placed in the cells. In the grid topology, each node can communicate with other nodes in its cells or with the ones in neighboring cells. Four flows are generated randomly. In the random topology, 20 nodes are randomly generated and located over an 800×800-m^2 terrain according to a uniform distribution. Ten flows are generated and transmitted between randomly selected nodes.

We consider CBR flows, which start at random times within the first 5 s and remain on until the end of the simulation, which is 100 s. The CBR flows generate packets with interarrival times 0.01 ms. IEEE 802.11 is used in the MAC layer (with updates for Diff-Max implementation as explained in Section IV). We simulated a Rayleigh fading channel with average channel loss rates 0%, 20%, 30%, 40%, 50%.[5] We have repeated each 100-s simulation for 10 seeds.

[5]We consider the loss rates in the range up to 50% because previous studies of IEEE 802.11b-based wireless mesh networks [16], [17], have reported packet loss rates as high as 50%.

The channel capacity is 1 Mb/s, the buffer size at each node is set to 1000 packets, and packet sizes are set to 1000 B. We compare our schemes Diff-Max, Diff-subMax, and wDiff-subMax to AODV and DSDV in terms of transport-level throughput, total utility (added over all flows), and average delay (averaged over all packets and flows). We employ log utility in our simulations, i.e., $g_s(x_s(t))=\log(x_s(t))$. On the other hand, packet delay is measured at the transport layer. Let $r_{s,k}$ be the time that the kth packet of flow s is received by the transport layer at the receiver side, and $t_{s,k}$ be the time that the same packet is seen by the transport layer at the transmitter side. Then, the packet delay is $r_{s,k}-t_{s,k}$.

The Diff-Max parameters are set as follows. For the flow control algorithm: $T_i=80$ ms, $R_i^{\max}=20$ packets, $M=200$. For the routing algorithm: $T_i^{'}=10$ ms, $F_i^{\max}=4$ packets.

2) Results: Fig. 10(a) shows the simulation results for the diamond topology, where only link $A-B$ is lossy. Diff-Max performs better than the other schemes for the range of loss rates since Diff-Max activates the links based on the per-link queue backlogs, loss rates, and link rates. On the other hand, Diff-subMax, wDiff-subMax, AODV, and DSDV use classical 802.11 MAC. When the loss rate over link $A-B$ increases, the total throughput of all the schemes decreases as expected. As can be seen, the decrease in our schemes Diff-Max, Diff-subMax, and wDiff-subMax is linear, while the decrease of AODV is quite sharp. The reason is that when AODV experiences loss over a path, it deletes the path and recalculates new routes. Therefore, AODV does not transmit over lossy links for some time period and tries to find new routes, which reduces throughput. On the other hand, DSDV performs better than AODV at low loss rates thanks to keeping track of multiple routes and exploiting a new route when one becomes lossy. Yet, it is worse than AODV at higher loss rates, as it requires more

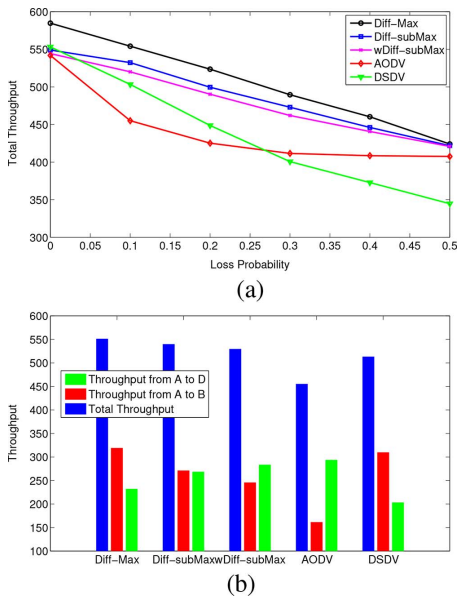


Fig. 10. Diamond topology. (a) Total throughput (in kb/s) versus average loss rate in the diamond topology. (b) Per-flow (as well as total) throughput of different policies when the average loss rate is set to 10%.

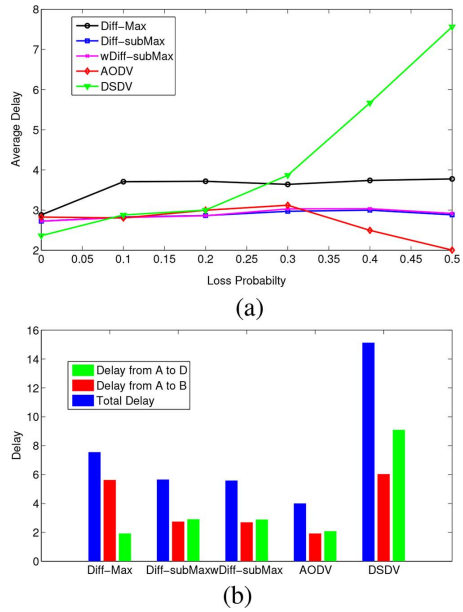


Fig. 11. Diamond topology. (a) Average delay (in seconds) versus average loss rate in the diamond topology. (b) Per-flow (as well as total) delay of different policies when the average loss rate is set to 50%.

packet exchanges among nodes at high loss rates, which consumes higher bandwidth and reduces throughput. Diff-subMax and wDiff-subMax improve throughput significantly as compared to both AODV and DSDV thanks to exploring routes to improve throughput. The improvement of our schemes is up to 22% and 21% over AODV and DSDV, respectively. Also, Diff-subMax and wDiff-subMax have similar throughput performance, which emphasizes the benefit of the routing part and the effective link-layer queue estimation mechanism of wDiff-subMax.

Fig. 10(a) also shows that when loss rate is 50%, the throughput improvement of all schemes (except DSDV) are similar, because at 50% loss rate, link $A-B$ becomes very inefficient, and all of the schemes transmit packets mostly from flow A to D over path $A-C-D$ and have similar performance at high loss rates. DSDV is worse because it requires more packet exchanges to keep the routing table, which wastes resources.

Fig. 10(b) elaborates more on the above discussion. It shows the throughput of two flows A to B and A to D as well as their total value when the loss rate is 10% over link $A-B$. As can be seen, the rate of flow $A-B$ is very low in AODV as compared to our schemes because AODV considers link $A-B$ to be broken at some periods during the simulation, while our schemes continue to transmit over this link. Although DSDV outperforms AODV, our schemes are still better than it in terms of throughput thanks to exploring routes to improve throughput.

Fig. 11(a) shows the delay versus loss probability for the diamond topology, where only the link $A-B$ is lossy. As can be seen, Diff-Max introduces higher delay as compared to Diff-subMax and wDiff-subMax because Diff-Max can delay packet transmission from some queues depending on their occupancy. In other words, Diff-Max transmits packets from the nodes with larger queue size, which may delay some packets significantly. On the other hand, Diff-subMax and wDiff-subMax transmit packets from the link-layer queues based on 802.11 MAC scheduling, which reduces delay. On the other hand, the delay performance of Diff-subMax and wDiff-subMax is comparable to and better than AODV and DSDV for all loss rates, which shows that our algorithms are quite efficient in terms of delay. The delay performance of DSDV is high and increases with loss rate as DSDV should update its routing table periodically and needed, which increases packet delay [13]. Fig. 11(b) shows per-flow and total delay of each algorithm when the loss rate over link $A-B$ is 50%. As can be seen, the delay of each flow is very large in DSDV while the delay performances of other algorithms are comparable.

Fig. 12 presents the results for the grid topology. In this scenario, one third of the links, which are chosen randomly, are lossy with 10% loss rate. Fig. 12(a) shows the total throughput of our schemes as well as AODV and DSDV. Although the throughput performances of our schemes are better than AODV, the total throughput of DSDV is slightly better than our schemes. The reason is that DSDV treats some flows (with longer paths) unfairly, and they do not get much (or even any) opportunity to transmit. Since the flows with shorter paths can transmit most of the time, the total throughput of DSDV becomes better. On the other hand, Fig. 12(b) shows that the total utilities of Diff-subMax and wDiff-subMax are better than DSDV. It is expected as our schemes are designed to maximize the total utility in (14). In other words, even though the total throughput of DSDV may be higher at some scenarios, the total utility, which we maximize, of Diff-subMax and wDiff-subMax is higher.

Fig. 13 presents the simulation results for the same grid topology setup. Fig. 13(a) shows the total utility versus average loss rate for our schemes as well as AODV and DSDV. Our schemes Diff-subMax and wDiff-subMax significantly improve the total utility as compared to both AODV and DSDV. Fig. 13(b) considers the same setup and presents average delay

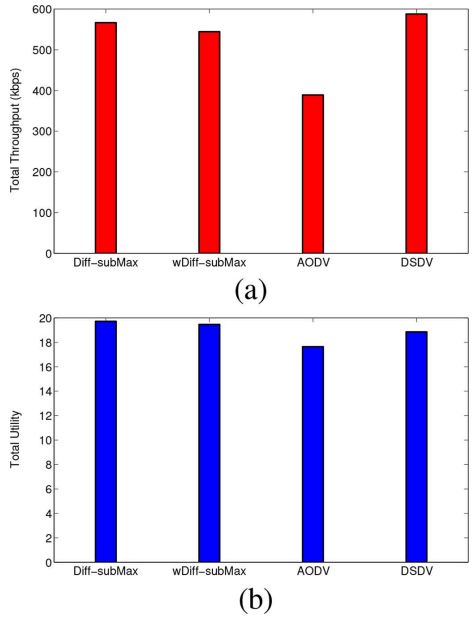


Fig. 12. Grid topology. (a) Total throughput. (b) Total utility of Diff-subMax, wDiff-subMax, AODV, and DSDV. There is no loss over the links.

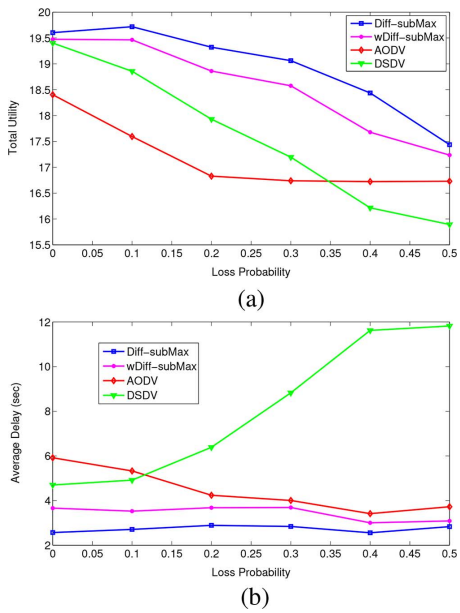


Fig. 13. Grid topology. (a) Total utility versus average loss rate. (b) Average delay versus average loss rate.

versus average loss rate. As can be seen, Diff-subMax and wDiff-subMax significantly improve delay as compared to AODV and DSDV.

Fig. 14 presents the simulation results for the random topology. In this scenario, one third of the links, which are chosen randomly, are lossy with a rate between 0% to 50%. Fig. 14(a) shows the total utility versus average loss rate results. As can be seen, our schemes significantly improve the total utility as compared to AODV and DSDV. The improvement in this scenario is higher as compared to the grid topology since there are more routing opportunities that can be exploited in this random topology scenario.

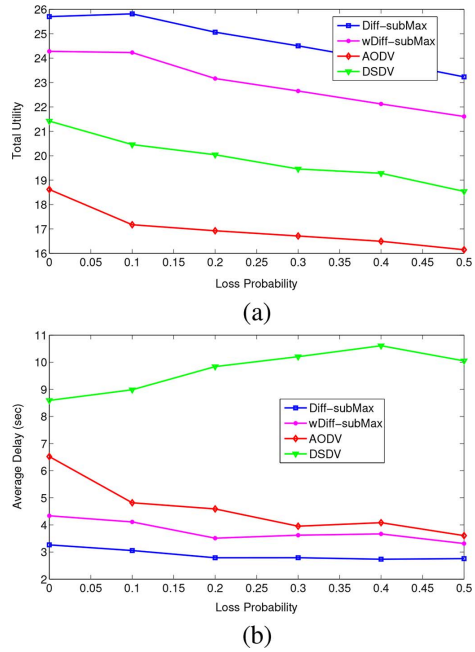


Fig. 14. Random topology. (a) Total utility versus average loss rate. (b) Average delay versus average loss rate.

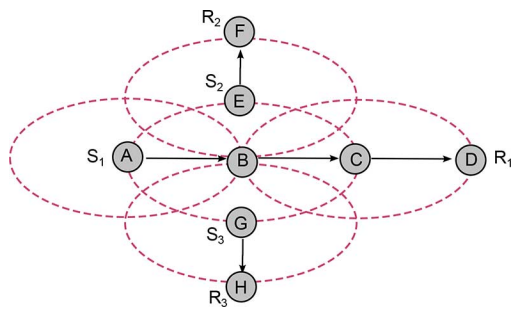


Fig. 15. Topology that we consider for the flow in the middle problem. There are three flows: Flow 1 from S_1 to R_1, Flow 2 from S_2 to R_2, and Flow 3 from S_3 to R_3. Each dashed ellipse shows a transmission and interference range of the node that is located in the center of the ellipse. In this scenario, Flow 1 suffers from the flow in the middle problem.

Fig. 14(b) shows the average delay versus average loss rate results for the random topology. Diff-subMax and wDiff-subMax improve the delay performance as compared to AODV and DSDV. The improvement as compared to DSDV is especially significant. These results show that Diff-subMax and wDiff-subMax improve both utility and delay as compared to AODV and DSDV.

C. Flow in the Middle Problem

In this section, we demonstrate the benefit of our modular algorithm design to address a specific problem called the "flow in the middle problem" [5].

Let us consider the topology shown in Fig. 15, where there are three flows: Flow 1 from S_1 to R_1, Flow 2 from S_2 to R_2, and Flow 3 from S_3 to R_3. In this scenario, Flow 1 suffers from the flow in the middle problem when 802.11 MAC is employed. In particular, node B is subject to more interference as compared to node E and G as it shares the medium with four other nodes. On the other hand, nodes E and G share the wireless medium with only two nodes. Since, 802.11 MAC tends to give equal

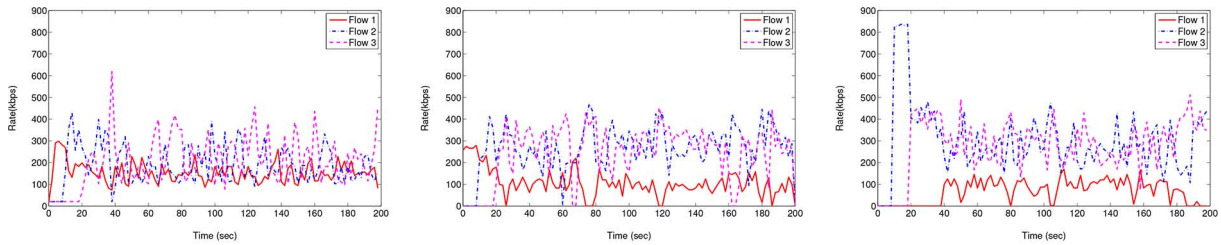


Fig. 16. Per-flow throughput versus time for the topology shown in Fig. 15. The channel capacity is 1 Mb/s, the buffer size at each node is set to 1000 packets, packet sizes are set to 1000 B, and link loss probabilities are 0. (a) Diff-subMax with updated 802.11 MAC. (b) AODV. (c) DSDV.

opportunity to all nodes in the wireless medium, node B will transmit less, which will reduce the rate of Flow 1. Thus, the rates of Flow 2 and Flow 3 will be higher, while Flow 1 has lower rate. Note that the flow in the middle problem arises from the fair share nature of 802.11 MAC. Since our Diff-subMax and wDiff-subMax algorithms also employ 802.11 MAC, the flow in the middle problem may also be seen in our algorithms.

However, thanks to our modular design, Diff-subMax could be easily updated to address the flow in the middle problem, which is not possible with AODV and DSDV. In particular, if the contention window size of 802.11 MAC is arranged depending on the weight of the chosen link (similar to [5] and [6]) in Diff-subMax, i.e., if the contention window size is inversely proportional to ω_{l^*}, where l^* is the chosen link, then the middle in the flow problem can be alleviated.

The ns-2 simulation results in Fig. 16(a) show that Diff-subMax with updated 802.11 MAC addresses the flow in the middle problem for the topology shown in Fig. 15. On the other hand, Flow 1 still suffers in AODV and DSDV as shown in Fig. 16(b) and (c). This shows the effectiveness of our modular design to address problems arising from different layers.

VI. Related Work

Backpressure and Follow-Up Work: This paper builds on backpressure, a routing and scheduling framework for communication networks [1], [2], which has generated a lot of interest in the research community [4], especially for wireless ad hoc networks [18]–[23]. Furthermore, it has been shown that backpressure can be combined with flow control to provide utility-optimal operation guarantee [3], [22]. This paper follows the main idea of backpressure and revisits it considering the practical challenges that are imposed by current networks.

Backpressure Implementation: The strengths of backpressure have recently increased the interest in the practical implementation of backpressure over wireless networks. Multipath TCP scheme is implemented over wireless mesh networks in [7] for routing and scheduling packets using a backpressure based heuristic. At the link layer, [5], [6], [24], and [25] propose, analyze, and evaluate link-layer backpressure-based implementations with queue prioritization and congestion window size adjustment. Backpressure is implemented over sensor networks [8] and wireless multihop networks [9], which are also the closest implementations to ours. The main differences in our work are that: 1) we consider separation of routing and scheduling to make practical implementation easier; 2) we design and analyze a new scheme, Diff-Max; 3) we simulate and implement Diff-Max over ns-2.

Backpressure and Queues: According to backpressure, each node constructs per-flow queues. There is some work in the literature to stretch this necessity. For example, [26] and [27] propose using real per-link and virtual per-flow queues. Such a method reduces the number of queues required in each node and reduces the delay, but it still makes routing and scheduling decisions jointly and does not separate routing from scheduling. Therefore, this approach requires strong synchronization between the network and link layers, which is difficult to implement in practice as explained in Section I.

VII. Conclusion

In this paper, we proposed Diff-Max, a framework that separates routing and scheduling in backpressure-based wireless networks. The separation of routing and scheduling makes practical implementation easier by minimizing cross-layer operations, and it leads to modularity. Our design is grounded in the NUM formulation of the problem and its solution. Based on the structure of Diff-Max, two practical schemes, Diff-subMax and wDiff-subMax, are developed. Simulations in ns-2 demonstrate significant improvement in terms of throughput, utility, and packet delay as compared to AODV and DSDV.

Appendix
Analysis of Deterministic Solutions

A. Diff-Max

In this section, we analyze the deterministic solution of Diff-Max. We first, explain the evolution of Lagrange multipliers, and then discuss the convergence of the solution to the optimal point.

Diff-Max—Lagrange Multipliers: The Lagrange multipliers u_i^s and $v_{i,j}$ are calculated using gradient descent

$$u_i^s(t+1) = \left\{ u_i^s(t) - \alpha_t \left[\sum_{j\in\mathcal{N}} f_{i,j}^s(t) - \sum_{j\in\mathcal{N}} h_{j,i}^s(t) - x_s(t) 1_{i=o(s)} \right] \right\}^+$$

$$v_{i,j}(t+1) = \left\{ v_{i,j}(t) + \beta_t \left[\sum_{s\in\mathcal{S}} f_{i,j}^s(t) - h_{i,j}(t) \right] \right\}^+ \quad (15)$$

where t is the iteration number, α_t and β_t are the step-sizes of the gradient descent algorithm, and the $^+$ operator makes the Lagrange multipliers positive.

Diff-Max—Convergence to the Optimal Point: The convergence of the distributed solutions of Diff-Max, i.e., (8), (10), (11), and (15), follows directly from the convergence of convex optimization problems through gradient descent [15], [28]. In particular, if $\lim_{t\to\infty}\alpha_t = 0, \sum_{t=0}^{\infty}\alpha_t = \infty$ and $\lim_{t\to\infty}\beta_t = 0, \sum_{t=0}^{\infty}\beta_t = \infty$, then the solution converges, i.e., $\lim_{t\to\infty}\|x(t) - x^*\| = 0$, where x^* is the utility optimal operating point of the convex optimization problem in (5).

B. Diff-Submax

In this section, we analyze Diff-subMax and its convergence properties.

Diff-subMax—Problem Formulation: Since Diff-subMax uses 802.11's CSMA/CA mechanism for internode scheduling, it is a solution to the following problem:

$$\begin{aligned}
&\max_{\boldsymbol{x},\boldsymbol{f},\boldsymbol{h}} \sum_{s\in\mathcal{S}} g_s(x_s)\\
&\text{s.t.} \sum_{j\in\mathcal{N}} f_{i,j}^s - \sum_{j\in\mathcal{N}} h_{j,i}^s = x_s 1_{[i=o(s)]} \quad \forall i\in\mathcal{N}, s\in\mathcal{S}\\
&\quad \sum_{s\in\mathcal{S}} f_{i,j}^s \le h_{i,j} \quad \forall (i,j)\in\mathcal{L}\\
&\quad f_{i,j}^s = h_{i,j}^s \quad \forall s\in\mathcal{S}, (i,j)\in\mathcal{L}\\
&\quad h_{i,j} \le (1-\bar{p}_{i,j})\bar{R}_{i,j}\tau_{i,j} \quad \forall i\in\mathcal{N}, j\in\mathcal{N}\\
&\quad \sum_{j\in\mathcal{N}} \tau_{i,j} \le \gamma_i \quad \forall i\in\mathcal{N}
\end{aligned} \tag{16}$$

where $\bar{p}_{i,j}$ and $\bar{R}_{i,j}$ are the loss probability and link rate over link $i-j$, $\tau_{i,j}$ is the percentage of time that link $i-j$ is used for transmission, and γ_i is the percentage of time node i is active for transmission. The percentage of time that a node is active, i.e., γ_i is determined by CSMA/CA, and it is constant in our problem. Note that the only difference of (16) as compared to (5) is the last two constraints. In particular, since CSMA/CA is employed for internode scheduling, it gives opportunity to each node for transmission. The percentage of these transmission opportunities is constant in our problem because CSMA/CA makes these decisions independent from our routing and internode scheduling decisions. Then, after node i is given opportunity for transmission by CSMA/CA, we determine the best link $i-j$ to activate. Thus, the sum of the percentages of per-link activations, i.e., $\sum_{j\in\mathcal{N}}\tau_{i,j}$, should be less than the percentage of constant node activation, i.e., γ_i, as shown in the last constraint of (16). Also, the percentages of link activations can be translated into the link transmission rates as shown in the fourth constraint of (16).

Diff-subMax—Decomposed Solution: If we solve the NUM problem in (16), we get the same flow control, and routing problems as in Diff-Max, as we also explained in Section IV-B. In particular, the flow control part solves

$$x_s = (g_s')^{-1}\left(u_{o(s)}^s\right) \tag{17}$$

as in (8), and the routing part solves

$$\max_{\boldsymbol{f}} \sum_{(i,j)\in\mathcal{L}}\sum_{s\in\mathcal{S}} f_{i,j}^s\left(u_i^s - u_j^s - v_{i,j}\right) \tag{18}$$

as in (10). On the other hand, the scheduling part changes as it solves

$$\begin{aligned}
&\max_{\boldsymbol{h}} \sum_{(i,j)\in\mathcal{L}} v_{i,j}h_{i,j}\\
&\text{s.t. } h_{i,j} \le (1-\bar{p}_{i,j})\bar{R}_{i,j}\tau_{i,j} \quad \forall i\in\mathcal{N}, j\in\mathcal{N}\\
&\quad \sum_{j\in\mathcal{N}} \tau_{i,j} \le \gamma_i \quad \forall i\in\mathcal{N}.
\end{aligned} \tag{19}$$

This problem is expressed as

$$\begin{aligned}
&\max_{\boldsymbol{h}} \sum_{(i,j)\in\mathcal{L}} v_{i,j}(1-\bar{p}_{i,j})\bar{R}_{i,j}\tau_{i,j}\\
&\text{s.t.} \sum_{j\in\mathcal{N}} \tau_{i,j} \le \gamma_i \quad \forall i\in\mathcal{N}.
\end{aligned} \tag{20}$$

This problem is equivalent to determining the link l^* according to $l^* = \arg\max_{[j\in\mathcal{N}_i]}\omega_{i,j}$, where $\omega_{i,j} = v_{i,j}(1-\bar{p}_{i,j})\bar{R}_{i,j}$. This solution is the deterministic version of what we proposed to implement in Section IV-B. Next, we discuss the Lagrange multipliers of Diff-subMax.

Diff-subMax—Lagrange Multipliers: The Lagrange multipliers; u_i^s and $v_{i,j}$ are calculated using gradient descent

$$\begin{aligned}
u_i^s(t+1) &= \left\{ u_i^s(t) - \alpha_t\left[\sum_{j\in\mathcal{N}} f_{i,j}^s(t) \right.\right.\\
&\qquad \left.\left. - \sum_{j\in\mathcal{N}}(1-\bar{p}_{j,i})\bar{R}_{j,i}\tau_{j,i}^s(t) - x_s(t)1_{i=o(s)}\right]\right\}^+\\
v_{i,j}(t+1) &= \left\{ v_{i,j}(t) + \beta_t\left[\sum_{s\in\mathcal{S}} f_{i,j}^s(t) \right.\right.\\
&\qquad \left.\left. -(1-\bar{p}_{i,j})\bar{R}_{i,j}\tau_{i,j}(t)\right]\right\}^+
\end{aligned} \tag{21}$$

where t is the iteration number, α_t and β_t are the step-sizes of the gradient descent algorithm, the $^+$ operator makes the Lagrange multipliers positive, and $\tau_{j,i}^s$ is the percentage of time that link $j-i$ is used for transmitting packets from flow s.

Diff-subMax—Convergence to the Optimal Point: The convergence of the solution set (17), (18), (20), and (21) follows directly from the convergence of convex optimization problems through gradient descent [15], [28]. In particular, if $\lim_{t\to\infty}\alpha_t = 0, \sum_{t=0}^{\infty}\alpha_t = \infty$ and $\lim_{t\to\infty}\beta_t = 0, \sum_{t=0}^{\infty}\beta_t = \infty$, then the solution converges, i.e., $\lim_{t\to\infty}\|x(t) - x^*\| = 0$, where x^* is the optimal solution to (16). Note that the optimal solution of Diff-subMax could be smaller than the optimal solution of Diff-Max because Diff-Max also optimizes internode scheduling, while Diff-subMax uses CSMA/CA for internode scheduling. However, Diff-subMax still optimizes flow control, routing, and intranode scheduling, and its deterministic version converges to the utility-optimal operating point.

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. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[2] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, Mar. 1993.

[3] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.

[4] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[5] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "DiffQ: Practical differential backlog congestion control for wireless networks," in *Proc. IEEE INFOCOM*, Rio de Janerio, Brazil, Apr. 2009, pp. 262–270.

[6] U. Akyol *et al.*, "Joint scheduling and congestion control in mobile ad-hoc networks," in *Proc. IEEE INFOCOM*, Phoenix, AZ, USA, Apr. 2008, pp. 1292–1300.

[7] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key, "Horizon: Balancing TCP over multiple paths in wireless mesh network," in *Proc. ACM MobiCom*, San Francisco, CA, Sep. 2008, pp. 247–258.

[8] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proc. ACM IPSN*, Stockholm, Sweden, Apr. 2010, pp. 279–290.

[9] R. Laufer, T. Salonidis, H. Lundgren, and P. L. Guyadec, "XPRESS: A cross-layer backpressure architecture for wireless multi-hop networks," in *Proc. ACM MobiCom*, Las Vegas, NV, USA, Sep. 2011, pp. 49–60.

[10] M. Chiang, S. T. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[11] "The Network Simulator-ns-2, Version 2.35," [Online]. Available: http://www.isi.edu/nsnam/ns/

[12] C. E. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," IETF, RFC 3561, July 2003.

[13] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. SIGCOMM*, 1994, pp. 234–244.

[14] X. Lin, N. B. Schroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[15] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. IEEE Decision Control*, Dec. 2004, vol. 2, pp. 1484–1489.

[16] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *Proc. ACM SIGCOMM*, Portland, OR, USA, Sep. 2004, pp. 121–132.

[17] C. Steger, P. Radosavljevic, and J. P. Frantz, "Performance of IEEE 802.11b wireless LAN in an emulated mobile channel," in *Proc. IEEE VTC*, Orlando, FL, USA, Oct. 2003, pp. 1479–1483.

[18] L. Tassiulas, "Scheduling and performance limits of networks with constantly changing topology," *IEEE Trans. Inf. Theory*, vol. 43, no. 3, pp. 1067–1073, May 1997.

[19] N. Kahale and P. E. Wright, "Dynamic global packet routing in wireless networks," in *Proc. IEEE INFOCOM*, Kobe, Japan, Apr. 1997, pp. 1414–1421.

[20] M. Andrews *et al.*, "Providing quality of service over a shared wireless link," *IEEE Commun. Mag.*, vol. 39, no. 2, pp. 150–154, Feb. 2001.

[21] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[22] A. L. Stolyar, "Greedy primal dual algorithm for dynamic resource allocation in complex networks," *Queuing Syst.*, vol. 54, pp. 203–220, 2006.

[23] J. Liu, A. L. Stolyar, M. Chiang, and H. V. Poor, "Queue back-pressure random access in multihop wireless networks: Optimality and stability," *IEEE Trans. Inf. Theory*, vol. 55, no. 9, pp. 4087–4098, Sep. 2009.

[24] A. Sridharan, S. Moeller, and B. Krishnamachari, "Implementing backpressure-based rate control in wireless networks," in *Proc. ITA Workshop*, San Diego, CA, USA, Sep. 2008, pp. 341–345.

[25] A. Sridharan, S. Moeller, and B. Krishnamachari, "Making distributed rate control using Lyapunov drifts a reality in wireless sensor networks," in *Proc. WiOpt*, Berlin, Germany, Apr. 2008, pp. 452–461.

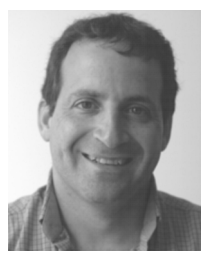
[26] E. Athanasopoulou, L. X. Bui, T. Ji, R. Srikant, and A. Stolyar, "Back-pressure-based packet-by-packet adaptive routing in communication networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 244–257, Feb. 2013.

[27] L. X. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1597–1609, Dec. 2011.

[28] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ: Prentice-Hall, 1989.

Hulya Seferoglu (S'04–M'11) received the B.S. degree in electrical engineering from Istanbul University, Istanbul, Turkey, in 2003, the M.S. degree in electrical engineering and computer science from Sabanci University, Istanbul, Turkey in 2005, and the Ph.D. degree in electrical and computer engineering from the University of California, Irvine, CA, USA, in 2010.

She is an Assistant Professor with the Electrical and Computer Engineering Department, University of Illinois at Chicago, Chicago, IL, USA. She worked as a Postdoctoral Associate with the Laboratory of Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA, USA, during 2011 to 2013. She worked as a summer intern with AT&T Labs Research, Florham Park, NJ, USA; Docomo USA Labs, Palo Alto, CA, USA; and Microsoft Research Cambridge, Cambridge, U.K., in 2010, 2008, and 2007, respectively. Her research interests are in the area of networking: design, analysis, and optimization of network protocols and algorithms. She is particularly interested in network optimization, network coding, and multimedia streaming.



Eytan Modiano (S'90–M'93–SM'00–F'12) received the B.S. degree in electrical engineering and computer science from the University of Connecticut at Storrs, Storrs, CT, USA, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, USA, in 1989 and 1992, respectively.

He was a Naval Research Laboratory Fellow between 1987 and 1992 and a National Research Council Post-Doctoral Fellow during 1992 to 1993. Between 1993 and 1999, he was with the Massachusetts Institute of Technology (MIT) Lincoln Laboratory, Lexington, MA, USA, where he was a project leader for MIT Lincoln Laboratory's Next Generation Internet (NGI) project. Since 1999, he has been on the faculty at MIT, Cambridge, MA, USA, where he is a Professor with the Department of Aeronautics and Astronautics and the Laboratory for Information and Decision Systems (LIDS). His research is on communication networks and protocols with emphasis on satellite, wireless, and optical networks.

Prof. Modiano is an Associate Fellow of the AIAA. He is an Editor-at-Large for the IEEE/ACM TRANSACTIONS ON NETWORKING, and served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY and IEEE/ACM TRANSACTIONS ON NETWORKING. He was the Technical Program Co-Chair for IEEE Wiopt 2006, IEEE INFOCOM 2007, and ACM MobiHoc 2007.