Exploring the Inefficiency and Instability of Back-Pressure Algorithms

Bo Ji, Changhee Joo, and Ness B. Shroff

***Abstract*—In this paper, we focus on the issue of stability in multihop wireless networks under flow-level dynamics, and explore the inefficiency and instability of the celebrated Back-Pressure algorithms. It has been well-known that the Back-Pressure (or MaxWeight) algorithms achieve queue stability and throughput optimality in a wide variety of scenarios. Yet, these results all rely on the assumptions that the set of flows is fixed, and that all the flows are long-lived and keep injecting packets into the network. Recently, in the presence of flow-level dynamics, where flows arrive and request to transmit a finite amount of packets, it has been shown that the MaxWeight algorithms may not guarantee stability due to channel fading or inefficient spatial reuse. However, these observations are made only for single-hop traffic, and thus have resulted in partial solutions that are limited to the single-hop scenarios. An interesting question is whether straightforward extensions of the previous solutions to the known instability problems would achieve throughput optimality in multihop traffic setting. To answer the question, we explore potential inefficiency and instability of the Back-Pressure algorithms, and provide interesting examples that are useful to obtain insights into developing an optimal solution. We also conduct simulations to further illustrate the instability issue of the Back-Pressure algorithms in various scenarios. Our study reveals that new types of inefficiencies may arise in the settings with multihop traffic due to underutilization of the link capacity or inefficient routing, and the stability problem becomes more challenging than in the single-hop traffic counterpart.**

I. Introduction

It has now been two decades since the seminal work of [1], which developed a joint routing and scheduling algorithm, called the Back-Pressure algorithm (or equivalently, the MaxWeight algorithm for single-hop traffic.) This algorithm is throughput-optimal, i.e., it can stabilize the network under any feasible load. The Back-Pressure algorithm computes the weight of a link as the maximum "back-pressure" (i.e., the queue-length or delay difference between the queues at the transmitting node and receiving node of the link for each flow) over all the flows, solves the well-known MaxWeight problem, and chooses a subset of non-interfering links that have the maximum weighted link-rate sum. Each chosen link then transmits packets of the flow that has the maximum "back-pressure" at the link.

B. Ji is with Department of ECE at the Ohio State University. C. Joo is with School of ECE at UNIST, Korea. N. B. Shroff is with Departments of ECE and CSE at the Ohio State University.

Emails: ji@ece.osu.edu, cjoo@unist.ac.kr, shroff@ece.osu.edu.

This work has been supported in part by the Army Research Office MURI Award W911NF-08-1-0238; the NSF grants CNS-1065136 and CNS-1012700; and the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science, and Technology under Grant No. 2012-0003227.

In more recent years, the Back-Pressure algorithm has gained enormous popularity, and its distinguishing property of guaranteeing queue stability and achieving optimal throughput has been extended to a wide variety of scenarios (see [2], [3] and references therein.) However, all of these fascinating results require certain assumptions on the traffic flows, i.e., the set of flows is fixed and all the flows are *long-lived* (i.e., they keep injecting packets into the network.) In practice, however, flows arrive and request to deliver a finite amount of packets, and are thus *short-lived*. In the presence of these short-lived flows, the well-known *last packet problem* [4] can occur under the queue-length-based Back-Pressure algorithm: a queue that lacks subsequent packet arrivals may not receive any service for a long time, since the queue-length-based algorithms give a higher priority to links with a larger queue length. More importantly, the queue-length-based Back-Pressure algorithm may not even be throughput-optimal in the presence of flow-level dynamics. This could occur because the number of flows may keep increasing with time, although each flow has a finite number of packets. In the following, we briefly discuss the known results on the inefficiency and instability of the Back-Pressure (or MaxWeight) algorithm, and their solutions.

In [5], the authors examine the potential instability of the MaxWeight algorithm in the presence of flow-level dynamics. Their clever counterexamples show that in a wireless downlink system with time-varying link rates, the MaxWeight algorithm may fail to achieve the optimal throughput performance. The inefficiency leading to instability essentially comes from failure to opportunistically exploit better link rates in environments with channel fading. They have also developed a solution based on a priori knowledge of the arrival and channel statistics. In [6], the authors propose the Workload-based Scheduling with Learning (WSL) algorithm, and prove that it is throughput-optimal under flow-level dynamics. In the WSL algorithm, all the short-lived flows are grouped into a virtual aggregate queue, whose backlog is measured as the total estimated workload, i.e., the number of time-slots required to transmit the remainder of a flow based on the best channel condition seen by the short-lived flow so far. The WSL algorithm makes scheduling decisions by comparing the backlog of the virtual aggregate queue with the maximum weight (i.e., product of the link rate and the queue length) over the long-lived flows. When the virtual aggregate queue dominates, WSL chooses to serve a short-lived flow that either sees its best channel condition or can completely deliver all of its remaining packets within the current time-slot, or arbitrarily picks a short-lived flow if no such flow exists.

When the queues for long-lived flows dominate, WSL simply runs the MaxWeight algorithm over the long-lived flows and chooses to serve the one that has the maximum weight. In [7], the same authors extend the WSL algorithm to a multi-channel system, and make it more practical than in [6] by removing the assumptions that the type (long-lived or short-lived) of a flow is known a priori, and that packets of a short-lived flow arrive all at once. In contrast to the above queue-length-based solutions, [8] proposes a delay-based MaxWeight algorithm, which provides an intuitive way around the last packet problem, successfully addresses the instability issue of its queue-length-based counterpart, and achieves the optimal throughput performance at no extra cost.

It should be noted that in the aforementioned instability results, rate variations (along with flow-level dynamics) play a critical role in leading to the MaxWeight algorithm being inefficient. Interestingly, the work of [9] provides another set of counterexamples to show that even without rate variations, instability of the MaxWeight algorithm can still occur, due to inefficient spatial reuse. This reveals that channel fading and rate variations are not the only causes of inefficiency and instability associated with flow-level dynamics. Further, since the solutions in [5]–[8] cannot be easily extended to the settings considered in [9], the authors develop a region-based MaxWeight scheduling algorithm to counter the instability effects. In the region-based MaxWeight algorithm, the entire network is partitioned into a finite number of regions. In each time-slot, the algorithm selects a subset of non-interfering regions based on the aggregate backlog in each region, and then serves a flow in each selected region. However, as mentioned by the authors themselves in [9], it is quite difficult for their solution to identify an adequate number of regions, without explicit knowledge of the traffic parameters.

Without a doubt the aforementioned results have opened up a new window, through which we can observe the flaws of the celebrated Back-Pressure algorithm. However, all of these known instability results and their remedies are established for limited cases of single-hop traffic only. It has largely been an open question whether new types of inefficiency and instability can occur in a broader setting of multihop traffic, and if the known remedies can be readily extended to these more general settings. Motivated by these questions, in this paper, we focus on the settings with multihop traffic, and explore new types of inefficiency and instability of the Back-Pressure algorithms by presenting interesting counterexamples in these settings.

Our contributions are summarized as follows. First, we focus on the inefficiency and instability in the settings of multihop traffic with fixed routes, and present two counterexamples to show that both queue-length-based and delay-based Back-Pressure algorithm may fail to guarantee stability in the presence of flow-level dynamics. The essential cause of the instability comes from inefficient schedule reuse, which is similar to the spatial inefficiency identified in [9]. Second, we focus on a wireless downlink system where relay-assisted 2-hop communications can be adopted to improve throughput performance. In this setting, we identify new types of inefficiencies by providing two counterexamples, in which the queue-length-based Back-Pressure algorithm is not throughput-optimal in the presence of flow-level dynamics. The instability issue comes from underutilization of the link capacity or inefficient routing due to insufficient paths information. We conduct numerical experiments to further illustrate the instability issue of the Back-Pressure algorithms in a variety of scenarios. Our investigation reveals that new types of inefficiencies and instability of the Back-Pressure algorithms can arise in multihop network traffic settings. *Moreover, to the best of our knowledge, this is the first work showing that not only inefficient scheduling but also inefficient routing may cause instability under flow-level dynamics.* Although it is perhaps very challenging to combat different types of inefficiencies in a unified solution, we believe that these examples provide useful insights for designing a unified optimal solution in a more general network setting.

The remainder of the paper is organized as follows. In Section II, we present the description of our system model. In Section III, we review the Back-Pressure algorithms that we investigate in this paper. Then, in Sections IV and V, we show that the Back-Pressure algorithm can lead to instability due to different types of inefficiencies in the settings with or without dynamic routing, respectively, by providing counterexamples along with numerical experiments. Finally, we conclude our paper in Section VI.

II. System Model

We consider a multihop wireless network with a single frequency channel. Time is assumed to be slotted. We use a directed graph $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$ to represent the network in time-slot t, where $\mathcal{V}(t)$ is the set of nodes and $\mathcal{E}(t)$ is the set of directed links. Nodes are wireless transmitters/receivers and links are wireless channels between two nodes if they can directly communicate with each other. The graph $\mathcal{G}(t)$ is time-varying in the presence of flow-level dynamics, as new users (or nodes) may join the network and depart after service.

Let $b(l)$ and $e(l)$ denote the transmitting node and receiving node of link $l = (b(l), e(l))$, respectively. We consider the binary symmetric interference model, i.e., for any links $l, k \in \mathcal{E}(t)$, if node $e(l)$ receives interference from $b(k)$, then node $e(k)$ also receives interference from node $b(l)$. Let $c_l(t)$ denote the link capacity/rate[1] of link l in time-slot t, i.e., link l can transmit at most $c_l(t)$ packets during time-slot t if none of the links that interfere with l is transmitting at the same time. Different from [5]–[8] where the instability relies on rate variations, we assume that link rates are fixed, i.e., $c_l(t) = c_l$ for all time-slots $t = 0, 1, 2, \cdots$ and for all links l. A set of links M is called a *feasible* schedule, if the interference constraints are satisfied, i.e., no two links in M interfere with each other. Let $\mathcal{M}(t)$ denote the set of all feasible schedules over $\mathcal{G}(t)$.

A flow is a stream of packets from a source node to a destination node. Packets are injected at the source, and

[1]We will use link capacity and link rate interchangeably throughout the paper.

traverse multiple links to the destination via hop-by-hop communications. Let $\mathcal{F}(t)$ denote the set of flows present in the system in time-slot t. A flow is "long-lived" (also called long flow for simplicity) if its source node keeps injecting packets into the network, and is "short-lived" (also called short flow for simplicity) if it has only a finite number of packets. A short flow will leave the network once all of its packets are successfully transmitted to the destination.

We assume that each node i maintains a First-In First-Out (FIFO) queue $Q_{i,f}$ for each flow[2] f. In a setting with fixed routes (e.g., Examples 1 and 2 in Section IV), each node only needs to maintain a FIFO queue for each flow passing through it. By slightly abusing the notation, we also let $Q_{i,f}(t)$ denote the queue length of $Q_{i,f}$ at the beginning of time-slot t. By convention, we set $Q_{i,f}(t) = 0$ if node i is the destination of flow f. We let $W_{i,f}(t)$ denote the waiting time (or delay) of the head-of-line (HOL) packet of $Q_{i,f}$ in the network, which is measured from the time when the HOL packet arrived to the source node of flow f until time-slot t. Let $i^-(f)$ denote the previous hop node for flow f before node i. We set $W_{i,f}(t) = W_{i^-(f),f}(t)$ if $Q_{i,f}(t) = 0$, and set $W_{i^-(f),f}(t) = 0$ if node i is the source node of flow f. The network is said to be *stable*, if the total number of packets in the system remains finite (i.e., the number of flows in the network remains finite, since we assume that every flow has a finite number of packets.)

III. Review of Back-Pressure Algorithms

In this section, we review the well-known Back-Pressure algorithm based on queue lengths (called Q-BP for simplicity) [1], and its delay-based counterpart (called D-BP for simplicity) for multihop traffic settings with *fixed routes* [4].

We start by describing the operations of the MaxWeight algorithm with generic link weights. Let $P_l(t)$ denote the weight of link l, then the MaxWeight algorithm chooses a feasible schedule M^* such that the weighted link-rate sum is maximized over $\mathcal{M}(t)$. That is,

$$M^* \in \operatorname*{argmax}_{M \in \mathcal{M}(t)} \sum_{l \in M} P_l(t) c_l. \tag{1}$$

Ties can be broken arbitrarily if there is more than one feasible schedule that has the same maximum weighted sum.

Next, we specify the link weight assignment rule for the Back-Pressure algorithms.

Q-BP: Let $\Delta Q_{l,f}(t)$ denote the *queue differential* at link l for flow f at the beginning of time-slot t. That is,

$$\Delta Q_{l,f}(t) \triangleq Q_{b(l),f}(t) - Q_{e(l),f}(t).$$

Then, we specify the weight of link l as

$$P_l(t) = \max\{\Delta Q_{l,f_l(t)}(t), 0\}, \tag{2}$$

where $f_l(t)$ is an arbitrary flow that has the maximum queue differential at link l in time-slot t, i.e.,

$$f_l(t) \in \operatorname*{argmax}_{f \in \mathcal{F}(t)} \Delta Q_{l,f}(t).$$

[2]The Back-Pressure algorithms will also work if each node maintains a FIFO queue for all the flows that share the same destination node.

D-BP: Define the delay $\hat{W}_{i,f}(t)$ as

$$\hat{W}_{i,f}(t) \triangleq W_{i,f}(t) - W_{i^-(f),f}(t),$$

and define the *delay differential* as

$$\Delta \hat{W}_{l,f}(t) \triangleq \hat{W}_{b(l),f}(t) - \hat{W}_{e(l),f}(t).$$

Then, we specify the weight of link l as

$$P_l(t) = \max\{\Delta \hat{W}_{l,f_l(t)}(t), 0\}, \tag{3}$$

where $f_l(t)$ is an arbitrary flow that has the maximum delay differential at link l in time-slot t, i.e.,

$$f_l(t) \in \operatorname*{argmax}_{f \in \mathcal{F}(t)} \Delta \hat{W}_{l,f}(t).$$

With the link weight $P_l(t)$ specified in (2) (resp. in (3)), Q-BP (resp. D-BP) solves the MaxWeight problem (1), and in time-slot t, schedules all links l in the chosen set M^* to transmit packets for flow $f_l(t)$ if $\Delta Q_{l,f_l(t)}(t) > 0$ (resp. if $\Delta \hat{W}_{l,f_l(t)}(t) > 0$).

Remark: Q-BP is a throughput-optimal solution to the joint problem of routing and scheduling in a more general setting. While D-BP has been shown to be throughput-optimal only in a special setting of fixed routes. It is still an open question whether delay-based scheduling algorithms like D-BP can achieve the optimal throughput jointly with dynamic routing [4].

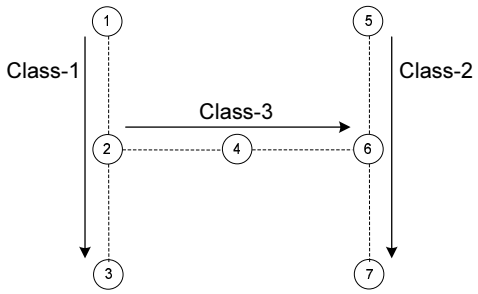
IV. Instability of Back-Pressure Algorithms with Fixed Routes

For ease of exposition only, throughout this section, we consider the *node-exclusive* interference model[3], where two links sharing a common node cannot be scheduled simultaneously. The examples of instability can be readily generalized to more general interference models. In this section, we assume that each flow has a single, fixed, and loop-free route, and that the Back-Pressure algorithms maintain per-flow queues, i.e., each node maintains a FIFO queue for every flow passing through the node. We will later allow dynamic routing as well as per-destination queueing in the counterexamples (Section V).

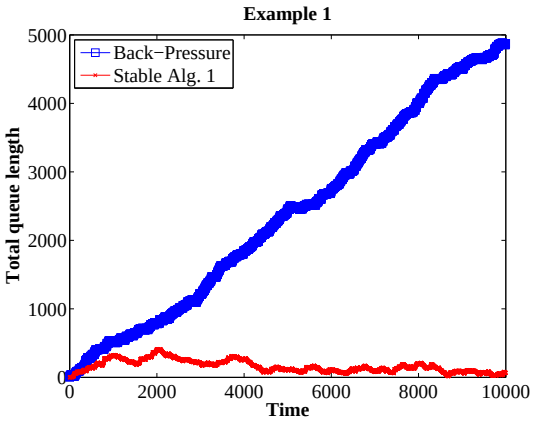
We provide two example networks, in which the Back-Pressure algorithms may not be throughput-optimal due to inefficient schedule reuse induced by certain traffic patterns. Along with each example, we also provide numerical results to further illustrate the instability issue, by comparing the scheduling performance of the Back-Pressure algorithm and that of a *stable* algorithm, which will be discussed in each example. Note that these stable algorithms are used to illustrate that the Back-Pressure algorithms cannot support certain feasible arrival rate vector (that can be supported by the stable algorithms), it does not necessarily mean that the stable algorithms are throughput-optimal, even in the particular examples that we consider.

Example 1 (Inefficient schedule reuse under Q-BP): We consider an "H"-type network topology as shown in

[3]It has been known as a good representation for Bluetooth or FH-CDMA networks [10], and is also called as *primary* or *1-hop* interference model.



(a) An "H"-type network topology with three classes of short flows.



(b) Comparison of Q-BP and Stable algorithm 1 (i.e., the Static Randomized scheduling algorithm as described in Example 1), with $B = 8$ and $\epsilon = \frac{B-6}{12B(2B-3)}$.

Fig. 1. Example 1: Instability of Q-BP due to inefficient schedule reuse.

Fig. 1(a), where every link has a unit link capacity. We assume that there are three classes of short flows (and no long flows). The routes of the flows are $(1 \to 2 \to 3)$ for Class-1, $(5 \to 6 \to 7)$ for Class-2, and $(2 \to 4 \to 6)$ for Class-3, respectively, each of which is represented by an arrow in Fig. 1(a). Class-1 and Class-2 flows arrive at node 1 and node 5 with a finite number of B packets (where $B > 6$), respectively, and Class-3 flows arrive at node 2 with one packet. The flow arrival process is as follows. At the beginning of each time-slot, with probability $\frac{1}{3B} - \epsilon$, two flows arrive simultaneously, one for Class-1 and the other for Class-2, at their respective sources, where $\epsilon \in (0, \frac{1}{3B})$ is a small real number. Also, with probability $\frac{1}{3} - \epsilon B$, one flow for Class-3 arrives at node 2, independently of the flow arrivals of the other two classes. We can calculate the arrival rate vector as $(\frac{1}{3} - \epsilon B)[1, 1, 1]$.

The arrival rate vector with any $\epsilon \in (0, \frac{1}{3B})$ is feasible, since every link can receive a service rate of $\frac{1}{3}$ under the *Static Randomized* scheduling algorithm, which activates each schedule in the set of $\{\{(1,2),(4,6)\}, \{(2,4),(5,6)\}, \{(2,3),(6,7)\}\}$ for $\frac{1}{3}$-fraction of time. Hence, the arrival rate vector $(\frac{1}{3} - \epsilon B)[1, 1, 1]$ is feasible for any ϵ and B such that $\epsilon < \frac{1}{3B}$.

However, the above arrival process with $\epsilon < \frac{B-6}{6B(2B-3)} < \frac{1}{3B}$ cannot be supported by the Q-BP algorithm. To see this, we observe that when two flows arrive simultaneously, say at time t, each for Class-1 and Class-2, neither $\max\{P_{(1,2)}(\tau), P_{(2,3)}(\tau)\}$ nor $\max\{P_{(5,6)}(\tau), P_{(6,7)}(\tau)\}$ is smaller than 2 for all $\tau \in [t, t + 2B - 6]$. This is because of the operations of Q-BP as well as the fact that the packets have to be forwarded via node 2 or node 6. On the other hand, neither of the links $(2, 4)$ and $(4, 6)$ has a weighted rate greater than 1, since all Class-3 flows have only one packet. Hence, in this period of $2B - 6$ time-slots, Q-BP selects a schedule from the set of $\{(1, 2), (5, 6)\}$, $\{(1, 2), (6, 7)\}$, $\{(2, 3), (5, 6)\}$, $\{(2, 3), (6, 7)\}$ only, and activates neither link $(2, 4)$ nor link $(4, 6)$. It is easy to see that if another flow pair of Class-1 and Class-2 arrive before $t + 2B - 6$, another $2B - 6$ time-slots would add up to the time interval during which links $(2, 4)$ and $(4, 6)$ cannot be served. Then for a large enough time period T, the total number of time-slots in which neither link $(2, 4)$ nor link $(4, 6)$ is activated is at least $(2B-6)(\frac{1}{3B}-\epsilon)T$. Thus, the summed service rate of links $(2, 4)$ and $(4, 6)$ is no greater than $1 - (2B - 6)(\frac{1}{3B} - \epsilon)$. Note that given the arrival rate of $\frac{1}{3} - \epsilon B$ at node 2, the summed service rate of links $(2, 4)$ and $(4, 6)$ needs to be at least $2(\frac{1}{3} - \epsilon B)$ so that the network is stable. Hence, the arrival rate vector cannot be supported under Q-BP if $2(\frac{1}{3} - \epsilon B) > 1 - (2B - 6)(\frac{1}{3B} - \epsilon)$, which occurs when $\epsilon < \frac{B-6}{6B(2B-3)}$.

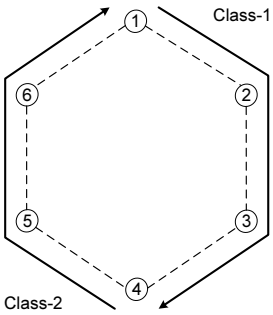
Numerical Experiment 1: We consider the system as described in Example 1, and set $B = 8$ and $\epsilon = \frac{B-6}{12B(2B-3)}$. We compare the performance of Q-BP and the Static Randomized scheduling algorithm as described in Example 1. We run the simulation for 10^4 time-slots for each algorithm, and plot the total queue length over time under both algorithms in Fig. 1(b).

The simulation results show that the Static Randomized scheduling algorithm keeps the queue length bounded. However, under the Q-BP algorithm, the total queue length keeps increasing with time, which implies instability.

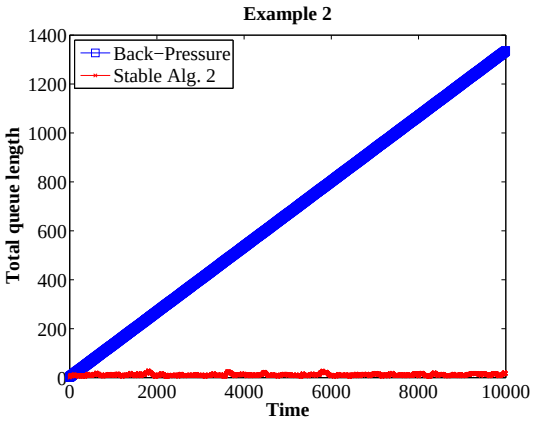
Remark: In the above example, the instability of Q-BP is essentially due to inefficient schedule reuse. The bursty arrivals force Q-BP to distribute the amount of time for each feasible schedule in an inefficient manner, which makes certain "regions" (e.g., links $(2, 4)$ and $(4, 6)$) receive insufficient amount of service. This type of inefficiency is similar to the inefficient spatial reuse identified in [9] for single-hop traffic.

As we mentioned in the introduction, the delay-based MaxWeight algorithm [8] has been developed to combat the instability of its queue-length-based counterpart. This delay-based remedy is simple and incurs no extra cost. However, it is developed only for countering the instability caused by failure to exploit wireless diversity from time-varying link rates associated with single-hop traffic. In the setting of multihop traffic and no link rate variations, it is unclear whether the delay-based algorithms can successfully solve the instability issue of their queue-length-based counterparts or not.

To answer the above question, we present the following example to show that the D-BP algorithm [4] can also result in instability issue in the presence of flow-level dynamics. To the best of our knowledge, D-BP is the only known throughput-



(a) A size-6 ring network topology with two classes of short flows.



(b) Comparison of D-BP and Stable algorithm 2 (i.e., the Static Randomized scheduling algorithm as described in Example 2).

Fig. 2. Example 2: Instability of D-BP due to inefficient schedule reuse.

optimal scheduling algorithm based on delay for multihop traffic without flow-level dynamics.

Example 2 (Inefficient schedule reuse under D-BP): We consider a size-6 ring network as shown in Fig. 2(a), where every link has a unit link capacity. We assume that there are two classes of short flows (and no long flows). The routes for the two classes of flows are $(1 \to 2 \to 3 \to 4)$ and $(4 \to 5 \to 6 \to 1)$, respectively, which are represented by the arrows in Fig. 2(a). We assume that each short flow arrives to the network with one packet and will leave the network when the packet is successfully delivered to its destination node.

We let A_1 and A_4 denote the number of flow arrivals at node 1 and node 4, respectively. Also, let $\hat{Q}_i$ denote the total number of packets (possibly from different flows) at node i *at the end* of each time-slot. We consider a specific traffic arrival process as shown in Table I, which also illustrates the queue-length evolution under D-BP. Specifically, the traffic arrival pattern repeats every 5 time-slots (starting from time-slot 1), and in each of the first two time-slots within each period, there is a concurrent flow arrival at both node 1 and node 4. Recall that D-BP does not activate any link with non-positive weight to transmit packets. For example, since the HOL delays are all 0 at the beginning of time-slot 1, none of the links are activated and no packet is transmitted. To help readers understand the queue-length dynamics in Table I, we provide in Tables II and III the weights (at the beginning of each time-slot) and the chosen links for the first 6 time-slots. In Table II, for ease of exposition, we show the weight of nodes 1, 2 and 3 for the Class-1 flows. The weight of nodes 4, 5 and 6 for the Class-2 flows can be obtained similarly. In Table III, a link has a "1" if the link is included in the schedule (i.e., it is activated to transmit packets) in the corresponding time-slot, and has a "0" otherwise. We call the flow that arrives at node 1 in time-slot 1 (resp. in time-slot 2) as flow 1 (resp. flow 2).

It is clear from Table I that if the same arrival pattern continues, the value of $\hat{Q}_1$ and $\hat{Q}_4$ will both increase by 1 every 15 time-slots, and will eventually become unbounded. On the other hand, the arrival rate vector for the above arrival process is $[\frac{2}{5}, \frac{2}{5}]$, which is feasible. To see this, we consider the *Static Randomized* scheduling algorithm that chooses

TABLE I
ARRIVAL PROCESS AND QUEUE-LENGTH DYNAMICS UNDER D-BP.

t	A_1,A_4	$\hat{Q}_1$,$\hat{Q}_2$,$\hat{Q}_3$,$\hat{Q}_4$,$\hat{Q}_5$,$\hat{Q}_6$
0	0 , 0	0 , 0 , 0 , 0 , 0 , 0
1	1 , 1	1 , 0 , 0 , 1 , 0 , 0
2	1 , 1	1 , 1 , 0 , 1 , 1 , 0
3	0 , 0	1 , 0 , 1 , 1 , 0 , 1
4	0 , 0	1 , 0 , 0 , 1 , 0 , 0
5	0 , 0	0 , 1 , 0 , 0 , 1 , 0
6	1 , 1	1 , 0 , 1 , 1 , 0 , 1
7	1 , 1	2 , 0 , 0 , 2 , 0 , 0
8	0 , 0	1 , 1 , 0 , 1 , 1 , 0
9	0 , 0	1 , 0 , 1 , 1 , 0 , 1
10	0 , 0	1 , 0 , 0 , 1 , 0 , 0
11	1 , 1	1 , 1 , 0 , 1 , 1 , 0
12	1 , 1	2 , 0 , 1 , 2 , 0 , 1
13	0 , 0	2 , 0 , 0 , 2 , 0 , 0
14	0 , 0	1 , 1 , 0 , 1 , 1 , 0
15	0 , 0	1 , 0 , 1 , 1 , 0 , 1
16	1 , 1	2 , 0 , 0 , 2 , 0 , 0
17	1 , 1	2 , 1 , 0 , 2 , 1 , 0
18	0 , 0	2 , 0 , 1 , 2 , 0 , 1
19	0 , 0	2 , 0 , 0 , 2 , 0 , 0
20	0 , 0	1 , 1 , 0 , 1 , 1 , 0
21	1 , 1	2 , 0 , 1 , 2 , 0 , 1
22	1 , 1	3 , 0 , 0 , 3 , 0 , 0
23	0 , 0	2 , 1 , 0 , 2 , 1 , 0
24	0 , 0	2 , 0 , 1 , 2 , 0 , 1
25	0 , 0	2 , 0 , 0 , 2 , 0 , 0

TABLE II
DELAY DYNAMICS UNDER D-BP (FOR NODES 1, 2, AND 3).

t	A_1	$W_{0,1}$,$W_{1,1}$,$W_{2,1}$,$W_{3,1}$,$W_{0,2}$,$W_{1,2}$,$W_{2,2}$,$W_{3,2}$	$W_{1,1}$,$W_{2,1}$,$W_{3,1}$,$W_{1,2}$,$W_{2,2}$,$W_{3,2}$
0	0	0 , 0 , 0 , 0 , 0 , 0 , 0 , 0	0 , 0 , 0 , 0 , 0 , 0
1	1	0 , 0 , 0 , 0 , 0 , 0 , 0 , 0	0 , 0 , 0 , 0 , 0 , 0
2	1	0 , 1 , 0 , 0 , 0 , 0 , 0 , 0	1 , 0 , 0 , 0 , 0 , 0
3	0	0 , 0 , 2 , 0 , 0 , 1 , 0 , 0	0 , 2 , 0 , 1 , 0 , 0
4	0	0 , 0 , 0 , 3 , 0 , 2 , 0 , 0	0 , 0 , 3 , 2 , 0 , 0
5	0	0 , 0 , 0 , 0 , 0 , 3 , 0 , 0	0 , 0 , 0 , 3 , 0 , 0

TABLE III
SCHEDULE DYNAMICS UNDER D-BP.

t	$(1,2)$,$(2,3)$,$(3,4)$,$(4,5)$,$(5,6)$,$(6,1)$
0	0 , 0 , 0 , 0 , 0 , 0
1	0 , 0 , 0 , 0 , 0 , 0
2	1 , 0 , 0 , 1 , 0 , 0
3	0 , 1 , 0 , 0 , 1 , 0
4	0 , 0 , 1 , 0 , 0 , 1
5	1 , 0 , 0 , 1 , 0 , 0

schedules $\{(1,2),(3,4),(5,6)\}$ and $\{(2,3),(4,5),(6,1)\}$ both for $\frac{1}{2}$-fraction of time, which results in a service rate of $\frac{1}{2}$ for every link. Clearly, the resultant service rate vector can support the arrival rate vector of $[\frac{2}{5},\frac{2}{5}]$.

Remark: Note that in the above example, we consider deterministic arrival process for ease of illustration. However, the example can be generalized to the case with stochastic arrival processes. For example, consider the arrival process as follows. In each time-slot, with probability $\epsilon > 0$, there is a flow arrival with one packet at both node 1 and node 4. In this case, any arrival rate vector with $\epsilon \in (\frac{1}{3},\frac{1}{2})$ is feasible, which, similarly, cannot be supported by D-BP due to the inefficient schedules it chooses.

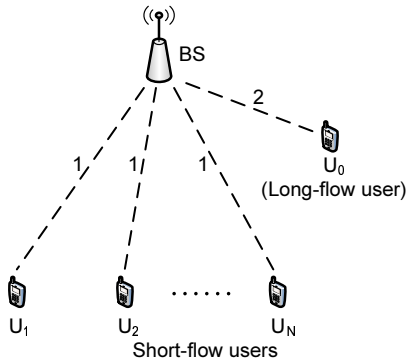
Numerical Experiment 2: We consider the system as described in Example 2, and consider the arrival process as specified in Table I. We compare the performance of D-BP and the Static Randomized scheduling algorithm described above. We run the simulation for 10^4 time-slots for each algorithm, and plot the total queue length over time under both algorithms in Fig. 2(b). Similarly as in the previous numerical experiment, the simulation results show that D-BP leads to instability.

Remark: In the above example, the instability of D-BP is also due to inefficient schedule reuse. However, unlike in Example 1, the packet arrivals are not bursty. Instead, the specific arrival pattern forces D-BP to always choose "small" schedules with two links rather than better schedules with three links. This type of scheduling inefficiency was also investigated in similar network topologies [4], [11], [12] for the greedy algorithms in the setting without flow-level dynamics.

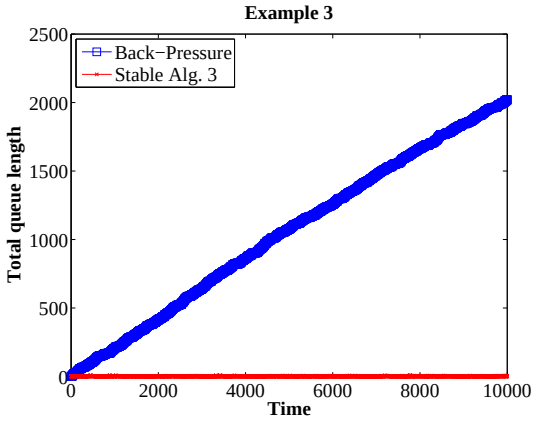
In the special cases of Examples 1 and 2, besides the Static Randomized scheduling algorithms that require knowledge of the arrival rates, a possible solution to the instability problem is to use per-destination queues at each node. We can do so in the scenarios where the network topology does not change with the new user arrivals. However, per-destination queueing may not help in the scenarios where new user arrivals could potentially change the network topology (as we will consider in Section V). Indeed, we will show that in such scenarios, the instability problem becomes more challenging, especially when routing needs to be integrated into the design as well.

V. Instability of Back-Pressure Algorithms with Dynamic Routing

In this section, we consider the downlink of a single-cell wireless network, where mobile users arrive to the network for downloading packets from the Base Station (BS). We assume that relay-assisted 2-hop communications can be adopted to improve throughput performance. We show through three examples that new types of inefficiencies can arise for the Back-Pressure algorithms, in particular, when some active link is underutilized or routing has to be taken into account. Similarly as in Section IV, along with each example, we also illustrate the instability issue with a numerical experiment that compares the scheduling performance of the Back-Pressure algorithms and that of a *stable* algorithm.



(a) A single-cell wireless downlink system with a long-flow user and multiple short-flow users.



(b) Comparison of MaxWeight and Stable algorithm 3 (i.e., the WSL algorithm), under the arrival rate vector of $[p_l, p_s] = [0.8, 0.4]$.

Fig. 3. Example 3: Instability of the MaxWeight algorithm due to underutilization of the link capacity.

To begin with, we first consider a single-hop traffic scenario, and show in Example 3 that instability of the MaxWeight algorithm can arise due to underutilization of the link capacity in the presence of flow-level dynamics. Then, in Example 4, we further show that this type of inefficiency can also occur in multihop traffic scenarios, which becomes even harder to combat.

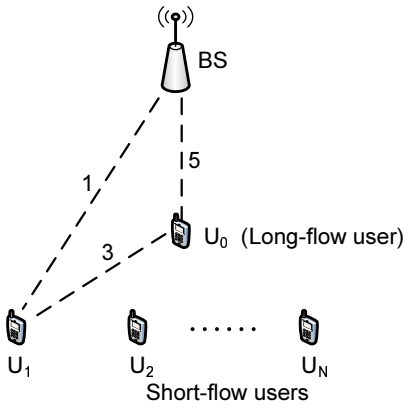
Example 3 (Link capacity underutilization (single-hop)): We consider a wireless downlink system with a single BS that needs to transmit packets to multiple mobile users. As shown in Fig. 3(a), user U_0 has a long flow and is called a long-flow user, and each U_i for $i \geq 1$ has a short flow and is called a short-flow user. The interference constraint is such that in each time-slot, only one link can be activated. The link capacity is labeled beside each link in Fig. 3(a). Specifically, the link capacity is 1 for each link (BS, U_i) with $i \geq 1$, and is 2 for link (BS, U_0), respectively. The arrival process is as follows. At the beginning of each time-slot, a short-flow user arrives with probability p_s and each user has one packet for downloading. For the long-flow user U_0, one packet arrives with probability p_l at the beginning of each

time-slot, independently of the short flow arrivals. Node U_i appears in the network topology when the short flow arrives for user U_i, and disappears once it successfully downloads all of its packets from the BS. Hence, in this system the network topology is time-varying due to flow-level dynamics.

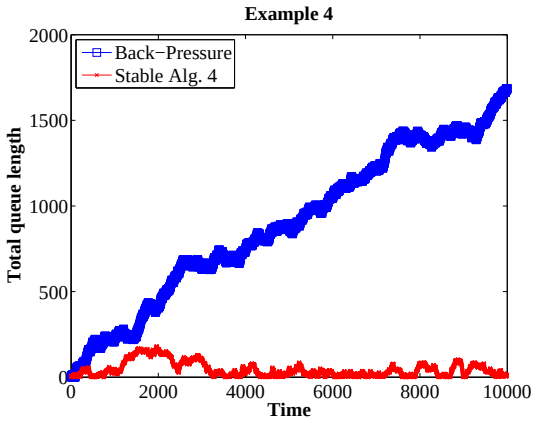
We now consider the MaxWeight algorithm. Whenever the long-flow user U_0 has one packet arrival, link (BS, U_0) will be activated to transmit the packet for U_0, as its weighted rate is no smaller than 2, which is greater than the weighted rate of the other links: the weighted rate at link (BS, U_i) is at most 1 for $i \geq 1$. Only when the queue for the long flow is empty, the MaxWeight algorithm will choose a link (BS, U_i) for some $i \geq 1$ and the BS transmits the packet to U_i. This implies that any arrival rate vector with $p_l + p_s > 1$ cannot be supported under the MaxWeight algorithm. However, we will show that any arrival rate vector satisfying $\frac{1}{2}p_l + p_s < 1$ is feasible. To see this, we consider the Workload-based Scheduling with Learning (WSL) algorithm developed in [6], where the workload is the number of time-slots required to completely serve the remainder of a short flow. Recall that WSL uses a virtual aggregate queue for all the short flows at the BS, and in this specific case, makes scheduling decisions by comparing the backlog (i.e., the summed workload of all the short flows) of the aggregate queue and the weight of the long flow. When the aggregate queue is chosen, the BS picks one link (BS, U_i) for some $i \geq 1$ and transmits the packet to U_i, which completes the short flow for U_i, and otherwise, the BS chooses to serve the long user. Due to results of [6], the arrival rate vector satisfying $\frac{1}{2}p_l + p_s < 1$ can be supported by WSL, and is thus feasible. An example of feasible arrival rate vectors is $[p_l, p_s] = [0.8, 0.4]$, which, however, cannot be supported by the MaxWeight algorithm.

Numerical Experiment 3: We consider the system as described in Example 3. We compare the performance of Q-BP and the WSL algorithm under the arrival rate vector of $[p_l, p_s] = [0.8, 0.4]$. We run the simulation for 10^4 time-slots for each algorithm, and plot the total queue length over time under both algorithms in Fig. 3(b). The simulation results show that the WSL algorithm keeps the queue length bounded. However, under the Q-BP algorithm, the total queue length keeps increasing with time, which implies instability.

Remark: In the above example, the instability of the MaxWeight algorithm is essentially due to underutilization of the link capacity. The specific arrival pattern forces the MaxWeight algorithm to serve first the long-flow user with only one packet arrival while it can wait until it has at least two packets. This is different from the known causes of inefficiency identified in [5] and [9], as well as those in Examples 1 and 2. That is, Example 3 relies on neither rate variations nor selection of inefficient schedules. Note that in the scenarios without flow-level dynamics, such type of inefficiency could also happen. Without flow-level dynamics, however, this inefficiency can only occur occasionally and does not lead to instability, because an unserved queue of any long flow will eventually build up, and will dominate the weight of a queue with insufficient packets.



(a) A single-cell wireless downlink system with relay-assisted 2-hop communications. Note that the link between U_i and U_0 and the link between U_i and the BS are similar to those for U_1, and are thus not displayed.



(b) Comparison of Q-BP and Stable algorithm 4 (i.e., the priority algorithm that gives a priority to the short flows), under the arrival rate vector of $[p_l, p_s] = [0.5, 0.28]$.

Fig. 4. Example 4: Instability of Q-BP due to underutilization of the link capacity.

In the following example, we revisit the inefficiency due to underutilization of the link capacity, but with multihop traffic.

Example 4 (Link capacity underutilization (multihop)): We consider a similar single-cell wireless downlink system as in Example 3. The key difference from Example 3 is that one can exploit the relay-assisted 2-hop communications to improve throughput performance. As shown in Fig. 4(a), there is one path between the BS and the long-flow user U_0, and there are two paths between the BS and each short-flow user U_i for $i \geq 1$: direct communications or relay-assisted 2-hop communications (i.e., using mobile user U_0 as a relay node). The interference constraint is such that in each time-slot, only one link, either between the BS and a mobile user or between two mobile users, can be activated. The link capacity is labeled beside each link in Fig. 4(a), i.e., $c_{(BS,U_0)} = 5$ (packets per time-slot), $c_{(BS,U_i)} = 3$ and $c_{(U_0,U_i)} = 1$ for all $i \geq 1$. The arrival process is as follows. At the beginning of each time-slot, a short-flow user arrives with probability

p_s and each user has three packets for downloading from the BS. For the long-flow user U_0, four packets arrive at the BS in a burst with probability p_l at the beginning of each time-slot, independently of the short flow arrivals.

Note that in this example, each short flow completes its transmission within 2 time-slots if using relay communications via user U_0. On the other hand, it needs 3 time-slots if using the direct communication link. Hence, one cannot achieve the optimal throughput unless by exploiting the better paths of relay communications. In the following, we show that the Q-BP algorithm indeed exploits the better paths of relay communications. Yet, the operations of Q-BP may be inefficient due to underutilization of the link capacity, and lead to instability.

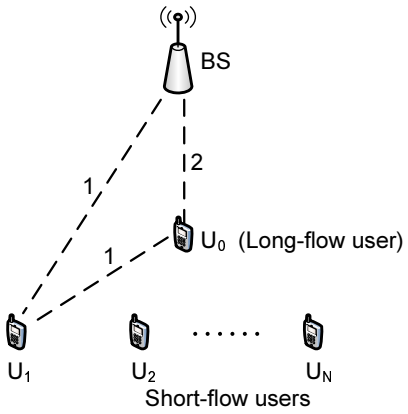
We now consider the Q-BP algorithm with *dynamic routing*. In this case, Q-BP still uses the same algorithm specified in Section III, except that flow routes need to be dynamically chosen. Whenever there is a burst of four packet arrivals for the long-flow user U_0, link (BS, U_0) will be activated to transmit packets for U_0, as its weighted rate is $(4-0)\times 5 = 20$, which is greater than the weighted rate of the other links: the weighted rate for a short flow at link (BS, U_0) is at most $(3-0)\times 5 = 15$, the weighted rate at link (BS, U_i) is at most $(3-0)\times 1 = 3$, and the weighted rate at link (U_0, U_i) is at most $(3-0)\times 3 = 9$, respectively. When a short-flow user U_i arrives and the queue at the BS for the long flow is empty, Q-BP will route the packets of the short flow to U_0, as link (BS, U_0) has a larger weighted rate of 15 versus the weighted rate of 3 at link (BS, U_i). Hence, it needs 2 time-slots to successfully transmit all the three packets from the BS to U_i. This implies that any arrival rate vector such that $p_l + 2p_s > 1$, cannot be supported under Q-BP. However, we will show that any arrival rate vector satisfying $\frac{4}{5}p_l + 2p_s < 1$ is feasible. To see this, we consider a policy that gives a priority to short flows. Specifically, when there is at least one short-flow user in the system, the policy chooses to serve a short flow U_i for some $i \geq 1$, by transmitting the three packets either from the BS to U_0 or from U_0 to U_i. Hence, each short flow requires 2 time-slots to completely receive the three packets. Note that in each time-slot, there is a short flow arrival with probability p_s. Then, the fraction of time remaining for U_0 to download its own packets is $1-2p_s$. Therefore, the arrival rate vector is feasible if $4p_l < 5(1-2p_s)$, since each burst of arrivals has 4 packets and the link (BS, U_0) has a capacity of 5. For example, $[p_l, p_s] = [0.5, 0.28]$ yields a feasible arrival rate vector, which, however, cannot be supported by Q-BP.

Numerical Experiment 4: We consider the system as described in Example 4. We compare the performance of Q-BP and the priority algorithm that gives a priority to short flows, under the arrival rate vector of $[p_l, p_s] = [0.5, 0.28]$. We run the simulation for 10^4 time-slots for each algorithm, and plot the total queue length over time under both algorithms in Fig. 4(b). Similarly as in the previous numerical experiment, the simulation results show that Q-BP leads to instability.

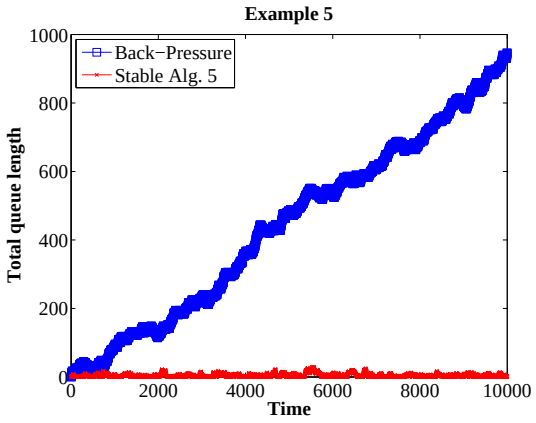
Remark: In the above example, the type of inefficiency is the same as in Example 3, i.e., due to underutilization of the link capacity. However, it is unclear how to extend the WSL policy to the multihop traffic scenarios with dynamic routing. Moreover, in the above example, one can further improve throughput performance by letting the BS forward five packets from different flows to U_0. Also, to prevent the inefficiency from occurring at links (U_0, U_i), node U_0 should not forward any packets to U_i until U_0 receives all the three packets for U_i from the BS.

In this example, we revisit the inefficiency due to underutilization of the link capacity in the multihop traffic scenarios, and show that it becomes more difficult to address the same type of inefficiency for multihop traffic than for the single-hop counterpart. In the following example, we show that even if the capacity of an activated link is fully utilized, Q-BP may still result in instability. This is caused by inefficient routing due to insufficient paths information.

Example 5 (Inefficient routing): We consider a wireless downlink system similar to that in Example 4, where there are two paths (direct communications and relay communications via user U_0) between the BS and each short-flow user U_i for



(a) A single-cell wireless downlink system with relay-assisted 2-hop communications. Note that the link between U_i and U_0 and the link between U_i and the BS are similar to those for U_1, and are thus not displayed.



(b) Comparison of Q-BP and Stable algorithm 5 (i.e., the WSL algorithm with short flows using direct communications), under the arrival rate vector of $[p_l, p_s] = [0.5, 0.2]$.

Fig. 5. Example 5: Instability of Q-BP due to inefficient routing.

$i \geq 1$. The link capacity is 2, 1 and 1, for link (BS, U_0), links (BS, U_i), and links (U_0, U_i), for $i \geq 1$, respectively, as shown in Fig. 5(a). We consider the following traffic arrival process: At the beginning of each time-slot, with probability p_s, a short-flow user arrives with two packets for downloading; the long user has a burst of two packet arrivals with probability p_l, independently of the short flow arrivals.

Under Q-BP, when the BS needs to transmit the packets for a short-flow user U_i, it will use the relay-assisted path since link (BS, U_0) has a larger link rate and thus a larger weighted rate than link (BS, U_i). Once it chooses link (BS, U_0), the two packets for user U_i will be forwarded to U_0, and need two additional time-slots to be forwarded to U_i. Hence, it requires a total of three time-slots for U_i to finish downloading. This implies that any arrival rate vector with $p_l + 3p_s > 1$ cannot be supported under Q-BP. However, we observe that it needs only two time-slots for U_i to finish downloading the two packets from the BS if it uses the path of direct communication, i.e., via link (BS, U_i). Hence, any arrival rate vector satisfying $p_l + 2p_s < 1$ is indeed feasible if the short flows are restricted to use static routing with direct communications (combined with the WSL scheduling algorithm.) For example, an arrival rate vector with $[p_l, p_s] = [0.5, 0.2]$ is feasible, but it cannot be supported by Q-BP.

Numerical Experiment 5: We consider the system as described in Example 5. We compare the performance of Q-BP and the WSL algorithm [6] associated with a static routing algorithm that restricts the short flows to choose direct communication path only, under the arrival rate vector of $[p_l, p_s] = [0.5, 0.2]$. We run the simulation for 10^4 time-slots for each algorithm, and plot the total queue length over time under both algorithms in Fig. 5(b). Similarly as in the previous numerical experiment, the simulation results show that Q-BP leads to instability.

Remark: In the above example, the instability essentially comes from inefficient routing due to insufficient paths information. Specifically, the link-rate heterogeneity leads Q-BP to choose a path that looks better, while is actually worse. This type of inefficiency can be prevalent for the dynamic routing schemes (e.g., Q-BP) in the presence of flow-level dynamics, and is completely different from any known types of inefficiencies that occur in the regime of link scheduling.

In the special cases of Examples 3 and 5, the inefficiency can be readily fixed. For example, in Example 3, an easy fix is to use the WSL algorithm [6], and in Example 5, one can simply choose direct communication link for the short-flow users and applies the WSL algorithm for scheduling. In Example 4, however, a more sophisticated algorithm needs to be developed for achieving throughput optimality. Further, a more challenging question is how to develop a unified solution to combat different types of inefficiencies in more general settings with multihop traffic and dynamic routing.

VI. Conclusion and Discussions

In this paper, we investigated the stability issue of the Back-Pressure algorithms in the presence of flow-level dynamics. Different from the previous works, we considered a more general setting of multihop traffic, and identified different types of inefficiencies. Although flow-level dynamics plays a critical role in the instability problem, there are many different types of inefficiencies that cause instability in multihop traffic scenarios. This makes it difficult to develop a unified solution that achieves the optimal throughput performance in a general network setting. We now summarize the known types of inefficiencies, including the new ones identified in this paper, as follows.

- Failure to opportunistically exploit better link rates.
- Inefficient schedule or spatial reuse.
- Underutilization of the link capacity.
- Inefficient routing due to insufficient paths information.

It is a very interesting problem to study the stability issue of the Back-Pressure algorithms in the presence of flow-level dynamics, and to identify potential causes of inefficiency in a more general setting. We believe that based on in-depth understanding of the essential elements of instability associated with flow-level dynamics, a unified optimal solution (preferentially perhaps, of back-pressure-type) that supports short-lived flows as well as long-lived flows can be developed.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[2] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[3] L. Georgiadis, M. Neely, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[4] B. Ji, C. Joo, and N. B. Shroff, "Delay-Based Back-Pressure Scheduling in Multi-Hop Wireless Networks," *IEEE/ACM Transactions on Networking*, 2012, To appear.

[5] P. van de Ven, S. Borst, and S. Shneer, "Instability of MaxWeight Scheduling Algorithms," in *The 28th IEEE International Conference on Computer Communications (INFOCOM)*, 2009, pp. 1701–1709.

[6] S. Liu, L. Ying, and R. Srikant, "Throughput-optimal opportunistic scheduling in the presence of flow-level dynamics," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 4, pp. 1057–1070, 2011.

[7] ——, "Stability with file arrivals and departures in multichannel cellular wireless networks," *Queueing Systems*, vol. 69, no. 3–4, pp. 259–291, December 2011.

[8] B. Sadiq and G. de Veciana, "Throughput optimality of delay-driven MaxWeight scheduler for a wireless system with flow dynamics," in *Proceedings of the 47th Annual Conference on Communication, Control and Computing (Allerton)*, 2009.

[9] P. van de Ven, S. Borst, and L. Ying, "Spatial inefficiency of MaxWeight scheduling," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE, 2011, pp. 62–69.

[10] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proceedings of the annual international conference on Mobile computing and networking (MobiCom)*. ACM New York, NY, USA, 2006, pp. 227–238.

[11] C. Joo, X. Lin, and N. B. Shroff, "Greedy Maximal Matching: Performance Limits for Arbitrary Network Graphs Under the Node-exclusive Interference Model," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2734–2744, 2009.

[12] ——, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1132–1145, 2009.