# No Packet Left Behind: Avoiding Starvation in Dynamic Topologies

Shay Vargaftik, *Student Member, IEEE,* Isaac Keslassy, *Senior Member, IEEE,* and Ariel Orda *Fellow, IEEE*

***Abstract*—Backpressure schemes are known to stabilize stochastic networks through the use of congestion gradients in routing and resource allocation decisions. Nonetheless, these schemes share a significant drawback, namely: the delay guarantees are obtained only in terms of average values. As a result, arbitrary packets may never reach their destination due to both the starvation and last-packet problems. These problems occur because in backpressure schemes, packet scheduling needs a subsequent stream of packets to produce the required congestion gradient for scheduling.**

**To solve these problems, we define a *starvation-free stability* criterion that ensures a repeated evacuation of all network queues. Then, we introduce SF-BP, the first backpressure routing and resource allocation algorithm that is *starvation-free stable*. We further present stronger per-queue service guarantees and provide tools to enhance weak streams. We formally prove that our algorithm ensures that all packets reach their destination for wide families of networks. Finally, we verify our results by extensive simulations using challenging topologies as well as random static and dynamic topologies.**

***Keywords*—Backpressure, starvation-free stability, throughput optimality, Lyapunov stability.**

## I. Introduction

### A. Background

The *Queue-length-based BackPressure* (Q-BP) routing and resource allocation technique is known to maximize network throughput. Since its introduction in [28], Q-BP has gained much popularity in the context of *dynamic networks*, and includes a vast collection of comprehensive theoretical as well as practical works in the literature, e.g. [3], [5]–[13], [16]–[19], [21], [22], [24], [26]–[35]. In such networks, the traditional two-phase algorithms [25], in which at first paths are being discovered and only then they are used to send data, may be less effective, as by the time these paths are discovered the topology changes again. This may eventually lead to an accumulation of packets in the network and even to instability, due to infinite looping, poor use of the network resources, and large overhead in maintaining the routes.

Instead, Q-BP attempts to exploit congestion gradients in order to achieve maximum network throughput. To do so, it assumes slotted time, and at each time slot, makes routing and resource-allocation decisions based on the maximization of the sum of the queue-length-differentials multiplied by the corresponding link capacities. It can be proved to achieve stability by showing a negative drift on a quadratic Lyapunov function that represents the sum of squared queue lengths.

The authors are with the Technion, Israel. E-mail: {shayvar@tx, isaac@ee, ariel@ee}.technion.ac.il.

### B. Related Work

Despite achieving stability within the entire capacity region of the network, Q-BP schemes also share a significant drawback, namely: the delay guarantees are obtained only in terms of average values. As a result, weak and infrequent streams (i.e., streams with a lower arrival rate) may suffer from *starvation* due to stronger streams (i.e., streams with a higher arrival rate) that produce larger congestion gradients. In addition, Q-BP schemes suffer from the *last-packet* problem, wherein the last packet of a stream may experience a potentially unbounded delay because there are no subsequent packets that could provide the needed queue-length-based congestion gradient to get this packet scheduled for transmission. Therefore, in Q-BP schemes, packets may never reach their destination due to both the *starvation* and the *last-packet* problems [11].

Many delay-reduction techniques have been introduced, such as [3], [9], [10], [34]. However, while these solutions may reduce the mean delay in the network, they hold no promises regarding individual packet arrivals. Namely, individual packets may still suffer from extremely long delays and even fail to reach their destination. In recent years, it has increasingly appeared that a *delay-based* backpressure technique would be a natural solution to solve these issues [11], [21]. The idea behind such a technique would be to make routing and resource-allocation decisions based on explicit delay information, such as the delay of the Head-of-Line (HoL) packet of each queue in the network. For instance, [21] proposed a delay-based technique for single-hop networks and achieved a worst-case delay bound, by discarding a small portion of the packets. However, this technique does not apply to multi-hop networks. A delay-based scheduling scheme was also proposed in [11]. However, it assumes fixed routes, and thus does not address the natural scenarios of dynamic networks where the topology changes and routes must be adapted to fully utilize the network. Newer works [14], [15] have suggested a delay-based *scheduling* for multi-hop dynamic networks. However, their approach does not provide routing, which presents a significant challenge in dynamic topologies.

Recently, [2] suggested a delay enhancement for encounter-based networks. Their idea is to use an additional set of duplicate buffers at each node for duplicate packets with a timeout mechanism to remove those packets from the network. However their approach does not offer tools to enhance weak and infrequent flows and does not provide guarantees regarding individual packet arrivals. [4] has proposed a technique with a queue-dependent bias function embedded into the backpressure term to improve the delay performance in the network. While their approach reduces the mean delay in the network, weak

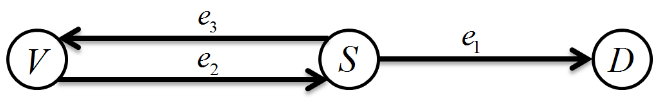


Fig. 1: Configuration with a single source $S$ sending data packets to a single destination $D$, illustrating *starvation*.

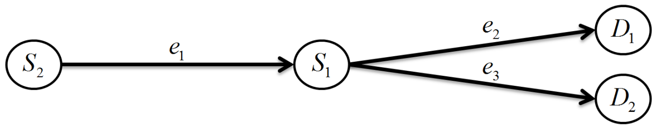


Fig. 2: Configuration with two sources $S_1$ and $S_2$ sending data packets to their respective destinations $D_1$ and $D_2$, illustrating the *last-packet problem*.

and infrequent flows may suffer even further from such policies that prefer large flows over small ones. [23] has suggested a loop-free backpressure routing using a link-reversal technique, but formally considers only a single-commodity in a static network. [30] has proposed stable user-defined scheduling to the network flows, but does not consider routing. Therefore, the establishment of a general delay-based backpressure routing and resource allocation technique for dynamic multi-hop networks, that maintains the desired $100\%$ throughput guarantee while giving guarantees about *individual* packet arrivals, remains an open problem.

### C. Motivation

We proceed to provide some insight regarding the fundamental challenges to the queue-length-based backpressure techniques. To do so, we present two instructive examples.

(1) To demonstrate *starvation*, consider the network presented in Figure 1. Assume that at each time slot, the source node $S$ exogenously injects a single data packet to the network. The link capacities satisfy $C(e_2) = C(e_3) = 1$, while $C(e_1)$ is time-dependent and either equals 1 or 100 with equal probability at each time slot. At the start, during the first time slot, a single data packet will be sent from node $S$. Since there are two allocated links during the first time slot (links $(S, V)$ and $(S, D)$), and these links are indistinguishable to the backpressure algorithm, this data packet may be sent to node $V$. But the queue-length-differential between nodes $V$ and $S$ will never become positive, because node $S$ will always hold one new packet at each time slot. Therefore, the packet in node $V$ will never come back using link $e_2$, and will never reach its destination $D$. Note that the network is heavily under-utilized, since its arrival rate could increase to more than 50 packets per time slot while still maintaining stability. Therefore, the network experiences *starvation* even though it is quite under-utilized.

(2) To demonstrate the *last-packet problem*, consider the network presented in Figure 2. $S_1$ and $S_2$ exogenously inject a single data packet each at each time slot, destined to $D_1$ and $D_2$ respectively. Moreover, the link capacities satisfy $C(e_1) = C(e_3) = 1$ and $C(e_2) = 10$. Assume that the links $e_2$ and $e_3$ are in different activation sets, and therefore cannot be activated in the same time slot. Further assume that the exogenous arrival of packets to source node $S_2$ stops after a finite positive number of time slots.

Using the same arguments as above, the last remaining data packets that arrived to $S_1$ from $S_2$ will never reach their destination. The reason is that the queue-length-differential between $S_1$ and $D_1$ multiplied by $C(e_2)$ never drops below 10. Thus, when the number of packets residing at $S_1$ and destined to $D_2$ is below 10, with no additional arrival process to increase the congestion gradient, these packets will *never* reach their destination. Note that, once again, the network is under-utilized, yet the *last-packet problem* still happens.

Note that the starvation and the last-packet problems can occur even in a static network with loop-free routing. Here, we provided examples that involve routing loops and activation sets to correspond with our system model that allows both.

Unfortunately, existing techniques provide only a partial solution. For instance, using either fixed priorities or an additive bias [18] cannot solve the *starvation* and the *last-packet* problems in general, since in order to set suitable parameters, all the topology states and arrival rates to the sources need to be known in advance. Furthermore, in order for a small number of packets to be scheduled in the presence of a much stronger stream, an extremely high priority or additive bias should be set. Such a choice will result in a significant and unnecessary growth of the delays of the other streams. Another possible approach is to rely on higher virtual arrival rates [3], [6]. Unfortunately, this will only solve the *starvation* problem, as the evacuation of packets can be guaranteed using this method, albeit at the cost of a permanent increase of the delays of all other commodities, since it will require a higher bandwidth across all links permanently even when it is not needed. However, the *last-packet problem* remains, because the additional virtual rate cannot be stopped in order to guarantee the evacuation of the last remaining packets in the network. Additionally, existing delay-reduction techniques such as [3], [9], [10], [34] naturally favor the stronger available links and the bigger queue-length differentials, since they strive to reduce the mean delay of packets. Therefore, they also do not solve the *starvation* and the *last-packet* problems in cases such as in our examples.

The scenarios with the *starvation* and the *last-packet* problems are characterized by lonely packets that are trapped due to the lack of a subsequent stream that could *push* them, i.e. supply the needed queue-length-differential to get them scheduled. These observations lead to our main intuition: in order to solve the *starvation* and the *last-packet* problems without causing an unnecessary and permanent growth of delays, *each packet should have the ability to be scheduled independently of a subsequent stream.*

### D. Contributions and paper outline

Accordingly, we define a stronger, delay-based, stability criterion. We term it *starvation-free stability*, and establish the first backpressure routing and resource allocation algorithm for multi-hop dynamic networks that is *starvation-free stable*. We further present stronger *per-queue* service guarantees and provide tools to enhance weak streams. We formally prove that using our algorithm ensures that *all packets reach their destination* for wide families of networks, thus addressing the starvation and last-packet issues.

In addition, we propose a packet reordering scheme that makes use of explicit delay information, relaxes the out-of-order arrivals of packets to their destination, reduces delays, and incurs an expected computational overhead that is logarithmic in the size of each queue. We provide intuition and formal arguments for how the system should be tuned to reach the best performance using our schemes.

Using extensive simulations, we show that we can indeed overcome the *starvation* and the *last-packet* problems of an infrequent stream as well as enhance weak streams while reducing the mean delay of all the other streams. We further show a significant delay reduction using our schemes in multiple randomly-generated static and dynamic networks based on the scale-free Barabasi-Albert model [1].

The rest of the paper is organized as follows. Section II describes the system model. Section III presents the starvation-free stability and its basic characteristics. Section IV presents our starvation-free algorithm. Section V presents the packet reordering scheme. Section VI presents extensive simulations that verify our results. Finally, conclusions are presented in Section VII.

## II. System model

In this section, we present our system model, which largely follows similar lines to the standard backpressure system model [7], [18].

### A. Network model

We consider a multi-hop dynamic network that operates in slotted time $t \in \{0, 1, 2, ...\}$. The network is classically represented by a directed and weighted time-dependent graph $G(t) = (V, E, C(S(t), I(t)))$, where $V$ is the set of transmitter/receiver nodes, and $E$ is the set of links between the nodes that can directly communicate with each other. $S(t)$ is the network's topology state (as defined in [7], Ch. 1). For simplicity, we assume that $S(t)$ is i.i.d. between time slots and takes values within a finite state space $\mathbf{S}$. $I(t)$ represents a link control action taken by the network during time slot $t$ (considering constraints such as activation sets, power limitations and more, depending on the topology state). $C(S(t), I(t)) = (C_{ab}(S(t), I(t)))_{ab}$ is the network links' capacity vector at time slot $t$, where $C_{ab}(S(t), I(t))$ is the capacity of link $(a, b)$ at time slot $t$. It is expressed in number of packets, and can take values in $\{0, 1, ..., C_{ab,\max}\}$, where $C_{ab,\max}$ is a finite constant.

Each node can be both a source and a destination of data. A *commodity* is defined by its destination and is denoted by $c \in V$. Packets that reach their destination are assumed to immediately leave the network. Thus, each node needs a maximum number of $|V| - 1$ (unbounded) queues for storing packets. $U_i^{(c)}(t)$ is the number of packets that belong to commodity $c$ and reside in node $i$ at time slot $t$. Let us denote by $\mu_{ab}^{(c)}(t)$ the transmission rate (in number of packets) offered to commodity $c$ in node $a$ at time slot $t$ over link $(a, b)$. Therefore for all links $(a, b) \in E$ and all commodities $c \in V$,

$$\mu_{ab}^{(c)}(t) \le \mu_{ab,\max}^{(c)} \le C_{ab,\max} \, \forall t \tag{1}$$

since the offered transmission rate cannot exceed the link capacity, where $\mu_{ab,max}$ is a tighter bound on the service rate that may exist due to additional restrictions on the network $(I(t))$.

### B. Exogenous arrival process and admissibility

Let us denote by $A_i^{(c)}(t)$ the number of packets of commodity $c$ that enter the network exogenously at node $i$ at time slot $t$. We assume for simplicity that, for all nodes $i \in V$, all commodities $c \in V \setminus \{i\}$ and all time slots $t$,

$$A_i^{(c)}(t) \in \left\{0, 1, ..., A_{i,\max}^{(c)}\right\} \forall t, \tag{2}$$

$$A_i^{(c)}(t) \text{ is i.i.d. over time slots,} \tag{3}$$

$$\mathbb{E}\left(A_i^{(c)}(t)\right) = \lambda_i^{(c)}, \tag{4}$$

where $A_{i,\max}^{(c)}$ is a finite constant and $\lambda = \left(\lambda_i^{(c)}\right)$ is the exogenous arrival rate vector to the network.

Let $\Lambda$ be the network capacity region [7], [18]. We say that an arrival rate vector $\lambda$ is *admissible* if $\lambda \in \Lambda$, and it is *strictly admissible* if there exists a vector $\epsilon > 0$ such that $\lambda + \epsilon \in \Lambda$.

## III. starvation-free stability

In this section we define a stronger notion of stability, namely, *starvation-free stability*. The motivation for this definition is the observation that, while various queue-length-based stability criteria [19], [20] guarantee that the mean length of the queues in the network is bounded, it is still possible under such stability definitions that some packets may reside at their queues for unbounded periods of time. Therefore, a queue-length-based stability does not imply that packets ever leave their queues since *a packet may stay stuck in a perfectly bounded queue*. To address this problem, we present the following stronger *starvation-free stability* criterion that not only ensures that the mean queue lengths are bounded, but also guarantees a *repeated* evacuation of all queues, even if their arrival process is *finite*.

### A. Starvation-free stability of a queue

Let $D(t)$ denote the delay (in number of time slots) of the HoL (head-of-line) packet of a FIFO queue.[1] We introduce the following definition:

**Definition 1.** *We define a queue as* starvation-free stable *if*

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}(D(\tau)) < \infty. \tag{5}$$

[1]The results in this section would equally apply to the delay of the oldest packet of a non-FIFO queue.

With this definition at hand, we proceed by obtaining the following key result.

**Theorem 1.** *Let $p$ be a packet that enters a starvation-free queue. Then $p$ will leave the queue w.p. (with probability) 1.*

*Proof:* We observe that, while the delay of the HoL packet of a FIFO queue may experience large drops, it can only grow by at most 1 from one time slot to another. Next, let us define two auxiliary stochastic precesses $a(t)$ and $b(t)$ as follows:

$$a\left(t\right)=\begin{cases}1 & \text{if} \quad D\left(t+1\right)=D\left(t\right)+1\\ 0 & \text{otherwise,}\end{cases} \tag{6}$$

and

$$b\left(t\right)=\begin{cases}0 & \text{if} \quad D\left(t+1\right)=D\left(t\right)+1\\ D\left(t\right)-D\left(t+1\right) & \text{otherwise.}\end{cases} \tag{7}$$

Then we can recursively express the changes in the HoL delay $D(t)$ of this queue using $a(t)$ and $b(t)$:

$$D\left(t+1\right)=\max\left\{D\left(t\right)-b\left(t\right),0\right\}+a\left(t\right). \tag{8}$$

We observe that for all $t$, $0 \leq a\left(t\right)$ and $b^{-}\left(t\right) \triangleq -\min\left\{b\left(t\right),0\right\} \equiv 0$. Therefore, $a\left(t\right)+b^{-}\left(t\right) \leq 1$, satisfying the boundedness condition that is required by Theorem 4 of [20] (case (b)) for a stochastic process with *queue dynamics*. Specifically, the proof of Theorem 4 of [20] applies to any arbitrary stochastic processes $a(t)$ and $b(t)$ that obey the boundedness condition. Namely, the result holds even though $a(t)$ and $b(t)$ depend non-causally on $D(t+1)$. Thus, applying Theorem 4 of [20] on (5) yields:

$$\lim_{t\to\infty}\frac{D\left(t\right)}{t}=0 \quad w.p.1. \tag{9}$$

Now, assume by way of contradiction that packet $p$ never leaves the queue. Then its delay would linearly grow to infinity. Since the delay of packet $p$ is upper-bounded by the HoL delay of the queue, we get a contradiction to (9), which concludes the proof. ■

Note that the proof of Theorem 1 requires only the boundedness condition of [20] (Theorem 4, case (b)). Thus, this result is not restricted to our system model, but holds whenever starvation-free stability and the boundedness condition are met.

### *B. Comparison with strong stability*

We begin by presenting the well-known queue-length-based *strong stability* [19], [20].

**Definition 2.** *Let $U(t)$ denote the number of packets in a queue. Then, the queue is* strongly stable *if*

$$\lim_{t\to\infty}\sup\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\left(U\left(\tau\right)\right)<\infty. \tag{10}$$

We proceed by showing that, whenever the arrival rate to the queue has a finite support, *starvation-free stability* is a strictly stronger stability criterion than *strong stability*.

**Property 1.** (i) *If a queue is starvation-free stable, then it is also strongly stable.*

(ii) *On the contrary, if a queue is strongly stable, then it may not be starvation-free stable.*

*Proof: (i)* The number of packets in the queue is bounded by the number of packets that have entered since the HoL last entered, i.e. by the product of the HoL delay by the number of packets that can enter at each slot. Let $c$ denote the upper bound on the number of packets that can enter the queue during a single time slot. Then, we obtain:

$$U\left(t\right)\leq c\cdot D\left(t\right) \quad \forall t \tag{11}$$

This inequality *deterministically* holds at all time-slots. In addition, by Definition 1, in a starvation-free stable queue, the expected time-averaged delay of the HoL packet is bounded. Therefore, the expected time-averaged queue size is also bounded, yielding the result.
*(ii)* Consider a queue with a single arrival at time 0, and no additional arrivals or departures. Such a queue is clearly strongly stable. However, since the HoL packet is stuck in the queue and never leaves it, then in this simple *deterministic* process, according to Theorem 1, this queue cannot be starvation-free stable. ■

Note that the proof of Property 1 requires only a bound on the number of packets that can enter the queue during a single time slot. Thus, this result is not restricted by our system model, but holds whenever the arrival rate to the queue has a finite support.

### *C. Starvation-free stability of a network*

We now naturally proceed to generalize the starvation-free stability of a queue, and define the starvation-free stability of a network of queues.

**Definition 3.** *We define a network as* starvation-free stable *if all its queues are starvation-free stable.*

In order to point out the guarantees that this definition provide, we start our analysis by establishing the following theorem.

**Theorem 2.** *Let $p$ denote an arbitrary packet that enters a starvation-free stable network. Then w.p. 1, packet $p$ will either reach its destination or it will be transmitted for an unbounded number of times.*

*Proof:* Let us assume by contradiction that packet $p$ never leaves the network and is transmitted for some finite number of times $M$. Then, after packet $p$ is transmitted for the $M$'th time, its delay will linearly grow to infinity. Since the delay of packet $p$ is upper-bounded by the HoL delay of its residence queue, we get a contradiction to (9), which concludes the proof. ■

We proceed by making the following observation. In a network in which loop-free routing can be assumed, every packet can be transmitted for only a bounded number of times, thus packet delays in such a network are bounded by the (finite) sum of the corresponding HoL delays. Using this observation, we obtain the following result:

**Corollary 1.** *In a starvation-free stable network with loop-free routing, all packets reach their destination w.p. 1.*

Let us emphasize that this result is *fundamentally different and stronger* than results that are obtained using typical backpressure schemes. The reason is that the latter rely on Little's Law, which can guarantee a finite *expected* mean delay for the network packets, while potentially allowing some packets to reside in the network forever. Therefore, these schemes can suffer from starvation. On the contrary, *our result applies to each and every packet in the network*.

To illustrate this fact, consider the example given in Figure 2. In this configuration, when using Q-BP, some data packets will reside in the network *forever*, even though the mean delay is bounded by Little's Law. However, when using a starvation-free stable scheme, such starvation cannot occur.

Note that the loop-free routing assumption covers a wide family of topologies including general 1-hop networks, DAGs (Directed Acyclic Graphs), networks with fixed routes, as well as all cases where one can ensure that any routing loop would be *eventually* broken.

## IV. STARVATION-FREE BACKPRESSURE ALGORITHM

We proceed to introduce our starvation-free backpressure algorithm. We begin by constructing our potential functions, such that the potential function of each queue upper-bounds the sum of its queue size and the product of its HoL delay by a constant factor.

### A. Constructing the potential functions

We begin our analysis by assuming that each packet holds a timestamp that represents the time-slot number at which it entered its current queue. Let us denote by $D_{i,k}^{(c)}(t)$ the number of time slots that the $k$th packet in the queue of commodity $c$ at node $i$ has stayed in this queue by time $t$; namely, it entered that queue at time slot $t - D_{i,k}^{(c)}(t)$. In particular, we denote $D_i^{(c)}(t) \equiv D_{i,1}^{(c)}(t)$ the HoL delay. Since $D_{i,k}^{(c)}(t)$ can be easily calculated by subtracting the packet's timestamp from the current time-slot number, it does not incur any constant update of dedicated fields with a heavy computational overhead.

We continue by introducing a novel approach of tracking the HoL delay changes of queues, without any additional assumption on either the network's topology state or the exogenous arrival processes, while incurring no calculation overhead. We would like the potential of each queue $Z_i^{(c)}(t)$ to upper-bound the sum of the queue size and the product of the HoL delay by a constant factor, i.e.

$$Z_i^{(c)}(t) \geq U_i^{(c)}(t) + \delta_i^{(c)} \cdot D_i^{(c)}(t). \qquad (12)$$

Intuitively, $\delta_i^{(c)}$ is a non-negative constant that represents the importance of the delay component $D_i^{(c)}(t)$ versus the queue-length component $U_i^{(c)}(t)$. It should be tuned according to the stream characteristics, as we later explain. The stability of this linear combination of queue length and HoL delay will provide the needed delay guarantees. Let us further denote by $\delta = \left(\delta_i^{(c)}\right)$ the vector of commodity delay coefficients $\delta_i^{(c)}$.

In order to construct the potential function $Z_i^{(c)}(t)$, we first introduce an equation that tracks the HoL delay changes in a queue, so that:

$$D_i^{(c)}(t+1) = D_i^{(c)}(t) + d_{i,in}^{(c)}(t) - d_{i,out}^{(c)}(t). \qquad (13)$$

**Incoming delay.** Intuitively, we want $d_{i,in}^{(c)}(t)$ to represent the delay entering the queue and increasing its HoL delay. To that end, we assign to $d_{i,in}^{(c)}(t)$ the value of 1 iff the queue remains non-empty at the following time slot. Since the value of $d_{i,in}^{(c)}(t)$ should be determined at time slot $t$, we observe that the queue is not empty at the following time slot iff at least one of the following two conditions holds: (a) the total service rate given to this queue at time slot $t$ is smaller than its occupancy, or (b) there is exogenous or endogenous arrival of packets to this queue at time slot $t$. Thus, we get:

$$d_{i,in}^{(c)}(t) = \begin{cases} 1 & \left(\sum\limits_b \mu_{ib}^{(c)}(t) < U_i^{(c)}(t)\right) \text{ or} \\ & \left(\sum\limits_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) > 0\right) \\ 0 & \text{otherwise} \end{cases}$$

**Outgoing delay.** Similarly, we would like $d_{i,out}^{(c)}(t)$ to represent the delay leaving the queue. To that end we assign to $d_{i,out}^{(c)}(t)$ the difference between the HoL delay at time $t$ and the HoL delay after packets depart the queue at time $t$. For instance, assume that the HoL packet has spent 17 time slots in the queue while the 3rd packet has only spent 4 slots, and that the first two packets depart the queue. Then the delay leaving the queue is the difference in delay between the 1st and 3rd packets, i.e. $17-4=13$. We further make the additional observation that if all the packets leave the queue at time slot $t$, then the leaving delay is precisely the delay of the HoL packet. Note that whenever the queue is empty or there is no service rate, the leaving delay is zero, so whenever $U_i^{(c)}(t) = 0$ or $\sum\limits_b \mu_{ib}^{(c)}(t) = 0$, we define $d_{i,out}^{(c)}(t) = 0$. We obtain:

$$d_{i,out}^{(c)}(t) = \begin{cases} D_i^{(c)}(t) & 0 < U_i^{(c)}(t) \leq \sum\limits_b \mu_{ib}^{(c)}(t) \\ 0 & \left(U_i^{(c)}(t) = 0\right) \text{ or } \left(\sum\limits_b \mu_{ib}^{(c)}(t) = 0\right) \\ D_i^{(c)}(t) - D_{i,1+\sum_b \mu_{ib}^{(c)}(t)}^{(c)}(t) & \text{otherwise} \end{cases}$$

**Bound on outgoing delay.** In addition, since our goal is to prevent starvation and enhance weaker commodities, we introduce an additional novel mechanism to that end. We notice that weaker commodities have fewer packets that can stay to help reinforce their potential. When their packets leave, the network essentially forgets that it should give some priority to these commodities. As a result, we want to bound $d_{i,out}^{(c)}(t)$ such that leaving packets will not immediately reset their accumulated delay upon their departure. Therefore, we further define a vector $d_{\max} = \left(d_{i,\max}^{(c)}\right)$ such that

$$\forall i, c, \quad d_{i,\max}^{(c)} \in \mathbb{R}^+ \cup \infty. \qquad (14)$$

We then construct a variable that represents the regulated outgoing delay:

$$\tilde{d}_{i,out}^{(c)}(t) = \min\left(d_{i,out}^{(c)}(t), d_{i,\max}^{(c)}\right). \quad (15)$$

$d_{i,\max}^{(c)}$ can be interpreted as the *inverse to the memory* of the network, as we shall further discuss in Section IV-D. This parameter can be used to enhance weaker commodities by *remembering* the delays that were experienced by earlier packets even after their departure. For instance, assume $d_{i,\max}^{(c)} = 100$. If the unique packet of commodity $c$ in node $i$ experiences a large delay of 1000, then upon its departure, we do not reset the potential of the commodity in the node to 0, but set it to reflect a delay of about $1000 - 100 = 900$. The goal is for this node to remember that this commodity should get some priority over stronger commodities.

Note that as a result, whenever $d_{i,\max}^{(c)} < \infty$, we will see that the corresponding potential function $Z_i^{(c)}(t)$ may be strictly positive even when the queue is empty. This may lead to a temporary waste of bandwidth, since a link may be allocated for this queue solely for the purpose of decrementing the previously-accumulated delay potential. This effect is similar to a *virtual queue* [3] that might reflect strictly positive potential even if there are no packets to send. The main difference is that our potential function does not grow if there are no packets in the queue, whereas the potential of a *virtual queue* may be constantly growing.

On the other hand, whenever $d_{i,\max}^{(c)} = \infty$, there may be an *unbounded* change in the value of the potential function between time slots, due to a possible arbitrary large drop in the delay of the HoL packet. In such a case, the inequality (12) becomes an equality.

**Potential functions.** Finally, after all these preparations, we get to formally define recursively the potential functions used by our algorithm by incorporating the above HoL delay considerations.

**Definition 4.** *We recursively define the potential function $Z_i^{(c)}(t)$ of our algorithm using:*

$$\begin{aligned} Z_i^{(c)}(t+1) = \; & \max\left(Z_i^{(c)}(t) - \sum_b \mu_{ib}^{(c)}(t) - \delta_i^{(c)} \cdot \tilde{d}_{i,out}^{(c)}(t), 0\right) \\ & + \sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) + \delta_i^{(c)} \cdot d_{i,in}^{(c)}(t) \end{aligned}$$

*Additionally, we initialize the potential as*

$$Z_i^{(c)}(0) = U_i^{(c)}(0) + \delta_i^{(c)} \cdot D_i^{(c)}(0) \quad (16)$$

*under the assumption that*

$$Z_i^{(c)}(0) < \infty. \quad (17)$$

*The potential at the commodity destination is always zero, i.e. $Z_i^{(i)}(t) = 0 \; \forall i, t$.*

With the potential functions at hand, we are ready now to introduce our starvation-free algorithm.

## B. The SF-BP algorithm

We now follow the standard backpressure framework to define our SF-BP (Starvation-Free BackPressure) algorithm. We define a *weight* for each link in the network as follows:

$$W_{ab}(t) = \max\left[Z_a^{\left(c_{ab}^{opt}(t)\right)}(t) - Z_b^{\left(c_{ab}^{opt}(t)\right)}(t), 0\right] \quad (18)$$

where

$$c_{ab}^{opt}(t) \triangleq \arg\max_c \left[Z_a^{(c)}(t) - Z_b^{(c)}(t)\right] \quad (19)$$

Given these weights and the current topology state $S(t)$, we can now formulate the following optimization problem:

$$\begin{array}{ll} \max & \sum_{ab} W_{ab}(t) C_{ab}(I(t), S(t)) \\ \text{s.t.}: & I(t) \in I_s(t) \end{array} \quad (20)$$

The solution of this optimization problem dictates the algorithm's choice of the resource allocation between the different links, where $I_s(t)$ is the set of all possible control actions available under the current topology state $S(t)$. Note that a network with independent links allows a fully distributed implementation of the solution algorithm.

For each $W_{ab}(t) > 0$, the network offers a transmission rate as follows (see [7], [18]):

$$\mu_{ab}^{(c)}(t) = \begin{cases} C_{ab}(I(t), S(t)) & c = c_{ab}^{opt}(t) \\ 0 & \text{otherwise.} \end{cases} \quad (21)$$

Whenever more than a single outgoing link is offered to a commodity at some node, *the allocation of data packets among the outgoing links is random.* As done in the classical Q-BP schemes [7], if there are not enough packets for commodity $c_{ab}^{opt}(t)$ in node $a$ to transmit over all outgoing links, *null* packets will be transmitted, with a random allocation of data packets and null packets over the corresponding outgoing links.

## C. Obtaining starvation-free stability

We proceed to show that our algorithm is starvation-free stable. We begin our analysis by using the Lyapunov drift technique to prove the expected boundedness of our potential functions.

**Lemma 1.** *Let $\lambda$ denote a strictly admissible arrival rate vector. Assume that the algorithm uses a vector $\delta$ such that $\lambda + \delta$ is also strictly admissible, i.e, there exists $\epsilon > 0$ such that $\lambda + \delta + \epsilon \in \Lambda$. Then*

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{ic} \mathbb{E}\left(Z_i^{(c)}(\tau)\right) \le \frac{B}{2\varepsilon_{\min}}.$$

*where*

$$\begin{aligned} B = & \sum_{ic}\left(\sum_b \mu_{ib,\max}^{(c)}\right)^2 \\ & + \sum_{ic}\left(\sum_a \mu_{ai,\max}^{(c)} + A_{i,\max}^{(c)} + \delta_i^{(c)}\right)^2 \end{aligned} \quad (22)$$

*and $\varepsilon_{\min} = \min(\varepsilon)$.*

*Proof:* We start with the observation that $\tilde{d}_{i,out}^{(c)}(t) \geq 0$ and $d_{i,in}^{(c)}(t) \leq 1$ for all $t$ and $(i,c)$. We apply it on Definition 4 (i.e., the potential dynamics) and immediately obtain:

$$\begin{aligned} Z_i^{(c)}(t+1) \leq \quad & \max\left(Z_i^{(c)}(t) - \sum_b \mu_{ib}^{(c)}(t), 0\right) \\ & + \sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) + \delta_i^{(c)} \end{aligned} \tag{23}$$

Now, Equation (23) yields an upper bound at each time slot on the potential functions used by our algorithm. For brevity, we omit the rest of the proof, as Equation (23) is determined by the standard Lyapunov technique presented in [7]. ■

We proceed with our analysis by showing that the potential of each queue upper-bounds a combination of its length and its HoL delay. This result provides the desired property of the potential functions that allows us to establish starvation-free stability.

**Lemma 2.** $\forall t, i \neq c, \quad Z_i^{(c)}(t) \geq U_i^{(c)}(t) + \delta_i^{(c)} D_i^{(c)}(t)$

*Proof outline:* Due to space limits, we present a proof outline, as the proof is relatively simple. The proof is by induction. The base is immediately derived from (16). Next we prove the inductive step by splitting into 3 different cases. The first case considers the scenario in which the result of the *max(,)* operator in the definition of the potential function is 0. The other two cases consider the scenarios in which the result of the *max(,)* operator is strictly greater than 0, with outgoing service rate that is either greater than, or smaller or equal to the queue occupancy. ■

With Lemmas 1 and 2 at hand, we can finally establish the starvation-free stability of our algorithm.

**Theorem 3.** *Assume that $\lambda + \delta$ is strictly admissible. Then*

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{ic} \mathbb{E}\left(D_i^{(c)}(\tau)\right) \leq \frac{B}{2\delta_{\min}\varepsilon_{\min}}$$

*where*

$$\begin{aligned} B = \quad & \sum_{ic}\left(\sum_b \mu_{ib,\max}^{(c)}\right)^2 \\ & + \sum_{ic}\left(\sum_a \mu_{ai,\max}^{(c)} + A_{i,\max}^{(c)} + \delta_i^{(c)}\right)^2, \end{aligned} \tag{24}$$

$\varepsilon_{\min} = \min(\varepsilon)$ *and* $\delta_{\min} = \min(\delta)$. *Thus the algorithm is starvation-free stable within the entire capacity region.*

*Proof:* Applying Lemma 2 to Lemma 1 yields:

$$\begin{aligned} & \delta_{\min} \cdot \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{ic} \mathbb{E}\left(D_i^{(c)}(\tau)\right) \leq \\ & \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{ic} \mathbb{E}\left(U_i^{(c)}(\tau) + \delta_i^{(c)} \cdot D_i^{(c)}(\tau)\right) \leq \\ & \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{ic} \mathbb{E}\left(Z_i^{(c)}(\tau)\right) \leq \frac{B}{2\varepsilon_{\min}} \end{aligned}$$

dividing both sides by $\delta_{min}$ to yield the result. ■

Note that, throughout the proof, we did not make any assumptions regarding the values of $d_{max}$. Thus, *starvation-free* stability of our algorithm is ensured for all $d_{i,\max}^{(c)} \in [0, \infty]$ Next, we obtain the following corollaries:

**Corollary 2.** *Assume that $\lambda + \delta$ is strictly admissible. Then* the network is strongly stable.

The correctness of this corollary follows immediately both from Property 1 and by applying Lemma 2 to Lemma 1.

**Corollary 3.** *Assume that $\lambda+\delta$ is strictly admissible. Then w.p. 1, all packets in the network either reach their destination or are transmitted for an unbounded number of times. In particular,* in loop-free scenarios, all packets reach their destination w.p. 1.

The correctness of this corollary follows immediately from Theorem 2, Theorem 3 and Corollary 1.

We proceed to analyze the influence of the vector parameters $d_{\max}$ and $\delta$ on system performance and provide further per-queue service guarantees.

## *D. The significance of $d_{\max}$ and $\delta$*

*Intuitively, $d_{i,\max}^{(c)}$ can be viewed as inversely proportional to the memory of the potential function.* Specifically, when $d_{i,\max}^{(c)} = 0$, the memory is infinite. The potential function grows in every time slot where the queue is not empty, and it carries congestion information from the beginning. Conversely, when $d_{i,\max}^{(c)} \to \infty$, there is negligible memory, and the outgoing packets reset the potential on their departure from the congestion they experienced. More generally, assigning $d_{i,\max}^{(c)} = k$ means that the potential function will carry the information about all the packets that experienced delay greater that $k$ while residing in that queue.

In addition, $\delta_i^{(c)}$ *can be viewed as the intensity in which the delay of the HoL packet affects the potential function.*

Therefore, jointly adjusting $\delta$ and $d_{\max}$ can either enhance weak streams or provide better service to large streams. In addition, they can be adjusted to adapt to the topological changes. For example, in static and $i.i.d.$ networks, the assignment of small $d_{\max}$ values will be the better choice as we would like the potential to have more memory. Conversely, in a highly dynamic network with unpredictable topology state evolution, we will prefer a memoryless system and therefore larger values of $d_{\max}$, because the reordering of nodes can create potential barriers that will take longer to reset.

Note that the $\delta$ values must respect the restriction that $\lambda+\delta$ is strictly admissible. Therefore, to set arbitrary $\delta$ values that respect this restriction, one must know the capacity region and the arrival rates. This limitation also holds for similar past techniques, such as the *shadow queues* technique [3]. However, when the capacity region and the arrival rates are unknown, it is still possible to obtain the results in the paper and the guarantee of starvation-free stability, as the $\delta$ values can be set to arbitrarily small values.

We turn to providing further insight on the influence of $d_{\max}$ and $\delta$ on system performance, and evaluating their mutual effect on queue lengths and HoL delay dynamics. We start our analysis by proving the following lemmas.

**Lemma 3.** *Define:*

$$P_i^{(c)}(t) \triangleq \delta_i^{(c)} \max\left\{d_{i,out}^{(c)}(t) - d_{i,\max}^{(c)}, 0\right\} - \max\left\{\sum_b \mu_{ib}^{(c)}(t) - U_i^{(c)}(t), 0\right\}. \tag{25}$$

*Assume that $\lambda + \delta$ is strictly admissible. Then, w.p. 1:*

$$\limsup_{t\to\infty} \frac{1}{t}\left[\sum_{\tau=0}^{t-1} P_i^{(c)}(\tau)\right] \le 0.$$

*Proof:* See Appendix A. ■

**Lemma 4.** *Assume that $\lambda+\delta$ is strictly admissible. Then, w.p. 1, for each queue $(i,c)$ such that $\delta_i^{(c)} > 0$:*

$$\lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\left[d_{i,out}^{(c)}(\tau) - d_{i,in}^{(c)}(\tau)\right] = 0.$$

*Proof:* See Appendix B. ■

Lemma 3 is derived from the stability of the network potentials (Lemma 1). It relates the network parameters $\delta$ and $d_{max}$ to the queue length and HoL delay dynamics. In addition, Lemma 4 is derived from the rate-stability of the HoL packet delay of the network queues (*i.e.,* (9)). Note that these results hold for each queue *independently*. Next, we use these results in order to obtain stronger guarantees for weak and preferred streams by setting appropriate $d_{max}$ and $\delta$ values, as we further show in Section IV-E.

## *E. Mean evacuation period guarantees*

So far, we have obtained a starvation-free stability for the entire network. Now, in order to evaluate the service guarantees provided by the $d_{max}$ and $\delta$ values for each queue in the network *independently*, we proceed to define the following indicator function:

$$\mathbb{1}_{i,Q}^{(c)}(t) = \begin{cases} 1 & \sum_b \mu_{ib}^{(c)}(t) \ge U_i^{(c)}(t) \\ 0 & \text{otherwise,} \end{cases} \tag{26}$$

Essentially, $\mathbb{1}_{i,Q}^{(c)}(t) = 1$ means that *all* packets of commodity $c$ that arrived at node $i$ before time slot $t$ must have left their queue by time slot $t$. We term the time period between two such events the *evacuation period* of the queue. We next turn to providing a bound on the mean evacuation period that can be experienced by any queue in the network. Specifically, we obtain the following result.

**Theorem 4.** *Assume that $\lambda + \delta$ is strictly admissible. Then, w.p. 1, for each queue $(i,c)$ such that $\delta_i^{(c)} > 0$:*

$$\liminf_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{1}_{i,Q}^{(c)}(\tau) \ge \frac{\delta_i^{(c)}\left(1 - d_{i,\max}^{(c)}\right)}{\delta_i^{(c)} + \mu_{i,O}^{(c)}}$$

*where $\mu_{i,O}^{(c)} = \sum_b \mu_{ib,\max}^{(c)}$.*

*Proof:* We begin by rewriting (25), yielding:

$$P_i^{(c)}(t) \ge \delta_i^{(c)}(d_{i,out}^{(c)}(t) - d_{i,max}^{(c)}) - \mathbb{1}_{i,Q}^{(c)}(t)\cdot \mu_{i,O}^{(c)}. \tag{27}$$

Next, we add and subtract $\delta_i^{(c)} d_{i,in}^{(c)}(\tau)$ to the right hand side of (27) and rearrange. This yields:

$$P_i^{(c)}(t) \ge \delta_i^{(c)}(d_{i,out}^{(c)}(t) - d_{i,in}^{(c)}(t)) - \delta_i^{(c)} d_{i,max}^{(c)} - \mathbb{1}_{i,Q}^{(c)}(t)\cdot \mu_{i,O}^{(c)} + \delta_i^{(c)} d_{i,in}^{(c)}(t). \tag{28}$$

We observe that the following inequality holds by definition:

$$\mathbb{1}_{i,Q}^{(c)}(t+1) \ge 1 - d_{i,in}^{(c)}(t). \tag{29}$$

Applying (29) on (28), adding and subtracting $\delta_i^{(c)}\cdot \mathbb{1}_{i,Q}^{(c)}(t)$ and rearranging yields:

$$\begin{aligned} P_i^{(c)}(t) \ge\ & \delta_i^{(c)}(1 - d_{i,max}^{(c)}) - \mathbb{1}_{i,Q}^{(c)}(t)\cdot(\mu_{i,O}^{(c)} + \delta_i^{(c)}) \\ & + \delta_i^{(c)}(d_{i,out}^{(c)}(\tau) - d_{i,in}^{(c)}(t)) \\ & - \delta_i^{(c)}\left(\mathbb{1}_{i,Q}^{(c)}(t+1) - \mathbb{1}_{i,Q}^{(c)}(t)\right) \end{aligned} \tag{30}$$

Rearranging (30), summing over time slots $[0,1,...,t-1]$ and dividing by $t$ yields:

$$\begin{aligned} & - \delta_i^{(c)}(1 - d_{i,max}^{(c)}) + \frac{1}{t}\sum_{\tau=0}^{t-1} P_i^{(c)}(\tau) \ge \\ & \delta_i^{(c)}\frac{1}{t}\sum_{\tau=0}^{t-1}\left(d_{i,out}^{(c)}(\tau) - d_{i,in}^{(c)}(\tau)\right) \\ & - \frac{\delta_i^{(c)}}{t}\left(\mathbb{1}_{i,Q}^{(c)}(t) - \mathbb{1}_{i,Q}^{(c)}(0)\right) \\ & - \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{1}_{i,Q}^{(c)}(\tau)\cdot(\mu_{i,O}^{(c)} + \delta_i^{(c)}) \end{aligned} \tag{31}$$

Taking limits of (31) and applying Lemmas 3 and 4 yields:

$$\begin{aligned} & - \delta_i^{(c)}(1 - d_{i,max}^{(c)}) \ge \\ & (\mu_{i,O}^{(c)} + \delta_i^{(c)})\cdot \limsup_{t\to\infty} -\frac{1}{t}\left[\sum_{\tau=0}^{t-1}\left(\mathbb{1}_{i,Q}^{(c)}(\tau)\right)\right]. \end{aligned} \tag{32}$$

Rearranging (32) and dividing by $\mu_{i,O}^{(c)} + \delta_i^{(c)}$ yields the result. ■

Now, as discussed in Section IV-D, for static and $i.i.d$ networks, the desired case is $d_{i,max}^{(c)} = 0$. For this, we are able to provide with a stronger guarantee.

**Corollary 4.** *Assume that $\lambda + \delta$ is strictly admissible. Then for each queue $(i,c)$ such that $\delta_i^{(c)} > 0$ and $d_{i,max}^{(c)} = 0$, we have w.p. 1:*

$$\liminf_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{1}_{i,Q}^{(c)}(\tau) \ge \frac{\delta_i^{(c)}}{\delta_i^{(c)} + \mu_{i,O}^{(c)}}.$$

We already found in Lemma 1 that the expected time-averaged sum of the potential functions is upper-bounded by a term that grows quadratically with respect to the incoming/outgoing link capacities, and is inversely proportional to $\varepsilon$. This result is the standard upper-bound obtained by the Lyapunov analysis, and also applies to Q-BP [7].

However, *Corollary 4 states three significantly stronger results*. First, the mean evacuation period of each queue in the network is constant. Namely, for each queue in the network, in addition to starvation-free stability, we obtain that the number of time slots in which all the packets at the queue must have left it before that time slot is proportional to time itself.

Second, our bound is expressed in terms of *pure* values, whereas the standard Lyapunov technique yields a bound that is expressed in terms of *expected* values.

Finally, since the result in Corollary 4 is *independent of the arrival rate vector* $\lambda$,[2] this result can be used in order to further enhance weak commodities and to ensure *fairer* resource sharing between commodities with significant differences in their arrival rates as illustrated above. As an example, consider two queues that are served by the same link. Then, *setting their corresponding $\delta$ entries unambiguously determines the upper bound on the mean evacuation period and, in turn, the starvation that they experience without any knowledge about their arrival rates.*

## V. Packet reordering scheme

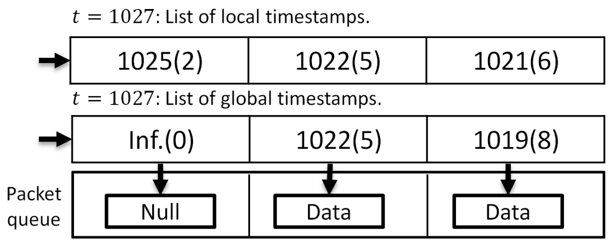
In order to reduce the maximum residence time of packets in the network and to relax the out-of-order arrivals to the destinations, *we introduce a reordering technique for packets in queues*. This technique incurs reasonable computational overhead and has no effect on the capacity region.

To that end, in addition to the assumption that each packet has a *local timestamp* that indicates at which time slot it entered its current queue, we assume that it also has a *global timestamp* that indicates at which time slot it entered the network at its source node.

*The global time of each entering packet determines its place in the queue*, such that older (global) packets are at the head of the queue and the first to leave it. Note that the correct place in the queue for a real packet that enters node $i$ at time $t$ and belongs to commodity $c$ can be found in logarithmic time, i.e. $O\left(\log\left(U_i^{(c)}\left(t\right)\right)\right)$ operations using a priority queue data structure.

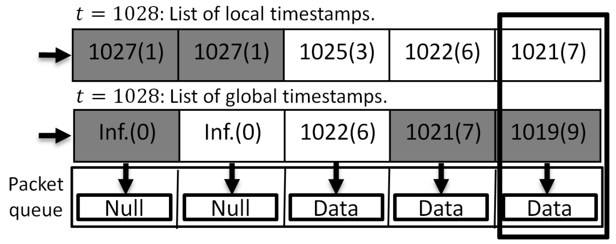
The global timestamp of *null* packets is ignored and treated as infinite, meaning that null packets are always kept as last priority for forwarding. However, we note that if an entering packet (with a local time of 0) can immediately move to be the HoL packet, this would mean that the local time of the HoL can arbitrarily jump in unexpected ways. This would be inconsistent with the purpose of the Lyapunov potentials, since the HoL delay would be negligible while there may be packets in the queue that have been waiting for a long time. Therefore, as illustrated in Figure 3, we need to redefine the local times. We keep the local timestamps in a separate queue, such that the $k^{\text{th}}$ local time is associated with the $k^{\text{th}}$ global time. The

[2]The arrival rates do restrict the possible choices for $\delta$ values as $\lambda+\delta$ must be strictly admissible. Within the allowed values, in Corollary 4, the bound on the mean evacuation period of each queue only depends on $\delta$.



$t$ = 1027: List of local timestamps.
1025(2)
1022(5)
1021(6)
$t$ = 1027: List of global timestamps.
Inf.(0)
1022(5)
1019(8)
Packet queue
Null
Data
Data

(a) Time slot 1027. There are three packets in the queue. As a result, the list of local timestamps contains three local timestamps (e.g., the first one is 1021, which arrived $1027-1021=6$ slots ago). Likewise, the list of global timestamps also contains three timestamps (e.g., the first one is 1019, of a packet that entered the network 8 slots ago).



(b) Time slot 1028. Two packets enter the queue: a data packet with a global time stamp of 1021, and a *null* packet. Their local and global timestamps (filled in grey) enter the corresponding lists, each one according to its order. Thus, note that the fourth local time is not necessarily the time that the packet with the fourth global time stayed in the queue. Finally, at time slot 1028, a single packet leaves the queue. It is the one that is associated with the smallest global time stamp, and it removes from the local-time list the smallest local time.

Fig. 3: Illustration of the reordering technique. The upper queue holds the local time stamps and the lower queue holds the global time stamps.

reordering of packets does not harm the delay guarantees, due to the simple observation that the global residence time of a packet is necessarily upper-bounded by the sum of the corresponding HoL delays of its *previous* residence nodes.

We expect that the reordering scheme will provide an additional significant performance enhancement especially when the network in not heavily loaded. The intuition for this improvement is the observation that the number of null packets in the network increases as the load decreases. Thus, prioritizing data packets over null packets can enhance performance significantly by using these null packets as a *basis* for the potential functions. Essentially, the delays of data packets will drop at the expense of null packets that will suffer from extremely large delays.

## VI. Simulations

In this section, we compare between the usual queue-based backpressure algorithm Q-BP and our starvation-free backpressure algorithm SF-BP algorithm, as well as between Q-BP with our reordering scheme (denoted QR-BP) and our SF-BP with our reordering scheme (denoted SFR-BP).

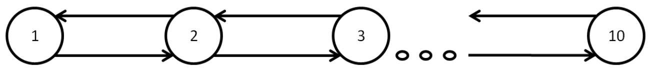


Fig. 4: Configuration with 10 nodes. The capacity of all links is identical.

| Comm./Time | $t \leq 5000$ | $t > 5000$ |
|---|---|---|
| Commodity 1 | $\lambda_5^{(10)} = 0.1$ | $\lambda_5^{(10)} = 0$ |
| Commodity 2 | $\lambda_6^{(9)} = 0.9$ | $\lambda_6^{(9)} = 1$ |
| Commodity 3 | $\lambda_7^{(8)} = 8$ | $\lambda_7^{(8)} = 8$ |

Fig. 5: Assumptions on the Poisson arrival rates of the three commodities according to simulation time, to illustrate the elimination of the *starvation* and the *last-packet problem*.

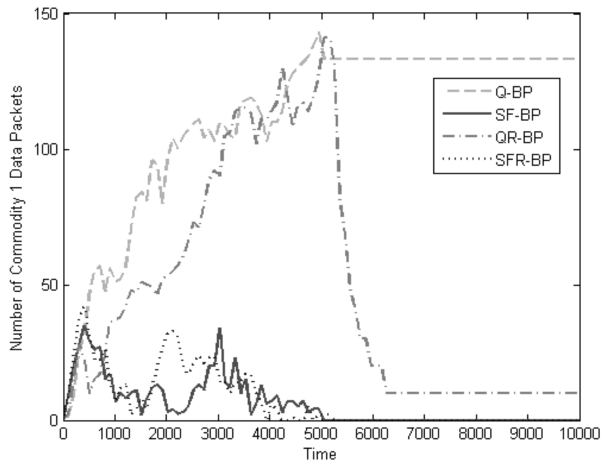


Fig. 6: Elimination of the *last-packet problem*. The figure plots the number of Non-Null packets of commodity 1 residing in the network as time evolves. It shows how following the *last-packet problem*, the traditional backpressure scheme cannot empty the network, even with the reordering feature, while our suggested algorithms do.

## A. Elimination of starvation and the last-packet problem

To illustrate the elimination of the *starvation* and the *last-packet* problems, we consider the network presented in Figure 4 and a scenario with three commodities, namely, nodes 5, 6 and 7 are the sources, and nodes 10, 9 and 8 are the respective destinations. The capacity of all links is 10. The Poisson arrival rates of the three commodities are presented in Figure 5. This scenario is especially challenging for the weak stream of commodity 1, which faces two challenges, namely: (a) nodes 1-4 present a potential well for packets; and (b) the stronger streams of commodities 2 and 3 cause *starvation* at nodes 6-9. When using our starvation-free scenario, guided by the intuition gained from Corollary 4, we assign $d_{i,\max}^{(c)} = 0\,\forall i, c$, and $\delta_i^{(10)} = \frac{0.5-\varepsilon}{|V|}\,\forall i \neq 10$, $\delta_i^{(9)} = \frac{0.4-\varepsilon}{|V|}\,\forall i \neq 9$ and $\delta_i^{(8)} = \frac{0.1-\varepsilon}{|V|}\,\forall i \neq 8$ where $\varepsilon = 10^{-4}$, since we want to enhance the weaker streams.

| Comm./Alg. | Commodity 2 | Commodity 3 |
|---|---|---|
| Q-BP | 214 | 84 |
| SF-BP | 49 | 29 |
| QR-BP | 47 | 41 |
| SFR-BP | 32 | 26 |

Fig. 7: Elimination of the *last-packet problem*. Mean delay of packets in number of time slots. Only the best $80\%$ of the packets are taken into account, since in traditional schemes the *last-packet problem* can prevent some packets from ever leaving the network.

Figures 6 and 7 present the results. When using the backpressure techniques (denoted Q-BP in the usual form, and QR-BP with our suggested reordering scheme), 133 and 10 packets (respectively) of commodity 1 remain in the network for a potentially unbounded time. On the other hand, in both of our starvation-free techniques (denoted SF-BP in the simple form, and SFR-BP with the reordering improvement), there is a complete elimination of the *last-packet problem*, and the delays of all streams are relaxed significantly. The use of SF-BP and SFR-BP results in the delivery of all packets of commodity 1 to their destination shortly after the stream stops.

## B. Performance enhancement

We consider the example presented in Figure 8. This network presents routing challenges, since nodes 4, 5, and 6 create a potential well for packets. When using our starvation-free algorithm, guided by the intuition gained from Section IV-E, we assign $d_{i,\max}^{(12)} = 0\,\forall i$ and $\delta_i^{(12)} = \frac{(\lambda_{\max}-\lambda-\varepsilon)}{|V|}$ for indices $i \neq 12$, where $\varepsilon = 0.0001$ and $\lambda_{\max}, \lambda$ are, respectively, the maximum possible load and the given load to the source.

Figure 9 presents the results. There is a significant delay reduction in both of our techniques compared to the queue-length based scenes for the entire range of the load parameter. Additionally, in both of our starvation-free scenarios, the mean delay of the 99th percentile of packets exhibits a $53\%$ decrease in SF-BP compared to Q-BP and a $45\%$ decrease in SFR-BP compared to QR-BP.

This example shows that the SF-PB algorithm improves performance in terms of mean queue occupancy and end-to-end packet delay even in the case of a single commodity. As can be seen in Figure 10, for both our starvation-free schemes, the mean number of data packets in the networks is lower than in Q-BP and QR-BP. In addition, for all scenarios, there is no packet accumulation in the intermediate queues.

## C. Random topologies

In this experiment we randomly generated 30 different topologies using the scale-free Barabasi-Albert model [1] with out-degree distribution equal to 2 for $|V| = 12$. There are two streams with Poisson arrivals and parameters $\lambda_1^{(12)}$ and $\lambda_2^{(11)}$ such that $\lambda_1^{(12)}$ is $80\%$ of the maximal flow between nodes 1 and 12 and $\lambda_2^{(11)}$ is $10\%$ of the maximal flow between

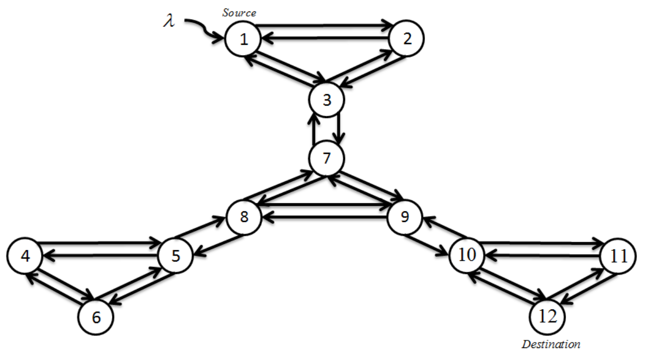


Fig. 8: Configuration with $|V| = 12$. All link capacities are i.i.d. between time slots and follow a geometric random distribution of mean value 5. Node 1 is the source having exogenous Poisson arrival process with parameter $\lambda$, and node 12 is the destination.

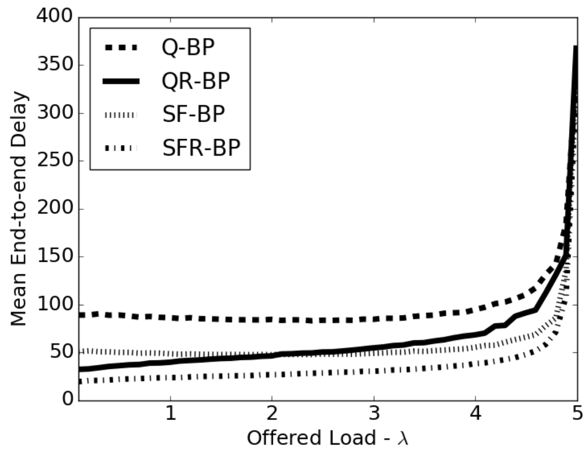


Fig. 9: Mean delay of packets as a function of the load offered to the source, using the configuration of Figure 8.

nodes 2 and 11. For our starvation-free scenarios, guided by the intuition gained from Section IV-E, we assign $d^{(c)}_{i,\max} = 0\,\forall i, c$ and $\delta^{(12)}_i = \frac{0.02-\varepsilon}{|V|}\,\forall i \neq 12$ and $\delta^{(11)}_i = \frac{0.08-\varepsilon}{|V|}\,\forall i \neq 11$ where $\varepsilon = 10^{-4}$, since we want to enhance the weaker stream.

## D. Static random topologies

The results for the first ten randomly generated topologies are illustrated in Figures 11a and 11c . In all randomly created topologies, using our starvation-free scenarios results in a performance enhancement of both commodities. Specifically, there is a 65% decrease in the mean delay of all 30 topologies in SF-BP compared to Q-BP, and a 48% decrease in SFR-BP compared to QR-BP. Additionally, the mean delay over all topologies of the weaker commodity destined to node 11 is decreased by a factor of 2.44 when compared to the standard technique. Finally, in *all* of these random topologies, the mean delay of the stronger commodity decreased as well, with the exception of similar performance in trivial topologies, i.e. the

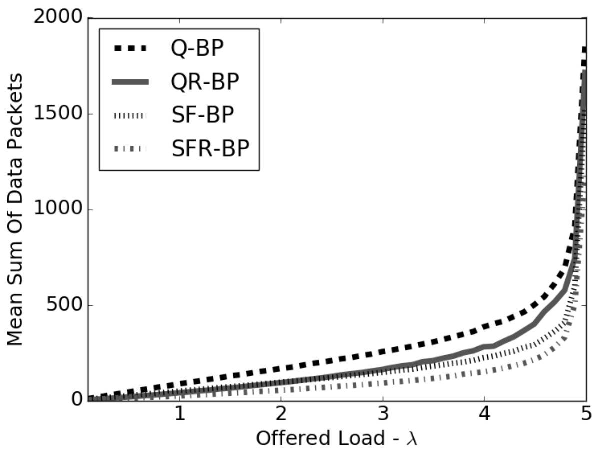


Fig. 10: Mean sum of data packets in the network as a function of the load offered to the source, using the configuration of Figure 8.

improvement of the weaker commodity was not at the expense of the stronger one.

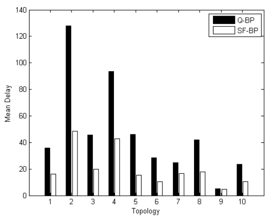
## E. Dynamic random topologies

In this experiment, the link capacities in the randomly created graphs are $i.i.d.$ from slot to slot and are geometrically distributed with mean value of 5. The results are presented in Figures 11b and 11d. In all randomly generated topologies, using our starvation-free scenarios results in a performance enhancement[3] for both commodities. However, as expected, the enhancement of our starvation-free scenarios is smaller than in the static case since the delay and congestion information are less effective when used in a network in which the topology changes occur on a per-slot basis. Specifically, there is a 44% decrease in the mean delay of all 30 topologies in SF-BP compared to Q-BP, and a 13% decrease in SFR-BP compared to QR-BP. Additionally, the mean delay over all topologies of the weaker commodity destined to node 11 decreased by up to 140% when compared to the standard technique, while the mean delay of the stronger commodity decreased as well again in *all* topologies. In fact, in all of our simulations with both dynamic and static topologies, we never found a non-trivial case when Q-BP outperformed SF-BP for some commodity.

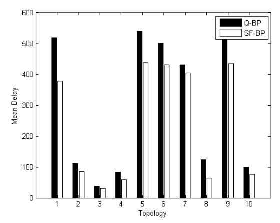
## VII. Conclusions

We introduced the *starvation-free stability* criterion, and established the first starvation-free stable backpressure routing-and-resource-allocation algorithm for multi-hop dynamic networks. We showed that our algorithm fully overcomes both the *starvation* and the *last-packet* problems while providing

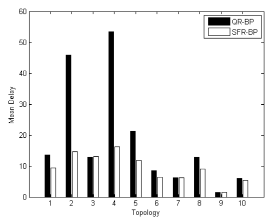
[3]Note that in topology 3 in Figure 11d for example, when using the reordering scheme, the measured mean delay of the weak commodity is similar in SF-BP and QR-BP. However, a closer investigation showed that the mean delay of packets in the network was much higher in QR-BP due to *starvation*. Taking this into consideration yielded the claimed result.



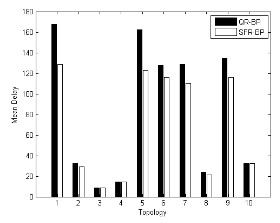
(a) First 10 *static* topologies. Mean delay of $\lambda_2^{(11)}$.



(b) First 10 *dynamic* topologies.



(c) Reordering scheme: First 10 *static* topologies.



(d) Reordering scheme: First 10 *dynamic* topologies.

Fig. 11: Mean delay of the weak commodity $\lambda_2^{(11)}$ for both static and dynamic randomly generated topologies.

100% throughput. Furthermore, we established a stronger per-queue mean evacuation period guarantee, and we provided schemes for the enhancement of weak streams. We formally established that our algorithm ensures that all packets reach their destination in wide classes of networks. In addition, we presented a packet reordering scheme that further reduces the maximal delays of packets in the network and relaxes the out-of-order arrivals to the destinations. Finally, we verified our results through extensive simulations using particularly challenging topologies as well as random static and dynamic topologies.

## VIII. Acknowledgments

The authors would like to thank Gal Mendelson for useful discussions, as well as associate editor Adam Wierman and the anonymous reviewers for their helpful comments. This work was partly supported by the Hasso Plattner Institute Research School, the Israel Ministry of Science and Technology, the Gordon Fund for Systems Engineering, the Technion Fund for Security Research, and the Shillman Fund for Global Security.

## References

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] M. Alresaini, K.-L. Wright, B. Krishnamachari, and M. J. Neely. Backpressure delay enhancement for encounter-based mobile networks while sustaining throughput optimality. *IEEE/ACM Trans. Netw.*, 2016.

[3] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. In *IEEE Infocom*, pages 2936–2940, 2009.

[4] Y. Cui, E. M. Yeh, and R. Liu. Enhancing the delay performance of dynamic backpressure algorithms. *IEEE/ACM Trans. Netw.*, 2016.

[5] A. Dimakis and J. Walrand. Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits. *Advances in Applied probability*, pages 505–521, 2006.

[6] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. In *IEEE Infocom*, volume 3, pages 1794–1803, 2005.

[7] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc, 2006.

[8] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. *IEEE/ACM Trans. Netw.*, 17(6):1846–1859, 2009.

[9] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari. LIFO-backpressure achieves near-optimal utility-delay tradeoff. *IEEE/ACM Trans. Netw.*, 21(3):831–844, 2013.

[10] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *IEEE Auto. Con.*, 56(4):842–857, 2011.

[11] B. Ji, C. Joo, and N. B. Shroff. Delay-based back-pressure scheduling in multihop wireless networks. *IEEE/ACM Trans. Netw.*, 21(5):1539–1552, 2013.

[12] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Trans. Netw.*, 18(3):960–972, 2010.

[13] C. Joo, X. Lin, and N. B. Shroff. Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks. *IEEE/ACM Trans. Netw.*, 17(4):1132–1145, 2009.

[14] B. Li, A. Eryilmaz, and R. Srikant. On the universality of age-based scheduling in wireless networks. *IEEE Infocom, 2015.*

[15] B. Li, R. Li, and A. Eryilmaz. Throughput-optimal scheduling design with regular service guarantees in wireless networks. *IEEE/ACM Trans. Netw.*, 23(5):1542–1552, 2015.

[16] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proc. IEEE Decision and Control*, 2004.

[17] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. In *Proc. ACM/IEEE IPSN*, 2010.

[18] M. J. Neely. Dynamic power allocation and routing for satellite and wireless networks with time varying channels. *MIT*, 2003.

[19] M. J. Neely. Queue stability and probability 1 convergence via Lyapunov optimization. *arXiv preprint arXiv:1008.3519*, 2010.

[20] M. J. Neely. Stability and capacity regions or discrete time queueing networks. *arXiv preprint arXiv:1003.3396*, 2010.

[21] M. J. Neely. Delay-based network utility maximization. *IEEE/ACM Trans. Netw.*, 21(1):41–54, 2013.

[22] J. Ni and R. Srikant. Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks. In *IEEE Trans. Inform. Theory*, pages 250–250, 2009.

[23] A. Rai, C.-P. Li, G. Paschos, and E. Modiano. Loop-free backpressure routing using link-reversal algorithms. In *Proc. ACM MobiHoc*, 2015.

[24] B. Sadiq and G. De Veciana. Throughput optimality of delay-driven maxweight scheduler for a wireless system with flow dynamics. In *IEEE Allerton Conference*, pages 1097–1102, 2009.

[25] A. U. Shankar, C. Alaettinoglu, K. Dussa-Zieger, and I. Matta. Transient and steady-state performance of routing protocols: Distance-vector versus link-state. In *Internetworking: Research and Experience*, 1996.

[26] A. Sridharan, S. Moeller, B. Krishnamachari, and M. Hsieh. Implementing backpressure-based rate control in wireless networks. In *IEEE Trans. Inform. Theory*, pages 341–345, 2009.

[27] A. L. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, 50(4):401–457, 2005.

[28] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Auto. Con.*, 37(12):1936–1948, 1992.

[29] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. Inform. Theory*, 39(2):466–478, 1993.

[30] S. Vargaftik, I. Keslassy, and A. Orda. Stable user-defined priorities. In *IEEE Infocom*, pages 1935–1943, 2017.

[31] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee. DiffQ: Practical differential backlog congestion control for wireless networks. In *IEEE Infocom*, pages 262–270, 2009.

[32] X. Wu, R. Srikant, and J. R. Perkins. Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. *IEEE Trans. on Mobile Computing*, 6(6):595–605, 2007.

[33] E. Yeh and R. Berry. Throughput optimal control of wireless networks with two-hop cooperative relaying. *IEEE Trans. Inform. Theory*, pages 351–355, 2007.

[34] L. Ying, S. Shakkottai, A. Reddy, and S. Liu. On combining shortest-path and back-pressure routing over multihop wireless networks. *IEEE/ACM Trans. Netw.*, 19(3):841–854, 2011.

[35] H. Yu and M. J. Neely. A new backpressure algorithm for joint rate control and routing with vanishing utility optimality gaps and finite queue lengths. In *IEEE Infocom*, pages 1926–1934, 2017.

## Appendix

### A. Proof of Lemma 3

We observe that for any $x, y \in \mathbb{R}$ it holds that:

$$x = \min\{x, y\} + \max\{x - y, 0\}. \tag{33}$$

Next using (33) we make the following observations:

$$\begin{aligned}\sum_b \mu_{ib}^{(c)}(t) = \min\left\{U_i^{(c)}(t), \sum_b \mu_{ib}^{(c)}(t)\right\} \\ + \max\left\{\sum_b \mu_{ib}^{(c)}(t) - U_i^{(c)}(t), 0\right\},\end{aligned} \tag{34}$$

$$\begin{aligned}d_{i,out}^{(c)}(t) = \min\left\{d_{i,out}^{(c)}(t), d_{i,\max}^{(c)}\right\} \\ + \max\left\{d_{i,out}^{(c)}(t) - d_{i,\max}^{(c)}, 0\right\},\end{aligned} \tag{35}$$

$$\begin{aligned}&U_i^{(c)}(t+1) = \max\left\{U_i^{(c)}(t) - \sum_b \mu_{ib}^{(c)}(t), 0\right\} + \\ &\sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) = U_i^{(c)}(t) - \\ &\min\left\{U_i^{(c)}(t), \sum_b \mu_{ib}^{(c)}(t)\right\} + \sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t).\end{aligned} \tag{36}$$

We additionally apply (33) to (15) and obtain:

$$\begin{aligned}\tilde{d}_{i,out}^{(c)}(t) = \min\left\{d_{i,out}^{(c)}(t), d_{i,\max}^{(c)}\right\} = \\ d_{i,out}^{(c)}(t) - \max\left\{d_{i,out}^{(c)}(t) - d_{i,\max}^{(c)}, 0\right\}\end{aligned} \tag{37}$$

Next, we reexamine Definition 4 and obtain:

$$\begin{aligned}&Z_i^{(c)}(t+1) = \\ &\max\left(Z_i^{(c)}(t) - \sum_b \mu_{ib}^{(c)}(t) - \delta_i^{(c)} \tilde{d}_{i,out}^{(c)}(t), 0\right) + \\ &\sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) + \delta_i^{(c)} \cdot d_{i,in}^{(c)}(t) \geq \\ &Z_i^{(c)}(t) - \sum_b \mu_{ib}^{(c)}(t) - \delta_i^{(c)} \cdot \tilde{d}_{i,out}^{(c)}(t) + \\ &\sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) + \delta_i^{(c)} \cdot d_{i,in}^{(c)}(t).\end{aligned} \tag{38}$$

Applying (34), (35), (36) and (37) to (38) yields:

$$\begin{aligned}&Z_i^{(c)}(t+1) \geq \\ &Z_i^{(c)}(t) - \sum_b \mu_{ib}^{(c)}(t) - \delta_i^{(c)} \cdot \tilde{d}_{i,out}^{(c)}(t) + \\ &\sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) + \delta_i^{(c)} \cdot d_{i,in}^{(c)}(t) = \\ &Z_i^{(c)}(t) + \sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) + \delta_i^{(c)} d_{i,in}^{(c)}(t) - \\ &\min\left\{\sum_b \mu_{ib}^{(c)}(t), U_i^{(c)}(t)\right\} - \\ &\max\left\{\sum_b \mu_{ib}^{(c)}(t) - U_i^{(c)}(t), 0\right\} - \\ &\delta_i^{(c)}\left(d_{i,out}^{(c)}(t) - \max\left\{d_{i,out}^{(c)}(t) - d_{i,\max}^{(c)}, 0\right\}\right).\end{aligned} \tag{39}$$

Rearranging (39) yields:

$$\begin{aligned}&Z_i^{(c)}(t+1) - Z_i^{(c)}(t) \geq \\ &\sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) - \min\left\{\sum_b \mu_{ib}^{(c)}(t), U_i^{(c)}(t)\right\} \\ &+ \delta_i^{(c)}\left(d_{i,in}^{(c)}(t) - d_{i,out}^{(c)}(t)\right) - \\ &\max\left\{\sum_b \mu_{ib}^{(c)}(t) - U_i^{(c)}(t), 0\right\} + \\ &\delta_i^{(c)} \max\left\{d_{i,out}^{(c)}(t) - d_{i,\max}^{(c)}, 0\right\}.\end{aligned} \tag{40}$$

Applying (13) and (36) to (40) yields:

$$\begin{aligned}&Z_i^{(c)}(t+1) - Z_i^{(c)}(t) \geq \\ &\left(U_i^{(c)}(t+1) - U_i^{(c)}(t)\right) - \\ &\max\left\{\sum_b \mu_{ib}^{(c)}(t) - U_i^{(c)}(t), 0\right\} + \\ &\delta_i^{(c)}\left(D_i^{(c)}(t+1) - D_i^{(c)}(t)\right) + \\ &\delta_i^{(c)} \max\left\{d_{i,out}^{(c)}(t) - d_{i,\max}^{(c)}, 0\right\}.\end{aligned} \tag{41}$$

Rearranging (41), summing over time slots from $\tau = 0$ to $\tau = t-1$ and dividing by $t$ yields:

$$\begin{gathered}\frac{1}{t}\sum_{\tau=0}^{t-1} P_i^{(c)}(\tau) \le \frac{1}{t}\left(Z_i^{(c)}(t) - Z_i^{(c)}(0)\right) - \\ \frac{1}{t}\left(U_i^{(c)}(t) - U_i^{(c)}(0)\right) - \frac{\delta_i^{(c)}}{t}\left(D_i^{(c)}(t) - D_i^{(c)}(0)\right).\end{gathered} \tag{42}$$

Taking limits of (42) and applying (16) and (17) yields:

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{t=0}^{t-1} P_i^{(c)}(\tau) \le \limsup_{t\to\infty} \frac{Z_i^{(c)}(t)}{t}. \tag{43}$$

Next, we reexamine Definition 4. Similarly to the analysis presented in Theorem 1, we define two auxiliary stochastic precesses $a(t)$ and $b(t)$ as follows:

$$\begin{gathered} Z_i^{(c)}(t+1) = \\ \max\left( Z_i^{(c)}(t) - \underbrace{\left(\sum_b \mu_{ib}^{(c)}(t) + \delta_i^{(c)} \cdot \tilde{d}_{i,out}^{(c)}(t)\right)}_{b(t)}, 0 \right) \\ + \underbrace{\sum_a \mu_{ai}^{(c)}(t) + A_i^{(c)}(t) + \delta_i^{(c)} \cdot d_{i,in}^{(c)}(t)}_{a(t)}. \end{gathered} \tag{44}$$

We observe that, again, similarly to the analysis presented in the proof of Theorem 1, $Z_i^{(c)}(t)$ respects the boundedness condition that is required by Theorem 4 of [20] (case (b)) for a stochastic process with *queue dynamics*. Thus, applying Lemma 1 and Theorem 4 of [20] on $Z_i^{(c)}(t)$ yields the result.

## B. *Proof of Lemma 4*

Rearranging (13), summing over time slots $[0, 1, ..., t-1]$ and dividing by $t$ yields:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\left[d_{i,out}^{(c)}(\tau) - d_{i,in}^{(c)}(\tau)\right] = \frac{1}{t}\left(D_i^{(c)}(0) - D_i^{(c)}(t)\right) \tag{45}$$

Taking the limit of (45) and applying (16) and (17) yields:

$$\lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\left[d_{i,out}^{(c)}(\tau) - d_{i,in}^{(c)}(\tau)\right] = -\lim_{t\to\infty} \frac{D_i^{(c)}(t)}{t}. \tag{46}$$

Next, according to Theorem 3, we obtain starvation-free stability. Thus, we apply (9) on (46) and this yields the result.

**Shay Vargaftik** received the B.Sc. degree from the Viterbi department of Electrical Engineering, Technion - Israel Institute of Technology, in 2012. He is now pursuing a Ph.D. degree in the same department. He is mainly interested in dynamic networks, queuing networks, data-center networks as well as in algorithm design and analysis. He was the recipient of the Hasso Plattner Institute and the IBM Ph.D. fellowship awards.

**Isaac Keslassy** (M'02, SM'11) received his M.S. and Ph.D. degrees in Electrical Engineering from Stanford University, Stanford, CA, in 2000 and 2004, respectively. He is currently an associate professor in the Viterbi department of Electrical Engineering at the Technion, Israel. His recent research interests include the design and analysis of data-center networks and high-performance routers. He was the recipient of the Allon, Mani, Yanai, and Taub awards, as well as an ERC Starting Grant and the ACM SIGCOMM'15 test-of-time award. He was associate editor for the IEEE/ACM Transactions on Networking, and TPC chair for ACM/IEEE ANCS'14.

**Ariel Orda** (S'84, M'92, SM'97, F'06) received the B.Sc. (summa cum laude), M.Sc., and D.Sc. degrees from the Technion - Israel Institute of Technology, Haifa, Israel, in 1983, 1985, and 1991, respectively, all in electrical engineering. Since 1994, he has been with the Viterbi Faculty of Electrical Engineering, Technion - Israel Institute of Technology, where he is currently the Herman and Gertrude Gross Professor of Communications. Since 2014, he has been the Dean of the Viterbi Faculty of Electrical Engineering at the Technion - Israel Institute of Technology. His research interests include network routing, the application of game theory to computer networking, survivability, QoS provisioning, wireless networks, and network pricing. He received several awards for research, teaching and service.