Stable User-Defined Priorities

Shay Vargaftik, Isaac Keslassy, Ariel Orda
Technion
{shayvar@tx, isaac@ee, ariel@ee}.technion.ac.il

***Abstract*—Network providers now want to enable users to define their own flow priorities, and commercial devices already implement this ability. However, it has been shown that directly applying arbitrary user-defined priorities can fundamentally destabilize a network.**

In this paper, we show that it is possible to apply user-defined priorities while keeping the network stable. We introduce U-BP, a scalable approach that extends backpressure-based scheduling techniques to service user-defined flow priorities and rates while maintaining throughput optimality and strong network performance. We explain how our approach relies on a dual-layer scheme with an exponential convergence to requested priorities. We further prove analytically the network stability of our solution, and show how it achieves a strong performance for high-priority flows.

I. Introduction

This paper is about the emerging topic of *user-defined priorities*, *i.e.,* giving users the ability to dynamically choose the priorities of their network flows. User-defined priorities have only recently been prominently featured in the academic literature, and especially in a recent paper on designing a user-centric network [1]. It is building on [2], [3], which expose an API and browser add-ons for users to dynamically express their preferences.

User-defined traffic prioritization is particularly appealing to network providers, because it can help them provide more value to their customers using preferential treatment dictated by the users themselves, without conflicting with network neutrality regulations. In fact, user-defined prioritization has already been implemented in the industry. OnHub, a commercial home WiFi router built by Google, already enables Boost, a user-defined prioritization of certain devices and applications [4]. It first relied on a simple Chrome add-on that allows prioritizing all flows related to specific open tabs. In the Asus OnHub, it has since enabled prioritization using a simple hand-waving above the router. Early users showed widely varying preferences. For instance, some users preferred to prioritize business calls; while others preferred to prioritize their Netflix streams over those of their children [1].

Unfortunately, while network providers are eager to enable user-defined priorities, it is unclear whether, and under what conditions, they can fundamentally keep a dynamic network stable when introducing dynamic priorities. In fact, [5] showed that while backpressure algorithms typically guarantee stable networks under either no priorities or fixed priorities, such algorithms break down and do not guarantee stability anymore under user-defined dynamic priorities. To our knowledge, no algorithm today can provide such guarantees.

The goal of this paper is to analyze whether it is fundamentally impossible to provide network stability for user-defined priorities. We find that it is in fact possible. We propose *U-BP* (User-defined BackPressure), a scalable approach that extends backpressure-based scheduling techniques to provide a user-defined flow-based quality of service while maintaining throughput optimality and overall network performance. In particular, we make the following contributions:

Alg. / Criterion	Priorities	Stability	Mean total queue length
Q-BP	×	✓	$O(n^2)$
m-Q-BP	✓ˣ	✓ˣ	$O(mn^2)$
m-DRPC	✓	×	—
U-BP	✓	✓	$O(n^2)$

TABLE I. Comparison between the standard *Q-BP* backpressure algorithm, *m-Q-BP*, *m-DRPC* and our solution *U-BP* for variable-rate traffic with user-defined priorities, given n nodes and m priorities (see Section III for algorithm details).

User-defined priorities and rates. While current papers typically assume fixed expected flow rates with fixed priorities, we allow the users to set their flow priorities in any arbitrary way, and send any arbitrary set of flows as long as it is upper-bounded by an admissible process.

Dual-layer approach. As illustrated in Table 1, simple backpressure-based approaches cannot efficiently handle the user-defined priorities, and either become unstable or obtain poor performance. Instead, we suggest using a dual-layer approach, *i.e.,* relying on virtual queues to handle the different priorities, while only informing the upper-layer backpressure scheme about the aggregate queue occupancy.

Weighted average priority. In order to reflect the difference in priorities between different destination-based commodities, we multiply the aggregate occupancy by the weighted-average packet priority when informing the backpressure scheme.

Gradual priority updates. Since updating priorities immediately throughout the network can lead to jumps of potential and to instability, we update priorities gradually. Specifically, we increase priorities exponentially fast, and decrease them immediately. We explain why this is a fundamental condition, and demonstrate how to optimize the priority update speed.

Lyapunov-based stability. We introduce an appropriate Lyapunov-based potential and prove the stability of the network for user-defined priorities.

Simulations. Finally, in simulations, we illustrate how our *U-BP* algorithm achieves significantly reduced delays for high-priority traffic while maintaining overall network performance.

II. System model

Network. We consider a network of n nodes. Each node can both be a source and a destination for data. We assume that the network operates in slotted time $t \in \mathbb{N}$. We denote by $S(t)$ the topology state of the network at time slot t, *i.e.,* the link capacities and interference model (*e.g.,* specifying that two links from the same node may not be used in the same slot because of wireless constraints). We assume that $S(t)$ takes values within a finite state space $\mathcal{S}$ and is i.i.d. between time slots.

User-defined flows. We want to introduce user-defined flows with user-defined priorities. In contrast with most of the literature, we cannot assume that the expected rate of flows belonging to a given (source, destination, priority)-tuple is fixed, because we want to allow users to mark and remark any application flow with any priority at any time. Instead, *we only assume that the expected total incoming traffic is upper-bounded by an admissible process*. We allow the priorities of packets to freely vary with time. The remainder of the model then follows well-defined literature conventions.

Specifically, we assume that there are m priorities. We also define per-destination commodities, so that packets that are destined to destination c belong to commodity c. We then define an *upper bound* $A_i^{(c)}(t)$ on the number of packets in commodity c that exogenously arrive to node i at time slot t, including all priorities. We assume that $\forall t$:

$$A_i^{(c)}(t) \text{ is i.i.d. between time slots,} \tag{1}$$

$$\mathbb{E}\left[A_i^{(c)}(t)\right] = \lambda_i^{(c)}, \tag{2}$$

$$A_i^{(c)}(t) \leq A_{i,max}^{(c)}. \tag{3}$$

Finally, we denote by $\lambda = \left(\lambda_i^{(c)}\right)$ the upper bound on the exogenous arrival rate vector to the network.

Scheduling. At each time slot t, we want to choose a schedule $\pi(t)$ that satisfies the topology constraints given by $S(t)$. We denote by $C_{ij}(S(t), \pi(t))$ the capacity of link (i,j) at time slot t. It is expressed in number of packets, assumed to be of fixed size. Additionally we assume that:

$$C_{ij}(S(t), \pi(t)) \leq C_{ij,max} \quad \forall t. \tag{4}$$

Each node holds one queue per commodity, so that it needs at most $n-1$ queues for storing packets destined to the other destination nodes. $U_i^{(c)}(t)$ is the number of packets that reside in the queue of commodity c in node i at time slot t. The transmission rate offered to commodity c in node i over link (i,j) at time slot t is denoted by $\mu_{ij}^{(c)}(t)$, and satisfies

$$\sum_c \mu_{ij}^{(c)}(t) \leq C_{ij}(S(t), \pi(t)) \leq C_{ij,max}. \tag{5}$$

Packets that exogenously or endogenously enter a queue during time slot t are admitted only at time slot $t+1$, and therefore cannot be forwarded during the time slot of their arrival. Additionally, packets that arrive at their destination are assumed to immediately leave the network.

Capacity region. We define the capacity region of the network Λ to be the closure of the set of all possible arrival vectors λ that the network can support with respect to the scheduling constraints. Additionally, if $\lambda \in \Lambda$ we say that λ is *admissible*, and if $\lambda + \varepsilon \in \Lambda$ for some $\varepsilon > 0$, we say that λ is *strictly admissible*. In addition, assuming λ is *admissible*, there exists an algorithm [6] in which the following holds for all $\forall t, i, c$:

$$\mathbb{E}\left[\sum_b \mu_{ib}^{(c)}(t) - \sum_a \mu_{ai}^{(c)}(t)\right] = \lambda_i^{(c)}, \tag{6}$$

where the expectation is taken with respect to $S(t)$, and possibly a random choice of $\pi(t)$ that belongs to the set $\pi_S(t)$ of all possible schedules at time slot t. Additionally, this algorithm is shown to be dependent only on $S(t)$, thus independent of any backlog information at time t.

Backpressure: *Q-BP*. As a baseline for our algorithm, for ease of exposition, we use the standard queue-length-based backpressure algorithm (*Q-BP*), which does not take priorities into account. However, our technique can be naturally extended to a wide range of potential functions that can be represented using *queue dynamics* and are amenable to formal analysis via the Lyapunov drift technique. The weight of each link in the network is defined as follows:

$$W_{ij}(t) = \max\left[U_i^{\left(c_{ij}^{opt}(t)\right)}(t) - U_j^{\left(c_{ij}^{opt}(t)\right)}(t), 0\right], \tag{7}$$

where

$$c_{ij}^{opt}(t) \triangleq \arg\max_c \left[U_i^{(c)}(t) - U_j^{(c)}(t)\right]. \tag{8}$$

With these weights at hand, the solution of the following optimization problem dictates the algorithm's choice of the scheduling among the different links.

$$\begin{array}{ll} \max & \sum_{ij} W_{ij}(t)\, C_{ij}(S(t), \pi(t)) \\ \text{s.t.:} & \pi(t) \in \pi_S(t). \end{array} \tag{9}$$

Then, for each strictly positive $W_{ij}(t)$, the network offers a transmission rate as follows:

$$\mu_{ij}^{(c)}(t) = \begin{cases} C_{ij}(S(t), \pi(t)) & \text{if } c = c_{ij}^{opt}(t) \\ 0 & \text{else.} \end{cases} \tag{10}$$

Following standard practice [7], whenever there are not enough packets for commodity $c_{ij}^{opt}(t)$ in node i to fill the allocated capacity as dictated by Eq. (10), it virtually transmits *null* packets (*i.e.,* in practice it simply asks to update the null packet counters).

III. The *U-BP* Algorithm

We now want to introduce our *U-BP* (User-defined Back-Pressure) algorithm. We will successively introduce and explain its three main components: (1) *dual-layer approach*; (2) *weighted-average priority*; and (3) *gradual priority update*.

A. Dual-layer approach

The standard Q-BP backpressure algorithm does not differentiate between priorities. At each node, all packets of a given commodity c are stored together in a single FIFO queue. Instead, we want to differentiate between the packets of the m different priorities. We assume that each priority $k \in \{1, \ldots, m\}$ is associated with a fixed weight α_k, and that the weight of a queue with priority-k packets is equal to the

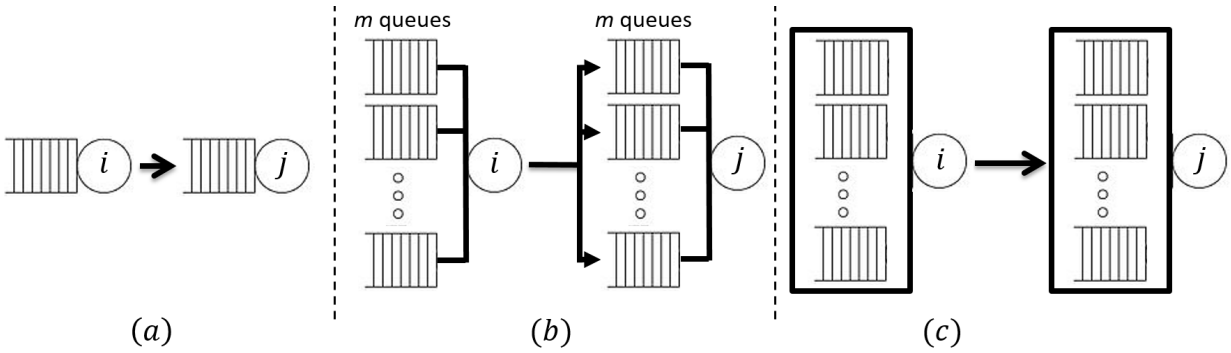


Fig. 1. Implementation of a single-commodity topology with m priorities. Focus on a single-link using different approaches. Comparison between (a) the standard *Q-BP* technique, in which all priority classes are queued in the same FIFO queue; (b) the single-layer solution for *m-DRPC* and *m-Q-BP* that isolates between the m priorities and reflects m commodities instead of one to the backpressure algorithm; and (c) our proposed dual-layer virtual-queue abstraction that preserves the number of commodities observed by the backpressure algorithm.

product of α_k by its occupancy. We assume that the weights are ordered and positive, *i.e.,* $0 < \alpha_1 < \cdots < \alpha_m$, and denote $A = (\alpha_1, \ldots, \alpha_m)$. We first consider two simple approaches before introducing our own approach.

Single-layer solution: *m-DRPC*. A first natural approach is to use m different FIFO queues for the m priorities. Therefore, instead of building a per-destination commodity, we define a per-(destination,priority) commodity. We can do so by (a) replicating each destination node into m virtual priority-based destination nodes, and (b) applying the DRPC priority-service algorithm framework [5], which enables backpressure to handle queues with different fixed priorities and fixed arrival rates to these queues. We denote this first algorithm *m-DRPC*.

Unfortunately, while this natural approach seems promising, [5] provides a counter-example to user-defined varying priorities that can be directly extended to demonstrate that *m-DRPC* may also lead to unstable networks even when the arrival rates are strictly within their capacity region.

Single-layer solution: *m-Q-BP*. We can also consider a simpler approach that we believe can guarantee network stability and provide priority isolation, but does not provide priority-based weights. We call this approach *m-Q-BP*. In this approach, we use again the solution described above, but normalize all weights to 1. That is, we simply tell the backpressure algorithm that each queue represents another commodity, and its queue weight is equal to its queue size.

Unfortunately, this algorithm has two major problems. First, while it guarantees priority isolation, it does not provide a better service for high-priority flows. Second, since in the backpressure algorithm the upper bound on mean queue lengths is proportional to the number of queues in the network, even with fixed priority rates, such a solution multiplies this bound by a factor of m. In turn, this also incurs a prohibitive penalty in terms of mean delay bounds by Little's Law (assuming fixed arrival rates).

Example 1. *Figure 1(a) provides a toy example to gain insight into this observation (more complex topologies are evaluated in the evaluation section). Consider a single-commodity topology with a link capacity of $C_{ij} = m$. Additionally, assume that a single data packet arrives for each of the m priorities at each time slot. Using the standard queue-length differential-based backpressure technique results in a mean delay of* 1*, since link (i, j) can serve all packets of node i within a single time slot.*

Figure 1(b) illustrates the m-DRPC *and* m-Q-BP *algorithms based on a topology with m queues. Each queue at node i is served no more than once every m time slots. Thus, this approach results in a mean delay of $\Omega(m)$.*

Dual-layer solution. We want to be able to (a) still rely on the backpressure algorithm in order to obtain a 100% throughput guarantee; but also (b) hide the differentiation into several priorities from the backpressure algorithm, since we have seen that displaying it can lead to a worse performance.

To do so, we suggest using a *dual-layer solution*. As illustrated in Figure 1(c), in layer 1, we create m *virtual FIFO queues* corresponding to the m priorities. We place incoming packets within their respective virtual queues. In layer 2 (bold line), we only present to the backpressure algorithm the total number of packets in all the m virtual queues, since we do *not* want it to use the occupancy of each specific virtual queue. (Therefore, this actually differs from virtual output queues in input-queued switches, where virtual queues are created to enable the algorithm to use the occupancy of each virtual queue).

We then apply the simple Q-BP backpressure algorithm in layer 2 to decide on the queues to receive service. When the backpressure algorithm decides to serve a packet from the layer 2 buffer, we can read any head-of-line packet from the layer 1 virtual queues. We are guaranteed that the resulting algorithm will be stable. Specifically, since we want to favor higher priorities, we pick each priority-based virtual queue with a probability that is proportional to its priority weight. For instance, if we have two virtual queues with respective priority weights $\alpha_1 = 1$ and $\alpha_2 = 2$, we pick them with probabilities $1/3$ and $2/3$, respectively.

B. Weighted average priority

The above dual-layer approach guarantees network stability, and provides differentiated treatment between priorities *within* a commodity. However, it does not favor higher priorities *between different commodities*. The reason is that the layer 2 backpressure algorithm performs scheduling by only considering the number of packets in each node as the potential, and not incorporating their priority.

Instead, we change the way we reflect the node potential to the backpressure algorithm. Instead of setting the potential as equal to the sum of the virtual queue sizes, we set it as equal to the scalar product of the virtual queue sizes by their priorities; in other words, *we multiply the total queue size by the weighted average priority*. Formally, let $U_i^{(k,c)}(t)$ denote the queue size of virtual-queue k, and

$$U_i^{(c)}(t) = \sum_k U_i^{(k,c)}(t), \tag{11}$$

denote the total queue size. Then the potential function $\tilde{\Phi}_i^{(c)}(t)$ of a layer 2 queue is defined as

$$\begin{aligned}\tilde{\Phi}_i^{(c)}(t) &= \langle A, U_i^{(\cdot,c)}(t)\rangle \\ &= \sum_k \alpha_k \cdot U_i^{(k,c)}(t) = \tilde{\theta}_i^{(c)} \cdot U_i^{(c)}(t),\end{aligned} \tag{12}$$

where we use the weighted average priority

$$\tilde{\theta}_i^{(c)} = \frac{\sum_k \alpha_k \cdot U_i^{(k,c)}(t)}{\sum_k U_i^{(k,c)}(t)}. \tag{13}$$

In addition, Eq. (13) yields:

$$0 < \alpha_1 \leq \tilde{\theta}_i^{(c)}(t) \leq \alpha_m \ \forall t. \tag{14}$$

C. Gradual priority update

We have described above how our *U-BP* algorithm can group queues of different priorities by providing a priority-based differentiated service both within each commodity and between different commodities. However, such a scheme would have a poor performance with arbitrary user-defined priorities, because it allows the queue weights to jump much more than in a simple backpressure scheme. In fact, these jumps are precisely the reason behind the instability of the *m-DRPC* algorithm. Instead, we will introduce our most significant change, which will enable us to prove the network stability: we will typically update the priority increases *exponentially fast* but not *immediately*; and only update priority decreases immediately. Namely, as we show later, if some users increase their flow priorities such that the weighted average priority at some node for some commodity grows from 1 to 10, we will intuitively do so over $O(\log(10))$ slots instead of doing it in a single slot. We later explain and demonstrate why these gradual changes are crucial to preserving the network stability and lowering congestion in the presence of user-defined priorities.

More specifically, we will refer to the immediate user-defined priority $\tilde{\theta}_i^{(c)}$ as the *reference priority* of queue $U_i^{(c)}(t)$. For our algorithm, we define a smoother *effective priority* $\theta_i^{(c)}(t)$. Our objective is to find a suitable technique such that the effective priority $\theta_i^{(c)}(t)$ of each queue will quickly converge to the reference priority $\tilde{\theta}_i^{(c)}(t)$ upon a priority change, while preserving stability within the entire capacity region. Accordingly, we define the potential function of queue (i,c) as the product of its weighted average effective priority by its total (layer 2) queue length:

$$\Phi_i^{(c)}(t) = \theta_i^{(c)}(t) \cdot U_i^{(c)}(t), \tag{15}$$

with

$$0 < \alpha_1 \leq \theta_i^{(c)}(t) \leq \alpha_m \ \forall t. \tag{16}$$

Intuitively, the key idea behind the dual use of the priority weights for both layer 1 (probability to serve a given virtual queue) and layer 2 (weighted averaging of the total queue potential reflected to the backpressure algorithm) is to evenly share the load incurred by flow prioritization. When a high-priority flow enters a node with lower-priority flows sharing its destination, the potential of this total layer 2 queue grows proportionally as well. Thus, this high-priority flow fast-forwarding does not harm too much the low-priority flows of the same commodity, as the total service rate of the queue is proportionally increased.

IV. *U-BP* STABILITY WITH USER-DEFINED PRIORITIES

In this section, we formally analyze our approach. Specifically, we first obtain the *priority change regulation function*, a measurable metric to assess the impact of having different per-flow service classes on network performance. Then, we use this function to derive a sufficient condition for obtaining throughput optimality using the Lyapunov drift technique. Using these results, we continue by constructing the Lyapunov potentials used by *U-BP*. We also consider the additional optimization criterion of the convergence speed of our used gradual priority weights to the user-defined priority weights. Finally, we show exponentially fast convergence to the desired priorities as long as the network is not overloaded.

A. Priority change regulation function

Our goal is to obtain a measurable metric to assess the impact of a queue priority change on the system performance and on the capacity region of the network. To that end, we begin our analysis by defining the *priority change regulation function* $P_i^{(c)}(t)$:

Definition 1. *Let*

$$\Delta\theta_i^{(c)}(t) \triangleq \theta_i^{(c)}(t) - \theta_i^{(c)}(t-1)$$

be the change in the priority of queue (i,c) between two consecutive time slots. Then, we define the priority change regulation function $P_i^{(c)}(t)$ *of queue (i,c) as follows:*

$$P_i^{(c)}(t) \triangleq \Delta\theta_i^{(c)}(t) \cdot \left(U_i^{(c)}(t)\right)^2.$$

We make the observation that when the priority of a queue is changed, the algorithm enters a *transition period* during which *unregulated* changes of priorities may lead to suboptimal choices of resource allocation (in terms of throughput). For ease of exposition, we next prove the following lemma on the Lyapunov drift, which we later use in our stability theorem.

Lemma 1. *For all queues (i,c) and for all t,*

$$\begin{aligned}&\theta_i^{(c)}(t+1) \cdot \left(U_i^{(c)}(t+1)\right)^2 - \theta_i^{(c)}(t) \cdot \left(U_i^{(c)}(t)\right)^2 \\ &\leq \theta_i^{(c)}(t) \cdot B_{i,max}^{(c)}(t) + P_i^{(c)}(t+1) - \\ &2\Phi_i^{(c)}(t)\left(\sum_b \mu_{ib}^{(c)}(t) - \sum_a \mu_{ai}^{(c)}(t) - A_i^{(c)}(t)\right),\end{aligned}$$

where:

$$B_{i,max}^{(c)}(t)=\left(\sum_{b}\mu_{ib}^{(c)}(t)\right)^{2}+\left(\sum_{a}\mu_{ai}^{(c)}(t)+A_{i}^{(c)}(t)\right)^{2}.$$

Proof: We start our proof with the well-known queue dynamics equation (see [7]):

$$\begin{aligned}U_{i}^{(c)}(t+1)\leq\max\left\{U_{i}^{(c)}(t)-\sum_{b}\mu_{ib}^{(c)}(t),0\right\}\\+\sum_{a}\mu_{ai}^{(c)}(t)+A_{i}^{(c)}(t).\end{aligned}\tag{17}$$

We employ Lemma 4.3 of [7] on Eq. (17) and obtain:

$$\begin{aligned}\left(U_{i}^{(c)}(t+1)\right)^{2}\leq\left(U_{i}^{(c)}(t)\right)^{2}+\left(\sum_{b}\mu_{ib}^{(c)}(t)\right)^{2}+\\\left(\sum_{a}\mu_{ai}^{(c)}(t)+A_{i}^{(c)}(t)\right)^{2}-2\cdot U_{i}^{(c)}(t)\cdot\\\left(\sum_{b}\mu_{ib}^{(c)}(t)-\sum_{a}\mu_{ai}^{(c)}(t)\right)+2\cdot U_{i}^{(c)}(t)\cdot A_{i}^{(c)}(t).\end{aligned}\tag{18}$$

Rearranging Eq. (18) yields:

$$\begin{aligned}\left(U_{i}^{(c)}(t+1)\right)^{2}-\left(U_{i}^{(c)}(t)\right)^{2}\leq B_{i,max}^{(c)}(t)-2U_{i}^{(c)}(t)\cdot\\\left(\sum_{b}\mu_{ib}^{(c)}(t)-\sum_{a}\mu_{ai}^{(c)}(t)\right)+2U_{i}^{(c)}(t)A_{i}^{(c)}(t).\end{aligned}\tag{19}$$

Next we multiply Eq. (19) by $\theta_{i}^{(c)}(t)$ to create the potential $\Phi_{i}^{(c)}(t)$ at the right hand side of the inequality and obtain

$$\begin{aligned}\theta_{i}^{(c)}(t)\cdot\left[\left(U_{i}^{(c)}(t+1)\right)^{2}-\left(U_{i}^{(c)}(t)\right)^{2}\right]\leq\\\theta_{i}^{(c)}(t)\cdot B_{i,max}^{(c)}(t)-\\2\cdot\underbrace{\theta_{i}^{(c)}(t)\cdot U_{i}^{(c)}(t)}_{\Phi_{i}^{(c)}(t)}\left(\sum_{b}\mu_{ib}^{(c)}(t)-\sum_{a}\mu_{ai}^{(c)}(t)\right)+\\2\cdot\underbrace{\theta_{i}^{(c)}(t)\cdot U_{i}^{(c)}(t)}_{\Phi_{i}^{(c)}(t)}\cdot A_{i}^{(c)}(t).\end{aligned}\tag{20}$$

In order to maintain the telescoping series in the left hand side of the equation, when summing over time slots, we rewrite the left hand side of the equation as follows:

$$\begin{aligned}\theta_{i}^{(c)}(t)\cdot\left[\left(U_{i}^{(c)}(t+1)\right)^{2}-\left(U_{i}^{(c)}(t)\right)^{2}\right]=\\\theta_{i}^{(c)}(t+1)\cdot\left(U_{i}^{(c)}(t+1)\right)^{2}-\theta_{i}^{(c)}(t)\cdot\left(U_{i}^{(c)}(t)\right)^{2}\\+\left(\underbrace{\theta_{i}^{(c)}(t)-\theta_{i}^{(c)}(t+1)}_{-\Delta\theta_{i}^{(c)}(t+1)}\right)\cdot\left(U_{i}^{(c)}(t+1)\right)^{2}.\end{aligned}\tag{21}$$

Combining Eq. (20) with Eq. (21) yields the result. ∎

With this Lemma at hand, we can proceed to establish a sufficient condition on the *priority change regulation function* that ensures the stability of *U-BP* within the entire capacity region of the network. Specifically, we next seek a condition that will guide us in determining how to gradually adapt $\theta_{i}^{(c)}(t)$ when the user-defined priority $\tilde{\theta}_{i}^{(c)}(t)$ changes, such that throughput optimality is maintained.

B. Stability proof using Lyapunov technique

We observe that according to Lemma 1, providing an appropriate upper bound on the regulation function $P_{i}^{(c)}(t+1)$ can ensure the desired negative drift of the Lyapunov function $\theta_{i}^{(c)}(t)\cdot\left(U_{i}^{(c)}(t)\right)^{2}$ and therefore the throughput optimality. Therefore, we will want to use a gradual priority change such that this upper bound is maintained. Using Lemma 1 and this observation, we prove the following result.

Theorem 1. *Let λ be strictly admissible. Assume*

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}[P_{i}^{(c)}(\tau+1)]<\infty.$$

Then:

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}\left(\Phi_{i}^{(c)}(\tau)\right)<\infty.$$

Proof: Summing the result obtained by Lemma 1 over all (i,c) entries, rearranging and taking conditional expectation yields:

$$\begin{aligned}\mathbb{E}\left[\sum_{ic}\theta_{i}^{(c)}(t+1)\cdot\left(U_{i}^{(c)}(t+1)\right)^{2}|\mathbf{\Phi}(t)\right]-\\\mathbb{E}\left[\sum_{ic}\theta_{i}^{(c)}(t)\cdot\left(U_{i}^{(c)}(t)\right)^{2}|\mathbf{\Phi}(t)\right]\leq\\\mathbb{E}\left[\sum_{ic}\theta_{i}^{(c)}(t)\cdot B_{i,max}^{(c)}(t)|\mathbf{\Phi}(t)\right]+\\\mathbb{E}\left[\sum_{ic}P_{i}^{(c)}(t+1)|\mathbf{\Phi}(t)\right]+\\2\sum_{ic}\Phi_{i}^{(c)}(t)\cdot\mathbb{E}\left[A_{i}^{(c)}(t)|\mathbf{\Phi}(t)\right]-2\mathbb{E}\\\left[\sum_{ic}\Phi_{i}^{(c)}(t)\left(\sum_{b}\mu_{ib}^{(c)}(t)-\sum_{a}\mu_{ai}^{(c)}(t)\right)|\mathbf{\Phi}(t)\right].\end{aligned}\tag{22}$$

Applying Eq. (1)-(3) and Eq. (5)-(10) on Eq. (22) yields:

$$\begin{aligned}\mathbb{E}\left[\sum_{ic}\theta_{i}^{(c)}(t+1)\cdot\left(U_{i}^{(c)}(t+1)\right)^{2}|\mathbf{\Phi}(t)\right]-\\\mathbb{E}\left[\sum_{ic}\theta_{i}^{(c)}(t)\cdot\left(U_{i}^{(c)}(t)\right)^{2}|\mathbf{\Phi}(t)\right]\leq\\\theta_{\max}\cdot B-2\varepsilon\cdot\sum_{ic}\Phi_{i}^{(c)}(t)+\\\mathbb{E}\left[\sum_{ic}P_{i}^{(c)}(t+1)|\mathbf{\Phi}(t)\right],\end{aligned}\tag{23}$$

where:

$$\begin{aligned}B=\sum_{ic}B_{i,\max}^{(c)}=\\\sum_{ic}\left(\sum_{b}\mu_{ib,\max}^{(c)}\right)^{2}+\sum_{ic}\left(\sum_{a}\mu_{ai,\max}^{(c)}+A_{i,\max}^{(c)}\right)^{2}.\end{aligned}\tag{24}$$

Taking expectations on Eq. (23) and summing over time slots yields:

$$\begin{aligned}\mathbb{E}[\sum_{ic}\theta_{i}^{(c)}(t)\cdot\left(U_{i}^{(c)}(t)\right)^{2}-\sum_{ic}\theta_{i}^{(c)}(0)\cdot\left(U_{i}^{(c)}(0)\right)^{2}]\\\leq t\cdot\theta_{\max}\cdot B-2\varepsilon\cdot\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}[\Phi_{i}^{(c)}(\tau)]\\+\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}[P_{i}^{(c)}(\tau+1)]\end{aligned}\tag{25}$$

Rearranging, dividing by t and taking limits of Eq. (25) yields:

$$\begin{aligned}&\limsup_{t\to\infty}\tfrac{1}{t}\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}\left(\Phi_i^{(c)}(\tau)\right)\leq\tfrac{\theta_{\max}B}{2\varepsilon}\\&+\tfrac{1}{2\varepsilon}\limsup_{t\to\infty}\tfrac{1}{t}\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}[P_i^{(c)}(\tau+1)]<\infty,\end{aligned}\tag{26}$$

yielding the result. ■

As a consequence, by applying Eq. (16), we further obtain:

Corollary 1 (*U-BP* stability)**.** *Let λ be strictly admissible and*

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}[P_i^{(c)}(\tau+1)]<\infty.$$

Then:

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{ic}\mathbb{E}[U_i^{(c)}(\tau)]<\infty,$$

i.e., U-BP *achieves stability within the entire capacity region.*

Therefore, any choice of $\theta_i^{(c)}(t)$ values that respect the assumption of Corollary 1 on the regulation function will also ensure stability. Out of all such values, we would like to pick those that would converge the fastest to our reference priority and maintain overall network performance. Accordingly, we next proceed to bounding the priority change regulation function and assess the theoretical impact of our choice on network performance.

C. Bounding the regulation function

Our goal is to maximize the convergence rate of the weighted-average *effective priority* of each queue (i,c) to its user-defined *reference priority*. To that end, for each queue (i,c), we define a positive constant $P_i^{(c)}$ and propose to gradually change the priority of queue (i,c) such that its corresponding value of $P_i^{(c)}(t)$ will keep obeying the following bound:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}P_i^{(c)}(\tau+1)\leq P_i^{(c)}<\infty\ \ \forall t.\tag{27}$$

The idea behind this choice is that, since queue regulation is independent among queues, it allows a completely distributed implementation in networks with independent links (*i.e.,* when the interference between different links is negligible). Moreover, such a heuristic does not incur additional computational overhead even when links are dependent.

In order to derive sufficient values for the priorities within the potential function, we analyze Eq. (26) and apply Eq. (27). Specifically, we optimize these values with respect to the upper bound on mean queue lengths and obtain the following result:

Proposition 1 (Setting priorities)**.** *Assume that* U-BP *sets*

$$P_i^{(c)}=\theta_{\max}\cdot B_{i,max}^{(c)}.$$

Then, $P_i^{(c)}$ maximizes the convergence rate of a queue to its desired priority with respect to the upper bound of the mean queue length.

Proof: Consider Eq. (26). Then:

$$\Phi_i^{(c)}\leq\frac{\theta_{\max}B_{i,max}^{(c)}+P_i^{(c)}}{2\varepsilon},\tag{28}$$

and

$$U_i^{(c)}\leq\frac{\theta_{\max}B_{i,max}^{(c)}+P_i^{(c)}}{2\varepsilon\cdot\theta_{\min}}.\tag{29}$$

Our goal is to maximize $\Delta\theta_i^{(c)}$. To do so, we construct the following optimization problem that seeks the optimal value of $P_i^{(c)}$ with respect to the upper bound on the mean queue length:

$$\begin{aligned}&\underset{P_i^{(c)}}{\text{maximize}}:\Delta\theta_i^{(c)}\\&s.t.\\&(a)\,U_i^{(c)}=\tfrac{\theta_{\max}\cdot B_{i,max}^{(c)}+P_i^{(c)}}{2\varepsilon\cdot\theta_{\min}}\\&(b)\,P_i^{(c)}=\Delta\theta_i^{(c)}\cdot\left(U_i^{(c)}\right)^2.\end{aligned}\tag{30}$$

Applying (a) on (b) and deriving with respect to $P_i^{(c)}$ yields:

$$P_i^{(c)}=\theta_{\max}\cdot B_{i,max}^{(c)},\tag{31}$$

thus establishing the result. ■

D. Constructing the potential function

Next, using this result, we can finally construct potential function $\Phi_i^{(c)}(t)$ and formally define how we update the effective priority. Applying Theorem 1 on the definition of the priority change regulation function, we obtain:

$$\theta_{\max}\cdot B_{i,max}^{(c)}=\Delta\theta_i^{(c)}\cdot\left(U_i^{(c)}\right)^2,\tag{32}$$

i.e.,

$$\Delta\theta_i^{(c)}=\frac{\theta_{\max}\cdot B_{i,max}^{(c)}}{\left(U_i^{(c)}\right)^2}.\tag{33}$$

Additionally, we observe that when a priority of a queue is *decreased*, the regulation function becomes *negative*. Using these observations, we formulate the potential function as follows:

$$\Phi_i^{(c)}(t)=\theta_i^{(c)}(t)\cdot U_i^{(c)}(t),\tag{34}$$

where we can finally define the effective priority:

$$\begin{aligned}&\theta_i^{(c)}(t)=\\&\min\left\{\tilde{\theta}_i^{(c)}(t),\theta_i^{(c)}(t-1)\cdot\left(1+\tfrac{B_{i,max}^{(c)}(t-1)}{\left(U_i^{(c)}(t-1)\right)^2}\right)\right\},\end{aligned}\tag{35}$$

with

$$B_{i,\max}^{(c)}(t)=\max_{\tau\in[0,t]}\{B_{i,\max}^{(c)}(\tau)\}.\tag{36}$$

We finally defined the effective priority $\theta_i^{(c)}(t)$. Note that since $B_{i,max}^{(c)}$ might be hard to estimate, Eq. (36) achieves it by carrying the largest value seen so far[1].

[1]Carrying the largest value of $B_{i,max}^{(c)}$ is the result that is obtained from an analytical point of view. However, in networks where on a rare occasion these values might be abnormally large, one can regulate them by disregarding abnormally large values, and still obtain network stability.

Next, by applying Theorem 1 to Lemma 1, we obtain:

Corollary 2. *Let λ denote a strictly admissible arrival rate vector. Assume that the potential function of each queue respects Eq. (34), Eq. (35) and Eq. (36). Then:*

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{ic}\mathrm{E}\left(\Phi_i^{(c)}(\tau)\right)\leq\frac{\theta_{\max}B}{\varepsilon}.$$

This result immediately follows from Eq. (26). Again, since for all t and (i,c) it holds that $\theta_i^{(c)}(t)>0$, we obtain *stability within the entire capacity region of the network*. Interestingly, this is exactly twice the standard upper bound that is obtained without allowing dynamic user-defined priority service [7].

E. Convergence rate to the reference priorities

The convergence of the potential function upon a priority decrease is immediate, as dictated by Eq. (35). However, upon a priority increase, the convergence rate is bounded by the congestion experienced by the queue. The intuition for this asymmetry lies in Eq. (9) that specifies the resource allocation process employed by the backpressure algorithm at each time slot. Specifically, an unregulated increase of a queue priority may cause a suboptimal resource allocation choice from a throughput perspective, since it may force the algorithm to serve high-priority queues with little consideration for the better links offered at that time slot. Note that when a queue is more congested, then even a small increase in its priority may be enough to serve it over other less congested queues with better available links. On the contrary, when a queue's priority is decreased, the algorithm can more efficiently take advantage of the better links.

We proceed to approximate the convergence time of a queue potential upon a priority increase. We consider a scenario in which at time slot t, the weighted average priority of a queue increases from θ_l to θ_h and remains fixed until it converges. From Eq. (35), considering a hypothetical state of a fixed queue length, we obtain:

$$\theta_{i,h}^{(c)}=\theta_{i,l}^{(c)}\cdot\left(1+\frac{B_{i,max}^{(c)}}{\left(U_i^{(c)}\right)^2}\right)^T, \tag{37}$$

where T is the number of time slots required for the queue potential to converge. Taking the logarithm of both sides of Eq. (37) and rearranging yields:

$$T=\frac{\log\left(\frac{\theta_{i,h}^{(c)}}{\theta_{i,l}^{(c)}}\right)}{\log\left(1+\frac{B_{i,max}^{(c)}}{\left(U_i^{(c)}\right)^2}\right)}. \tag{38}$$

From Eq. (38), by considering the Taylor expansion of the logarithm function and assuming a congested network such that $\frac{B_{i,max}^{(c)}}{\left(U_i^{(c)}\right)^2}<<1$, we obtain that the convergence is slightly slower:

$$T\propto\frac{\left(U_i^{(c)}\right)^2}{B_{i,max}^{(c)}}.$$

Conversely, when the network is not highly loaded, T is proportional to $\log\left(\frac{\theta_{i,h}^{(c)}}{\theta_{i,l}^{(c)}}\right)$, thus the potential converges within a few time slots, and at most $O(\log(\alpha_m/\alpha_1))$.

V. Evaluations

We now run simulations to evaluate the relative performance of three algorithms with user-defined priorities: (a) the standard *Q-BP* backpressure algorithm, (b) the *m-DRPC* algorithm that combines the DRPC priority-service algorithm for fixed queue priorities with our m priority-based queues, and (c) our suggested *U-BP* algorithm.

A. User-defined priorities over a shared link

This test is partially based on the counter-example provided by [5] to their priority service algorithm. Specifically, we test a scenario with two senders sending data to a single receiver. We use a strict interference model such that only a single link out of the two can be activated at the same time slot. The capacity of each link is i.i.d between time slots, and takes values within $\{1,2,3\}$. We randomly inject high-priority and low-priority flows to both sources, such that the high-priority flows consist of 50% of the total traffic considering both marking and remarking of flow priorities. The times and durations of all flows are randomly chosen under the constraint that their sum is upper-bounded by a Poisson process with $\lambda=0.9$ for each source to correspond with our model. The priority weights are set to 1 and 2 for low- and high-priority flows, respectively.

Figure 2 depicts the evaluation results. Figures 2(a) and 2(b) show the end-to-end delay of packets using *U-BP* and *Q-BP* for high-priority and low-priority flows, respectively. As expected, the end-to-end delay of our *U-BP* algorithm for high-priority flows is significantly lower, while the overall network performance is roughly the same for both algorithms.

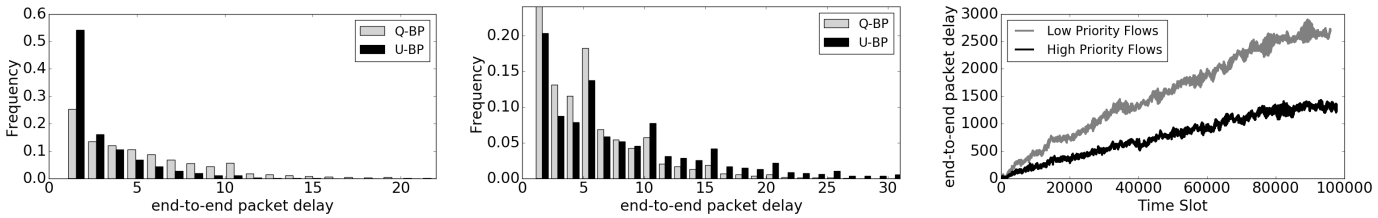
In addition, Figure 2(c) illustrates how *m-DRPC* cannot stabilize the network due to the unregulated changes of flow priorities . This happens even though the load is strictly within the capacity region of the network.

B. User-defined priorities over a random dynamic network

We create a random network with 16 nodes based on the scale-free model of [8]. Next, among these 16 nodes, we randomly choose four nodes as sources and four nodes as destinations, *i.e.,* create up to 16 potential flows. We fix the routes among the sources and the destinations and make sure each route shares at least one link with at least one other route. All link capacities are i.i.d. and take values within $\{1,2,3\}$.

Among these sources and destinations, we randomly create high- and low-priority flows such that the high-priority flows use 15% of the traffic. We repeated this experiment 100 times, and consistently obtained similar results in non-trivial topologies. Since different randomly generated networks produce routes with different lengths and different loads on bottleneck links, we choose a representative experiment.

Figure 3 presents the simulation results. First, Figure 3(a) shows how the end-to-end delay of high-priority flows in *U-BP* is significantly lower than in both *Q-BP* and *m-DRPC*. In fact,

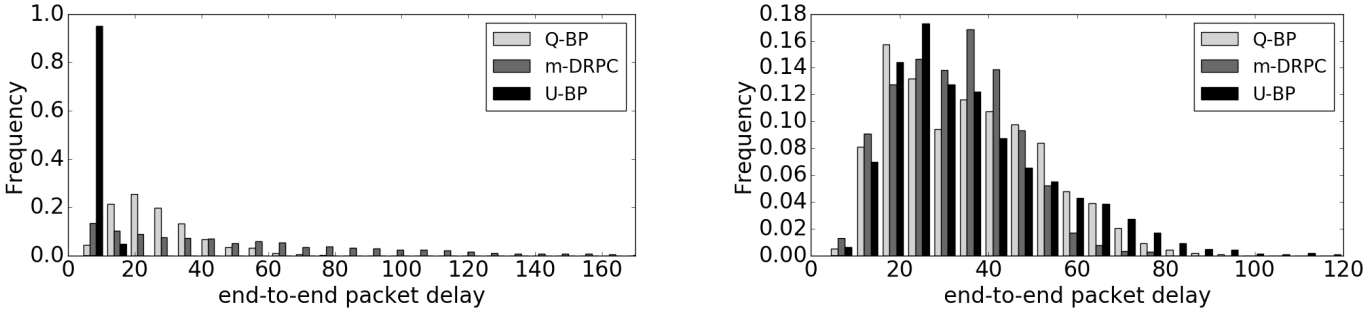


(a) High-priority flows with *Q-BP* and *U-BP*. The mean delays are 5.354 and 3.004 time slots, respectively.

(b) Low-priority flows with *Q-BP* and *U-BP*. The mean delays are 5.615 and 8.107 time slots, respectively.

(c) The *m-DRPC* algorithm leads to an unstable network with increasing delays as a function of time.

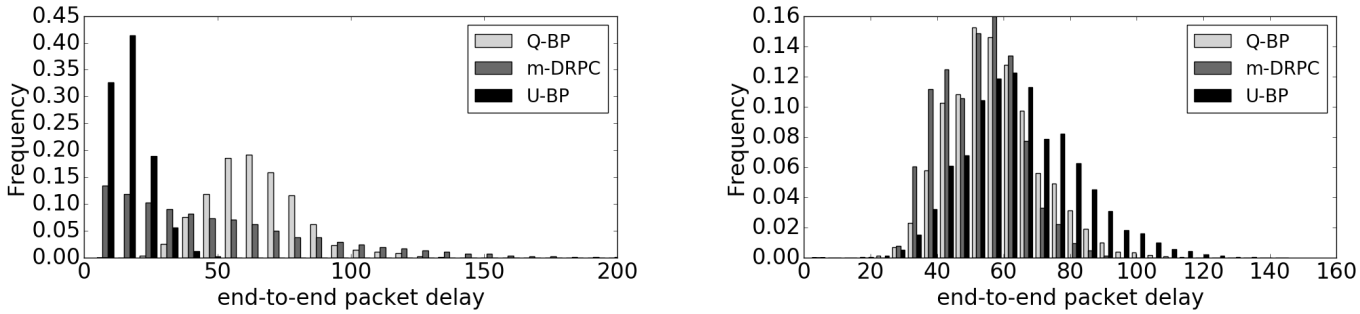
Fig. 2. Performance comparison of *Q-BP*, *m-DRPC* and our *U-BP* when two sources send data to a single receiver under a strict interference model (*i.e.,* only one of the links can be activated at each time slot).



(a) High-priority flows. The average delay in time slots is 27.54 for *Q-BP*, 53.4 for *m-DRPC* and 7.18 for *U-BP*.

(b) Low-priority flows. The average delay in time slots is 35.87 for *Q-BP*, 34.23 for *m-DRPC* and 36.07 for *U-BP*.

Fig. 3. Performance comparison of *Q-BP*, *m-DRPC* and our *U-BP* in a random network topology with 16 nodes, time-varying link capacities, and random traffic with 15% of high-priority packets.



(a) High-priority flows. The average delay in time slots is 64.4 for *Q-BP*, 65.7 for *m-DRPC* and 15.7 for *U-BP*.

(b) Low-priority flows. The average delay in time slots is 57.33 for *Q-BP*, 54.48 for *m-DRPC* and 62.87 for *U-BP*.

Fig. 4. Performance comparison in a random network with a strict interference model, such that each node can transmit or receive data using a single link at each time slot.

due to the moderate load, *m-DRPC* maintains stability in this specific scenario (in our experiments we have seen similar scenarios where *m-DRPC* could not stabilize the network), but still presents a long-tailed distribution. This follows from the fact that the high-priority flows only compose 15% of the traffic, and thus suffer from long delays because of the high number of low-priority flows, even though they have a higher priority. Figure 3(b) shows that the overall performance is maintained for *U-BP*. Namely, the low-priority flow delays are only slightly increased to compensate for the better delays of the high-priority flows.

C. User-defined priorities over a random dynamic network with a strict interference model

We repeat the above experiment with the following changes: (a) The capacities of the links can take values within $\{1, \ldots, 10\}$; and (b) we use a strict interference model, *i.e.,* a node is allowed to transmit and receive data only through a single available link at each time slot.

Figure 4(a) shows again that the delays of the high-priority flows are much lower for *U-BP*. As a tradeoff, Figure 4(b) shows that the delays of the low-priority flows in *U-BP* are slightly larger than in other algorithms. Again, the delays of the low-priority flows in *m-DRPC* are lower, but at the expense of the significantly-long tail delay of the high-priority flows.

VI. Related work

Backpressure. *Q-BP* was introduced in the seminal work of [6] and received much attention in the literature due to its ability to stabilize a dynamic network within its entire capacity region. Numerous extensions to this technique

have been proposed e.g., [6], [7], [9]–[29]. These works include delay-reduction techniques [9], [11], [12], the drift-plus-penalty method that stabilizes a network while also minimizing the time average of a network penalty function [7], [10], [14], practical wireless protocols based on the backpressure idea [15]–[17], and many more. Recent works also focus on incorporating delay considerations into the backpressure queue potentials [30]–[32]. *U-BP* is an orthogonal tool to these approaches, and can potentially be combined with them to provide user-defined priority capabilities.

User-defined priorities. Several works have investigated whether some applications can receive special treatment from the network without conflicting with network neutrality. A recent paper [1] has shown that users do want some applications to receive a preferential treatment, and that it is practical for users to temporarily or permanently define high preferences for some applications. In addition, [2], [3] expose an API for applications and users to express their preferences, for example by leveraging software-defined networking capabilities. This recent research has already been applied in the industry. OnHub, a commercial home WiFi router built by Google, enables user-defined prioritization of certain devices and applications [4].

VII. Conclusions

In this paper we tackled the challenge of providing user-defined flow-based prioritization while preserving throughput optimality and overall network performance in term of end-to-end delay. We introduced *U-BP* (User-defined BackPressure), and analytically proved that it achieves optimal throughput. Using both the dual-layer approach and the gradual priority update, we have shown through a set of simulations that *U-BP* indeed efficiently supports user-defined flow-based prioritization while maintaining overall network performance.

VIII. Acknowledgments

The authors would like to thank Yiannis Yiakoumis for useful discussions. 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] Y. Yiakoumis, S. Katti, and N. McKeown, "Neutral net neutrality," *ACM SIGCOMM*, 2016.

[2] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: An API for application control of SDNs," in *ACM SIGCOMM Computer Communication Review*, 2013.

[3] Y. Yiakoumis, S. Katti, T.-Y. Huang, N. McKeown, K.-K. Yap, and R. Johari, "Putting home users in charge of their network," in *Proceedings of the ACM Conference on Ubiquitous Computing*, 2012.

[4] "Google OnHub," https://on.google.com/hub/.

[5] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE Journal on Selected Areas in Communications*, 2005.

[6] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. on Automatic Control*, 1992.

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

[8] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, 2002.

[9] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *IEEE Infocom*, 2009.

[10] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," Ph.D. dissertation, MIT, 2003.

[11] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "LIFO-backpressure achieves near-optimal utility-delay tradeoff," *IEEE/ACM ToN*, 2013.

[12] L. Huang and M. J. Neely, "Delay reduction via lagrange multipliers in stochastic network optimization," *IEEE Trans. on Aut. Control*, 2011.

[13] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inform. Theory*, 1993.

[14] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM ToN*, 2008.

[15] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "Diffq: Practical differential backlog congestion control for wireless networks," in *IEEE Infocom*, 2009.

[16] A. Sridharan, S. Moeller, B. Krishnamachari, and M. Hsieh, "Implementing backpressure-based rate control in wireless networks," in *Information Theory and Applications Workshop*, 2009.

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

[18] M. J. Neely, "Delay-based network utility maximization," *IEEE/ACM ToN*, 2013.

[19] B. Sadiq and G. De Veciana, "Throughput optimality of delay-driven maxweight scheduler for a wireless system with flow dynamics," in *Allerton*, 2009.

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

[21] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM ToN*, 2011.

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

[23] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. 43rd CDC, IEEE*, 2004.

[24] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, 2005.

[25] E. Yeh and R. Berry, "Throughput optimal control of wireless networks with two-hop cooperative relaying," in *IEEE Trans. Inform. Theory*, 2007.

[26] X. Wu, R. Srikant, and J. R. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE TMC*, 2007.

[27] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," *IEEE/ACM ToN*, 2009.

[28] J. Ni and R. Srikant, "Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks," in *Information Theory and Applications Workshop*, 2009.

[29] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM ToN*, 2010.

[30] S. Li, E. Ekici, and N. Shroff, "Throughput-optimal queue length based csma/ca algorithm for cognitive radio networks," *IEEE Transactions on Mobile Computing*, 2015.

[31] 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 ToN*, 2016.

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