Receiver-Based Flow Control for Networks in Overload

Chih-ping Li and Eytan Modiano, *Fellow, IEEE*

***Abstract*—We consider utility maximization in networks where the sources do not employ flow control and may consequently overload the network. In the absence of flow control at the sources, some packets will inevitably have to be dropped when the network is in overload. To that end, we first develop a distributed, threshold-based packet-dropping policy that maximizes the weighted sum throughput. Next, we consider utility maximization and develop a receiver-based flow control scheme that, when combined with threshold-based packet dropping, achieves the optimal utility. The flow control scheme creates virtual queues at the receivers as a *push-back* mechanism to optimize the amount of data delivered to the destinations via back-pressure routing. A new feature of our scheme is that a utility function can be assigned to a collection of flows, generalizing the traditional approach of optimizing per-flow utilities. Our control policies use finite-buffer queues and are independent of arrival statistics. Their near-optimal performance is proved and further supported by simulation results.**

***Index Terms*—Finite-buffer networks, flow control, network overload, queueing analysis, robust control, stochastic networks, utility maximization.**

I. Introduction

FLOW control in data networks aims to provide fair allocation of resources and regulate the source rates of traffic flows in order to prevent network overload. In recent years, network utility maximization problems have been studied to optimize network performance through a combination of flow control, routing, and scheduling, whose optimal operations are revealed as the solution to the utility maximization problems (e.g., see [1]–[9]). Most studies in network flow control focus on source-based algorithms that require all sources to react properly to congestion signals such as packet loss or delay. However, in the presence of a greedy or malicious source that injects excessive traffic into the network, the throughput of other data flows may be adversely affected or even starved. In such scenarios, source-based flow control may be ineffective. We are thus motivated to study optimal control policies in overloaded networks where the sources are uncooperative.

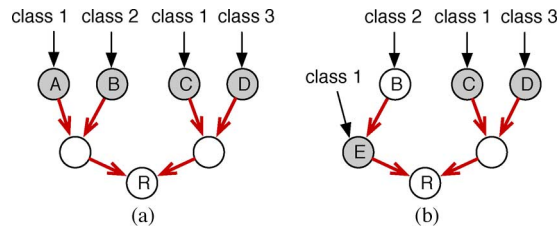


Fig. 1. (a) Tree network with three classes of traffic. When the network is overloaded by exogenous traffic, nodes (A, B, C, D) form a defense perimeter and perform rate throttling by dropping packets to optimize network performance. (b) Resulting network topology when node A in Fig. 1(a) is misconfigured so that it forwards everything it receives. The control policy developed in this paper performs packet dropping at all intermediate nodes and seamlessly establishes a new defense perimeter (E, C, D) without any changes in network operations.

We consider the problem of maximizing throughput utilities in a network, assuming that all traffic flows do not employ flow control and may overload the network. Flows are categorized into classes so that flows in a class have a shared destination. A class may simply be a flow specified by a source–destination pair, or corresponds to a subset of flows that communicate with a common Web site. A utility function is assigned to each traffic class, and the sum of the class-based utilities is maximized as a means to control the aggregate throughput of flows in each class. The use of class-based utility functions is partly motivated by the need of mitigating network congestion caused by a collection of data flows whose aggregate throughput needs to be controlled [13]. Without flow control at the sources, some packets will be dropped when the network is overloaded. To provide differentiated services to multiple traffic classes, we consider the scenario where the destinations can perform flow control to regulate the received throughput of each traffic class. The question we seek to answer is how to design in-network packet dropping and receiver-based flow control strategies to maximize the sum of class-based utilities and stabilize the network.

In-network packet dropping and receiver-based flow control enhance the robustness of network operations. For example, consider the tree network in Fig. 1(a) that serves three classes of traffic. When the network is overloaded, an optimal packet-dropping policy implemented at all network nodes guarantees that the receiver R is protected from excessive traffic, and the throughput of all traffic classes is optimized via receiver-end flow control. Now, suppose that node A in Fig. 1(a) is misconfigured and forwards everything it receives; effectively, node A becomes a greedy user. In this scenario,

Manuscript received March 18, 2013; revised September 10, 2013 and January 16, 2014; accepted January 19, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor U. Ayesta. Date of publication February 19, 2014; date of current version April 14, 2015. This work was supported by the DTRA under Grants HDTRA1-07-1-0004 and HDTRA-09-1-005, the ARO MURI under Grant No. W911NF-08-1-0238, and the NSF under Grant CNS-1116209. This work was presented in part at the IEEE International Conference on Computer Communications (INFOCOM), Turin, Italy, April 14–19, 2013.

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: cpli@mit.edu; modiano@mit.edu).

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

Digital Object Identifier 10.1109/TNET.2014.2302445

since all nodes perform packet dropping, we have a new controlled network domain in Fig. 1(b) in which the receiver R remains protected and the throughput of all traffic classes is reoptimized without any changes in network operations. In contrast, under source-based flow control that relies on end-users or bordering nodes to regulate traffic, if a user becomes greedy or a bordering node is misconfigured, then all downstream nodes are overloaded.

In this paper, we develop such receiver-based flow control policies using tools from stochastic network optimization [14], [15]. Our main contributions are threefold. First, we formulate a utility maximization problem that assigns a utility function to a class of flows, of which the usual per-flow-based utility maximization is a special case. Second, given an arbitrary arrival rate matrix (possibly outside the network's stability region), we characterize the set of achievable throughput vectors in terms of queue overflow rates. Third, using a simple decomposition of the utility functions, we design a network control policy consisting of: 1) a set of flow controllers at the receivers; 2) a packet-dropping mechanism at internal nodes; and 3) back-pressure routing between intermediate nodes. The receiver-based flow controllers adjust throughput by modifying the differential backlogs between the receivers and their neighboring nodes—a small (or negative) differential backlog is regarded as a *push-back* mechanism to slow down data delivery to the receivers. To deal with undeliverable data due to network overload, we design a threshold-based packet-dropping mechanism that discards data whenever queues grow beyond certain thresholds. The threshold-based packet-dropping policy, without the use of flow control, suffices to maximize the weighted sum throughput. Moreover, the combined flow control and packet-dropping mechanism has the following properties.

1) It is distributed and only requires information exchange between neighboring nodes.
2) It uses finite-size buffers.
3) It is nearly utility-optimal (throughput-optimal as a special case), and the performance gap from the optimal utility goes to zero as buffer sizes increase.
4) It does not need the knowledge of arrival rates and thus is robust to time-varying arrival rates that can go far beyond the network's stability region.
5) This policy works seamlessly without the need of explicitly deciding whether a network enters or leaves an overload period.
6) The policy can be implemented in parts of a network that include the receivers, treating the rest of the network as exogenous data sources (see Fig. 2).

There has been a significant amount of research in the general area of stochastic network control. Utility-optimal policies that combine source-end flow control with back-pressure routing have been studied in [6]–[9] (and references therein). These policies optimize per-flow utilities and require infinite-capacity buffers. However, they are not robust in the face of uncooperative users who may not adhere to the flow control scheme. A closely related problem to that studied in this paper is that of characterizing the queue overflow rates in networks under overload. In a single-commodity network, a back-pressure policy is shown to achieve the most balanced queue overflow rates [16], and controlling queue growth rates using the max-weight policy is discussed in [17]. The queue growth rates in networks under max-weight and α-fairness policies are analyzed in [18] and [19]. We finally note that the importance of controlling an aggregate of data flows has been addressed in [13], and rate-limiting mechanisms in front of a Web server to achieve some notion of max-min fairness against distributed denial-of-service (DDoS) attacks have been proposed in [20]–[22].

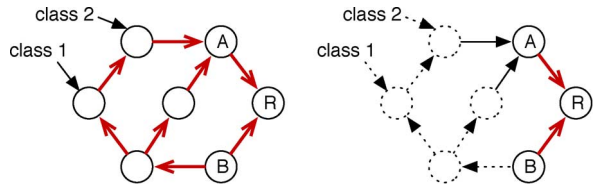


Fig. 2. Our receiver-based policy can be implemented in the whole network on the left, or implemented only at nodes A, B, and R on the right, where R is the only receiver. The rest of the network on the right may be controlled by another network operator or follow a different network control scheme.

The outline of the paper is as follows. The network model is given in Section II. We formulate the utility maximization problem and characterize the set of achievable throughput vectors in terms of queue overflow rates in Section III. Section IV introduces a threshold-based packet-dropping policy that maximizes the weighted sum throughput without the use of flow control. Section V presents a receiver-based flow control and packet-dropping policy that solves the utility maximization problems. Simulation results that demonstrate the near-optimal performance of our policies are given in Sections IV and V.

II. Network Model

We consider a network with nodes $\mathcal{N} = \{1, 2, \ldots, N\}$ and directed links $\mathcal{L} = \{(n, m) \mid n, m \in \mathcal{N}\}$. Assume time is slotted. In every slot, packets randomly arrive at the network and are categorized into a collection $\mathcal{C}$ of classes. The definition of a data class is flexible except that we assume packets in a class $c \in \mathcal{C}$ have the same destination σ_c. For example, each class can be a flow specified by a source–destination pair. Internet service providers may assign business users to one class and residential users to another. Media streaming applications may categorize users into classes according to different levels of subscription. While classification of users/flows in various contexts is a subject of significant importance, in this paper we assume for simplicity that the class to which a packet belongs can be ascertained from information contained in the packet (e.g., source/destination address, tag, priority field, etc.). Let $A_n^{(c)}(t) \in \{0, 1, \ldots, A_{\max}\}$ be the number of exogenous class c packets arriving at node n in slot t, where $A_{\max}$ is a finite constant. Let $A_{\sigma_c}^{(c)}(t) = 0$ for all t. We assume $A_n^{(c)}(t)$ are independent across classes c and nodes $n \neq \sigma_c$, and are i.i.d. over slots with mean $\mathbb{E}[A_n^{(c)}(t)] = \lambda_n^{(c)}$. We assume that the arrival rates $\lambda_n^{(c)}$ are unknown in the network.

Packets are relayed toward the destinations according to dynamic routing and link rate allocation decisions. For simplicity, we assume that there are no prespecified paths for each traffic class; our results can be easily generalized to the case where each class is given a collection of paths in advance. Each link

$l = (n, m) \in \mathcal{L}$ transmits data from node n to node m and has a fixed capacity $\mu_l^{\max}$ (in units of packets/slot).[1] Under a given control policy, let $\mu_l^{(c)}(t)$ be the service rate allocated to class c data on link l in slot t. The service rates must satisfy the link capacity constraints $\sum_{c\in\mathcal{C}} \mu_l^{(c)}(t) \le \mu_l^{\max}$ for all t and all links l. Class c packets that arrive at a node $n \neq \sigma_c$ but are yet forwarded are stored in a queue $Q_n^{(c)}(t)$. Let $Q_n^{(c)}(t)$ be the queue backlog of class c packets at node n at time t[2]; assume initially $Q_n^{(c)}(0) = 0$ for all c and n. Destinations do not buffer packets, and we have $Q_{\sigma_c}^{(c)}(t) = 0$ for all c and t. For now, we assume every queue $Q_n^{(c)}(t)$ has an infinite-capacity buffer; we show later that our control policy needs only finite-size buffers. To resolve potential network congestion due to traffic overload, a queue $Q_n^{(c)}(t)$, after forwarding data to neighboring nodes in slot t, discards $d_n^{(c)}(t)$ packets from the remaining backlog at the end of the slot. Packets discarded from the queue $Q_n^{(c)}(t)$ are removed from the network immediately. The drop rate $d_n^{(c)}(t)$ depends on the control policy and takes values in $[0, d_{\max}]$ for some finite $d_{\max}$. Let $L_{\text{in}}(n)$ be the set of incoming links of node n, and $L_{\text{out}}(n)$ be the set of outgoing links of node n. The queue backlog process $Q_n^{(c)}(t)$ evolves over slots according to

$$Q_n^{(c)}(t+1) \le \left[\left(Q_n^{(c)}(t) - \sum_{l\in L_{\text{out}}(n)} \mu_l^{(c)}(t)\right)^+ - d_n^{(c)}(t)\right]^+ + A_n^{(c)}(t) + \sum_{l\in L_{\text{in}}(n)} \mu_l^{(c)}(t) \qquad \forall c,\ n \neq \sigma_c \quad (1)$$

where $(\cdot)^+ \triangleq \max(\cdot, 0)$. The inequality in (1) is due to the fact that endogenous arrivals may be strictly less than the allocated incoming link rates $\sum_{l\in L_{\text{in}}(n)} \mu_l^{(c)}(t)$ when neighboring nodes do not have sufficient packets to send.

For convenience, we define the maximum transmission rate into and out of a node by

$$\mu_{\max}^{\text{in}} \triangleq \max_{n\in\mathcal{N}} \sum_{l\in L_{\text{in}}(n)} \mu_l^{\max} \quad \mu_{\max}^{\text{out}} \triangleq \max_{n\in\mathcal{N}} \sum_{l\in L_{\text{out}}(n)} \mu_l^{\max}.$$

Throughout the paper, we use the following assumption.

Assumption 1: We assume $d_{\max} \ge A_{\max} + \mu_{\max}^{\text{in}}$.

Equation (1) shows that $A_{\max} + \mu_{\max}^{\text{in}}$ is the largest amount of data that can arrive at a node in a slot and is an upper bound on the maximum queue overflow rate at any node. Assumption 1 ensures that the maximum packet-dropping rate $d_{\max}$ is no less than the maximum queue overflow rate, so that we can always choose packet-dropping rates to stabilize the network.

III. Problem Formulation

We assign to each class $c \in \mathcal{C}$ a utility function g_c. Given an (unknown) arrival rate matrix $\boldsymbol{\lambda} = (\lambda_n^{(c)})$, let $\Lambda_{\boldsymbol{\lambda}}$ be the set of all achievable throughput vectors $(r_c)_{c\in\mathcal{C}}$, where r_c is the aggregate throughput of class c data received by the destination σ_c. Note that $\Lambda_{\boldsymbol{\lambda}}$ is a function of $\boldsymbol{\lambda}$. We seek to design a control policy that solves the global utility maximization problem

$$\text{maximize} \quad \sum_{c\in\mathcal{C}} g_c(r_c) \quad (2)$$

$$\text{subject to} \quad (r_c) \in \Lambda_{\boldsymbol{\lambda}} \quad (3)$$

where the region $\Lambda_{\boldsymbol{\lambda}}$ is presented later in Lemma 1. We assume all functions g_c are concave, increasing, continuously differentiable, and have bounded derivatives with $g_c'(x) \le g_c'(0) < \infty$ for all $x \ge 0$ and all classes $c \in \mathcal{C}$.

A. Simple Example

Consider the tree network in Fig. 1(a) that serves three classes of traffic destined for node R. Class 1 data originates from two different sources A and C and may represent the collection of users located in different parts of the network communicating with R. If class-1 traffic is congestion-insensitive and overloads the network, without proper flow control, classes 2 and 3 will be starved due to the presence of class 1. The class-based utility maximization over the tree network is

$$\text{maximize} \quad g_1(r_{1A} + r_{1C}) + g_2(r_{2B}) + g_3(r_{3D}) \quad (4)$$

$$\text{subject to} \quad (r_{1A}, r_{1C}, r_{2B}, r_{3D}) \text{ feasible} \quad (5)$$

where r_{1A} denotes the throughput of class-1 data originating from A, and r_{1C}, r_{2B}, and r_{3D} are defined similarly. This utility maximization (4)–(5) provides fair resource allocation over aggregates of flows.

We remark that the class-based utility maximization problem (4)–(5) [or (2)–(3) in the general form] can potentially be solved by source-based admission control policies. However, such policies would require coordinations among the sources to optimize their aggregate admitted data rate and are infeasible in large-scale networks. In this paper, we develop a fully distributed control policy that solves (2)–(3) with information exchange only between neighboring nodes.

B. Achievable Throughput

The next lemma characterizes the set $\Lambda_{\boldsymbol{\lambda}}$ of all achievable throughput vectors in (3).

Lemma 1: Under i.i.d. arrival processes with an arrival rate matrix $\boldsymbol{\lambda} = (\lambda_n^{(c)})$, let $\Lambda_{\boldsymbol{\lambda}}$ be the closure of the set of all achievable throughput vectors $(r_c)_{c\in\mathcal{C}}$. Then, $(r_c)_{c\in\mathcal{C}} \in \Lambda_{\boldsymbol{\lambda}}$ if and only if there exist flow variables $\{f_l^{(c)} \ge 0 \mid c \in \mathcal{C}, l \in \mathcal{L}\}$ and queue overflow variables $\{q_n^{(c)} \ge 0 \mid c \in \mathcal{C},\ n \neq \sigma_c\}$ such that

$$\lambda_n^{(c)} + \sum_{l\in L_{\text{in}}(n)} f_l^{(c)} = q_n^{(c)} + \sum_{l\in L_{\text{out}}(n)} f_l^{(c)} \qquad \forall c, n \neq \sigma_c \quad (6)$$

$$\sum_c f_l^{(c)} \le \mu_l^{\max}, \qquad l \in \mathcal{L} \quad (7)$$

$$r_c \le \sum_{l\in L_{\text{in}}(\sigma_c)} f_l^{(c)} = \sum_{n\neq\sigma_c} \lambda_n^{(c)} - \sum_{n\neq\sigma_c} q_n^{(c)}, \qquad c \in \mathcal{C}. \quad (8)$$

In other words

$$\Lambda_{\boldsymbol{\lambda}} = \left\{(r_c) \mid (6)(7)(8) \text{ hold with } f_l^{(c)} \ge 0,\ q_n^{(c)} \ge 0\right\}.$$

[1]We focus on wireline networks in this paper for the ease of exposition. Our results and analysis can be generalized to wireless and switched networks in which link rate allocations are subject to fading and interference constraints.

[2]With a slight abuse of notation, we use $Q_n^{(c)}(t)$ to denote the queue, the packets in the queue, and the number of packets in the queue.

Proof of Lemma 1: See Appendix A. ■

In Lemma 1, (6) is the flow conservation constraint stating that the total flow rate of a class into a node is equal to the flow rate out of the node plus the queue overflow rate. Equation (7) is the link capacity constraint. The equality in (8) shows that the throughput of a class is equal to the sum of exogenous arrival rates less the queue overflow rates. Lemma 1 is closely related to the network capacity region Λ defined in terms of admissible arrival rates (see Definition 1); their relationship is shown in the next corollary.

Definition 1 ([23, Theorem 1]): The capacity region Λ of the network is the set of all arrival rate matrices $(\lambda_n^{(c)})_{n,c}$ for which there exist flow variables $f_l^{(c)} \geq 0$ such that

$$\lambda_n^{(c)} + \sum_{l \in L_{\text{in}}(n)} f_l^{(c)} \leq \sum_{l \in L_{\text{out}}(n)} f_l^{(c)} \qquad \forall c,\ n \neq \sigma_c \quad (9)$$

$$\sum_c f_l^{(c)} \leq \mu_l^{\max}, \qquad l \in \mathcal{L}. \quad (10)$$

Corollary 1: An arrival rate matrix $(\lambda_n^{(c)})$ lies in the network capacity region Λ if and only if there exist flow variables $f_l^{(c)} \geq 0$ such that flow conservation constraints (6) and link capacity constraints (7) hold with $q_n^{(c)} = 0$ for all n and c.

We remark that the solution to the utility maximization (2)–(3) activates the linear constraints (8); thus, the problem (2)–(3) is equivalent to

$$\text{maximize} \quad \sum_c g_c(r_c) \quad (11)$$

$$\text{subject to} \quad r_c = \sum_{n \neq \sigma_c} \lambda_n^{(c)} - \sum_{n \neq \sigma_c} q_n^{(c)} \qquad \forall c \quad (12)$$

$$(6) \text{ and } (7) \text{ hold} \quad (13)$$

$$f_l^{(c)} \geq 0,\ q_n^{(c)} \geq 0 \qquad \forall l, n, c. \quad (14)$$

Let (r_c^*) be the optimal throughput vector that solves (11)–(14). If the arrival rate matrix $(\lambda_n^{(c)})$ is in the network capacity region Λ, the optimal throughput is $r_c^* = \sum_{n \neq \sigma_c} \lambda_n^{(c)}$ for class $c \in \mathcal{C}$ data from Corollary 1. Otherwise, we have $r_c^* = \sum_{n \neq \sigma_c} (\lambda_n^{(c)} - q_n^{(c)*})$, where $q_n^{(c)*}$ is the optimal queue overflow rate.

C. Decomposition of the Network Control Problem

Our control policy that solves (11)–(14) has two main features. First, we have a packet-dropping mechanism discarding data from the network when queues build up. An observation is that, in order to optimize throughput and keep the network stable, we should drive the packet-dropping rate to be equal to the optimal queue overflow rate. Second, we need a flow controller driving the throughput vector toward the utility-optimal point. To convert the control objective (11) into these two control features, we define, for each class $c \in \mathcal{C}$, a utility function h_c related to g_c as

$$h_c(r_c) \triangleq g_c(r_c) - \theta_c r_c \quad (15)$$

where $\theta_c \geq 0$ are control parameters that will be chosen in the control policy. Using (12), we have

$$g_c(r_c) = h_c(r_c) + \theta_c \left[\sum_n \lambda_n^{(c)} - \sum_n q_n^{(c)} \right]. \quad (16)$$

Since $\lambda_n^{(c)}$ are unknown constants, maximizing $\sum_c g_c(r_c)$ is the same as maximizing

$$\sum_c \left[h_c(r_c) - \theta_c \sum_n q_n^{(c)} \right]. \quad (17)$$

This equivalent objective (17) is optimized by jointly maximizing the new utility $\sum_c h_c(r_c)$ at the receivers and minimizing the weighted queue overflow rates (i.e., the weighted packet-dropping rates) $\sum_c \theta_c q_n^{(c)}$ at each node n.

Optimizing the throughput vector $(r_c)_{c \in \mathcal{C}}$ at the receivers is nontrivial because it depends on the rates $(q_n^{(c)})_{n,c}$ at which packets are dropped across the network, and exogenous arrival rates are unknown. To develop a distributed control policy, we use the Lyapunov-drift-plus-penalty algorithm [15, Ch. 4] that allows us to transform the network utility maximization problem into an optimal queue control problem as follows (the Lyapunov-drift algorithm can be viewed as implementing the primal-dual method to solve optimization problems in stochastic networks [14, Sec. 4.10]). First, it is useful to introduce auxiliary control variables $\varphi_n^{(c)} \geq 0$ and $\nu_c \geq 0$ and consider an equivalent optimization problem to (11)–(14)

$$\text{maximize} \quad \sum_c h_c(\nu_c) - \sum_n \left[\sum_c \theta_c \varphi_n^{(c)} \right] \quad (18)$$

$$\text{subject to} \quad r_c = \nu_c \qquad \forall c \quad (19)$$

$$q_n^{(c)} \leq \varphi_n^{(c)} \qquad \forall c,\ n \neq \sigma_c \quad (20)$$

$$(12)\text{–}(14) \text{ hold.} \quad (21)$$

The equivalence of the two problems is easy to show after we replace (11) with (16). Second, we solve the problem (18)–(21) by designing a distributed network control policy with three tasks.

1) Each intermediate node n minimizes $\sum_c \theta_c \varphi_n^{(c)}$ in (18) and achieves the packet-dropping constraint $q_n^{(c)} \leq \varphi_n^{(c)}$ for all classes c in (20).
2) A receiver σ_c maximizes the utility $h_c(\nu_c)$ in (18) and achieves the throughput constraint $r_c = \nu_c$ in (19).
3) The flow conservation and link capacity constraints (21) are naturally satisfied under feasible network control decisions in every slot.

For the first task, we will construct a virtual queue $D_n^{(c)}(t)$ of which the quantities $q_n^{(c)}$ and $\varphi_n^{(c)}$ are the long-term arrival and service rate, respectively. Stabilizing the virtual queue $D_n^{(c)}(t)$ achieves the constraint $q_n^{(c)} \leq \varphi_n^{(c)}$ as a necessary condition for queue stability. Consequently, the first task is done by stabilizing virtual queues $D_n^{(c)}(t)$ while minimizing the weighted service rates $\theta_c \varphi_n^{(c)}$ of the queues $D_n^{(c)}(t)$. Similarly, the second task can be transformed into an optimal queue control problem that aims to equalize the arrival rate r_c and the service rate ν_c of a virtual queue and maximize the utility function $h_c(\nu_c)$. These two optimal queue control problems are solved by the Lyapunov-drift-plus-penalty algorithm, which gives rise to the network control policy. See Sections IV and V for details.

IV. Maximizing the Weighted Sum Throughput

For the ease of exposition, we first consider the special case of maximizing the weighted sum throughput in the network.

Define the linear utility function $g_c(r_c) = a_c r_c$ for each class $c \in \mathcal{C}$, where $a_c > 0$. We present a threshold-based packet-dropping policy that maximizes $\sum_c a_c r_c$. Surprisingly, flow control at the receivers is not needed here because maximizing the weighted sum throughput is equivalent to minimizing the weighted packet-dropping rate. Indeed, choosing $\theta_c = a_c$ in (15) yields $h_c = 0$ for all $c \in \mathcal{C}$. From (17), maximizing $\sum_c a_c r_c$ is equivalent to minimizing $\sum_{n,c} \theta_c q_n^{(c)}$. In Section V, we combine the threshold-based packet-dropping policy with receiver-end flow control to solve the general utility maximization problem (2)–(3).

A. Control Policy

Maximizing the weighted sum throughput requires each intermediate network node n to achieve the packet-dropping constraint $q_n^{(c)} \le \varphi_n^{(c)}$ and minimize $\theta_c \varphi_n^{(c)} = a_c \varphi_n^{(c)}$ for all classes c, according to the discussions in Section III-C. We define a virtual queue $D_n^{(c)}(t)$ associated with each physical queue $Q_n^{(c)}(t)$. The queue $D_n^{(c)}(t)$ evolves over slots according to

$$D_n^{(c)}(t+1) = \left[D_n^{(c)}(t) - \varphi_n^{(c)}(t)\right]^+ + \widetilde{d}_n^{(c)}(t) \tag{22}$$

where $\varphi_n^{(c)}(t) \in [0, d_{\max}]$ is the service allocation of the queue $D_n^{(c)}(t)$ in slot t, and $\widetilde{d}_n^{(c)}(t)$ is the number of packets discarded from the queue $Q_n^{(c)}(t)$ in slot t. From (1), we have

$$\widetilde{d}_n^{(c)}(t) = \min\left[\left(Q_n^{(c)}(t) - \sum_{l \in L_{\text{out}}(n)} \mu_l^{(c)}(t)\right)^+, d_n^{(c)}(t)\right] \tag{23}$$

which is strictly less than the allocated dropping rate $d_n^{(c)}(t)$ if $Q_n^{(c)}(t)$ has insufficient data. Assume $D_n^{(c)}(0) = V\theta_c = Va_c$ for all n and c, where $V > 0$ is a control parameter.[3] Consider the queueing dynamics (22) and the constraint $q_n^{(c)} \le \varphi_n^{(c)}$. The quantity $q_n^{(c)}$ represents the average packet-dropping rate at queue $Q_n^{(c)}(t)$, i.e., the average arrival rate of the virtual queue $D_n^{(c)}(t)$. Let $\varphi_n^{(c)}$ be the time average of service allocations $\varphi_n^{(c)}(t)$ at queue $D_n^{(c)}(t)$. From queueing theory, stabilizing the virtual queue $D_n^{(c)}(t)$ achieves $q_n^{(c)} \le \varphi_n^{(c)}$. Minimizing $a_c \varphi_n^{(c)}$ at node n for class c packets can be achieved by minimizing the weighted service allocation $a_c \varphi_n^{(c)}(t)$ at queue $D_n^{(c)}(t)$ in every slot. In this way, the task of an intermediate node n is converted into one that stabilizes the virtual queue $D_n^{(c)}(t)$ while minimizing its weighted service allocation $a_c \varphi_n^{(c)}(t)$ in every slot. This is solved by the Lyapunov-drift-plus-penalty algorithm (see details in Appendix C), according to which we propose the following policy.

Overload Resilient Algorithm (ORA)

Parameter Selection: Choose $\theta_c = a_c$ for all classes $c \in \mathcal{C}$, where $g_c(x) = a_c x$. Choose a parameter $V > 0$.

[3]It suffices to assume $D_n^{(c)}(0)$ to be finite. Our choice of $D_n^{(c)}(0) = V\theta_c$ avoids unnecessary packet dropping in the initial phase of the system.

Backpressure Routing: Over each link $l = (n, m) \in \mathcal{L}$, let $\mathcal{C}_l$ be the subset of classes that have access to link l. Compute the differential backlog $W_l^{(c)}(t) = Q_n^{(c)}(t) - Q_m^{(c)}(t)$ for each class $c \in \mathcal{C}_l$, where $Q_{\sigma_c}^{(c)}(t) = 0$ at the receiver σ_c. Define

$$W_l^{(c)*} = \max_{c \in \mathcal{C}_l} W_l^{(c)}(t), \quad c_l^* = \arg\max_{c \in \mathcal{C}_l} W_l^{(c)}(t).$$

Let the transmission rate of class c_l^* packets over link l be $\mu_l^{(c_l^*)}(t) = \mu_l^{\max} 1_{[W_l^{(c)*} > 0]}$, where $1_{[\cdot]}$ is an indicator function satisfying $1_{[E]} = 1$ if event E is true, and 0 otherwise. Let $\mu_l^{(c)}(t) = 0$ for all classes $c \in \mathcal{C}_l \setminus \{c_l^*\}$ over link l.

Packet Dropping: At queue $Q_n^{(c)}(t)$, allocate the packet-dropping rate

$$d_n^{(c)}(t) = d_{\max} 1_{\left[Q_n^{(c)}(t) > D_n^{(c)}(t)\right]} \tag{24}$$

where $d_{\max} > 0$ is a constant chosen to satisfy Assumption 1. At the virtual queue $D_n^{(c)}(t)$, allocate its service rate

$$\varphi_n^{(c)}(t) = d_{\max} 1_{\left[D_n^{(c)}(t) > V\theta_c\right]}. \tag{25}$$

Queue Update: Update queues $Q_n^{(c)}(t)$ and virtual queues $D_n^{(c)}(t)$ according to (1) and (22), (23) in every slot, respectively.

The packet-dropping subroutine in this policy is threshold-based. The choice of $d_n^{(c)}(t)$ in (24) is a back-pressure operation between the two queues $Q_n^{(c)}(t)$ and $D_n^{(c)}(t)$. The bang-bang choice of $\varphi_n^{(c)}(t)$ in (25) results from the aforementioned optimal queue control problem that has two conflicting goals: Stabilizing the virtual queue $D_n^{(c)}(t)$ needs large service allocations $\varphi_n^{(c)}(t)$, but minimizing the weighted average service rate $\theta_c \varphi_n^{(c)}$ requires small values of $\varphi_n^{(c)}(t)$. As it turns out, the ORA policy chooses the maximum value $\varphi_n^{(c)}(t) = d_{\max}$ if $D_n^{(c)}(t)$ grows beyond a threshold $V\theta_c$ for queue stability, and $\varphi_n^{(c)}(t) = 0$ otherwise. It is notable that the ORA policy needs only local information exchange between neighboring nodes and does not require the knowledge of exogenous arrival rates. Thus, network overload is autonomously resolved by each node making local decisions of routing, scheduling, and packet dropping.

B. Performance of the ORA Policy

Lemma 2 (Deterministic Bound for Queues): For each class $c \in \mathcal{C}$, define the constants

$$D_{\max}^{(c)} \triangleq V\theta_c + d_{\max} \quad Q_{\max}^{(c)} \triangleq V\theta_c + 2d_{\max}. \tag{26}$$

In the ORA policy, queues $Q_n^{(c)}(t)$ and $D_n^{(c)}(t)$ are deterministically bounded by

$$Q_n^{(c)}(t) \le Q_{\max}^{(c)} \quad D_n^{(c)}(t) \le D_{\max}^{(c)}, \quad \text{for all } t, c, \text{ and } n \ne \sigma_c$$

where $\theta_c = a_c$ for all classes c. In addition, we have $D_n^{(c)}(t) \ge V\theta_c - d_{\max}$ for all n, c, and t.

Proof: See Appendix B. ■

In Lemma 2, the term $Q_{\max}^{(c)}$ is the finite buffer size sufficient at queue $Q_n^{(c)}(t)$ and is chosen proportional to the coefficients $\theta_c = a_c$ in the linear objective function—the ORA policy gives a larger buffer to a traffic class that produces better rewards to avoid packet dropping in that class. The next theorem shows that the performance of the ORA policy approaches optimality as the buffer sizes increase.

Theorem 1: Define the long-term throughput of class c as

$$\overline{r_c} \triangleq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left[\sum_{l\in L_{\text{in}}(\sigma_c)} \widetilde{\mu}_l^{(c)}(\tau)\right] \tag{27}$$

where $\widetilde{\mu}_l^{(c)}(\tau)$ denotes the number of class c packets received by the destination σ_c over link l. The ORA policy yields the limiting weighted sum throughput satisfying

$$\sum_c a_c \overline{r_c} \geq \sum_c a_c r_c^* - \frac{B}{V} \tag{28}$$

where (r_c^*) is the optimal throughput vector that solves (11)–(14) under the linear objective function $\sum_c a_c r_c$, and B is a finite constant defined as

$$B \triangleq |\mathcal{N}||\mathcal{C}|\left[\left(\mu_{\max}^{\text{out}} + d_{\max}\right)^2 + \left(A_{\max} + \mu_{\max}^{\text{in}}\right)^2 + 2d_{\max}^2\right]$$

where $|\mathcal{A}|$ denotes the cardinality of a set $\mathcal{A}$.

We omit the proof of Theorem 1 because it is similar to that of Theorem 2 presented later in the general case of utility maximization. From (28), the ORA policy yields near-optimal performance by choosing the parameter V sufficiently large, where a large value of V implies a large buffer size of $Q_{\max}^{(c)} = V\theta_c + 2d_{\max}$. Moreover, as shown in Corollary 1, if the arrival rate matrix $(\lambda_n^{(c)})$ lies in the network capacity region Λ, then the optimal throughput for class c is $r_c^* = \sum_n \lambda_n^{(c)}$ and (28) reduces to $\sum_c a_c\overline{r_c} \geq \sum_c a_c(\sum_n \lambda_n^{(c)}) - (B/V)$. That we can choose V arbitrarily large leads to the next corollary.

Corollary 2: The ORA policy is (close to) throughput-optimal.

C. *Simplified* ORA *Policy*

Lemma 2 shows that the backlog of the virtual queue $D_n^{(c)}(t)$ takes values in the finite interval $[V\theta_c - d_{\max}, V\theta_c + d_{\max}]$ for all t under the ORA policy. This property of $D_n^{(c)}(t)$ allows us to simplify the ORA policy by using the imperfect queue-length information $D_n^{(c)}(t) = V\theta_c = Va_c$ for all n and t. This simplification eliminates the use of virtual queues $D_n^{(c)}(t)$. Because the error $|D_n^{(c)}(t) - V\theta_c|$ in the queue-length information is bounded by a finite constant $d_{\max}$, the performance of the simplified ORA policy remains optimal (as V goes to infinity).[4] The resulting policy is presented as follows.

[4]The optimality of the simplified ORA policy can be shown by plugging the inequality

$$\begin{aligned} &D_n^{(c)}(t)\mathbb{E}\left[\varphi_n^{(c)}(t) - d_n^{(c)}(t) \mid H(t)\right] \\ &\quad \geq V\theta_c \mathbb{E}\left[\varphi_n^{(c)}(t) - d_n^{(c)}(t) \mid H(t)\right] - d_{\max}^2 \end{aligned} \tag{29}$$

into the right-hand side of the (48) and following the rest of the Lyapunov-drift analysis. Equation (29) is easily derived using the property that $|D_n^{(c)}(t) - V\theta_c|$, $\varphi_n^{(c)}(t)$, and $d_n^{(c)}(t)$ take values in $[0, d_{\max}]$.

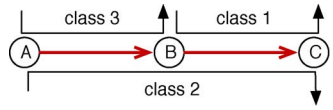


Fig. 3. Three-node network with three classes of traffic.

A Simplified ORA Policy

- Same as the ORA policy except that: (i) the virtual queues $D_n^{(c)}(t)$ are not used; (ii) we choose the packet-dropping rate $d_n^{(c)}(t) = d_{\max}1_{[Q_n^{(c)}(t) > V\theta_c]}$ at queue $Q_n^{(c)}(t)$.

It is not difficult to show that both the deterministic bound $Q_n^{(c)}(t) \leq Q_{\max}^{(c)}$ and Theorem 1 hold under the simplified ORA policy, except that a different finite constant B in (28) is used. The quantity $V\theta_c$ (or Va_c) is the threshold for dropping class c packets in the queue $Q_n^{(c)}(t)$. In order to maximize the weighted sum throughput, it makes sense to have a higher threshold for a traffic class that is "more important," so that packet dropping happens less likely in that class. Interestingly, it suffices to choose the threshold Va_c for class c traffic, where the thresholds are proportional to the weights a_c in the linear utility function.

D. *Simulation of the* ORA *Policy*

We conduct simulations for the ORA policy in the network shown in Fig. 3. The directed links (A, B) and (B, C) have the capacity of 1 packet/slot. There are three classes of traffic to be served. For example, class 1 data arrives at node B and is destined for node C. Classes 1 and 2 compete for service over (B, C), and classes 2 and 3 compete over (A, B). Each simulation below is run over 10^6 slots.

1) Fixed Arrival Rates: In each class, we assume a Bernoulli arrival process whereby 20 packets arrive to the network in a slot with probability 0.1, and no packets arrive otherwise. The arrival rate of each class is 2 packets/slot, which clearly overloads the network.

Let r_c be the throughput of class c. Consider the objective of maximizing the weighted sum throughput $3r_1 + 2r_2 + r_3$; the weights are rewards obtained by serving a packet in a class. The optimal solution is: 1) Always serve class 1 at node B because it yields better rewards than serving class 2. 2) Always serve class 3 at node A—although class 2 has better rewards than class 3, it does not make sense to serve class 2 at A only to be dropped later at B. The optimal throughput vector is therefore (1, 0, 1). Consider another objective of maximizing $3r_1 + 5r_2 + r_3$. Here, class 2 has a reward that is better than the sum of rewards of the other two classes. Thus, both nodes A and B should always serve class 2; the optimal throughput vector is (0, 1, 0). Table I shows the near-optimal performance of the ORA policy in both cases as V increases; see Fig. 4 for the running average throughput under the policy.

2) Time-Varying Arrival Rates: We show that the ORA policy is robust to time-varying arrival rates. Suppose classes 1 and 3 have a fixed arrival rate of 0.8 packets/slot. The arrival rate of class 2 is 2 packets/slot in the interval $\mathcal{T} = [3\times10^5, 6\times10^5)$,

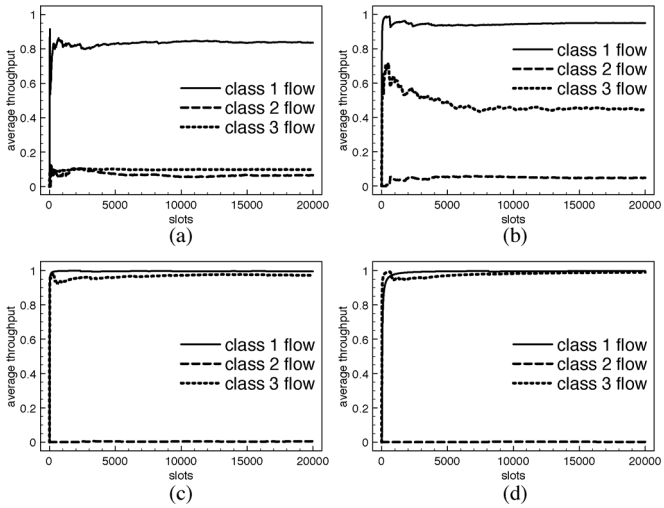


Fig. 4. Running average throughput of the three classes of flows in the line network in Fig. 3, with the objective of maximizing $3r_1 + 2r_2 + r_3$. (a) $V = 10$. (b) $V = 20$. (c) $V = 50$. (d) $V = 100$.

TABLE I
THROUGHPUT PERFORMANCE OF THE ORA POLICY UNDER FIXED ARRIVAL RATES. (a) MAXIMIZING $3r_1 + 2r_2 + r_3$. (b) MAXIMIZING $3r_1 + 5r_2 + r_3$

(a)				(b)			
V	r_1	r_2	r_3	V	r_1	r_2	r_3
10	.787	.168	.099	10	.185	.815	.083
20	.867	.133	.410	20	.107	.893	.095
50	.992	.008	.967	50	.031	.969	.031
100	.999	0	.999	100	.002	.998	.001
opt	1	0	1	opt	0	1	0

TABLE II
THROUGHPUT PERFORMANCE OF THE ORA POLICY UNDER TIME-VARYING ARRIVAL RATES

time interval	throughput in this interval	optimal value
$[0, 3 \cdot 10^5)$	$(.807, .104, .8)$	$(.8, .1, .8)$
$[3 \cdot 10^5, 6 \cdot 10^5)$	$(.002, .998, 0)$	$(0, 1, 0)$
$[6 \cdot 10^5, 10^6)$	$(.793, .106, .796)$	$(.8, .1, .8)$

and is 0.1 packets/slot elsewhere. We consider the objective of maximizing $3r_1 + 5r_2 + r_3$. The network is temporarily overloaded in the interval $\mathcal{T}$, and the optimal throughput in $\mathcal{T}$ is (0, 1, 0) as explained in the above case. The network is underloaded in the interval $[0, 10^6) \setminus \mathcal{T}$, in which the optimal throughput is (0.8, 0.1, 0.8).

We use the following parameters: $V = 200$, $A_{\max} = 20$, $d_{\max} = A_{\max} + \mu^{\text{in}}_{\max} = 21$, and $(\theta_1, \theta_2, \theta_3) = (3, 5, 1)$. Table II shows the near-optimal throughput performance of the ORA policy. Fig. 5 shows the sample paths of the queue processes $Q_B^{(1)}(t)$, $Q_B^{(2)}(t)$, $Q_A^{(2)}(t)$, and $Q_A^{(3)}(t)$ in the simulation. The queue backlogs suddenly build up when the network enters the overload interval $\mathcal{T}$, but are kept close to the upper bound $Q^{(c)}_{\max} = V\theta_c + 2d_{\max}$ without growing unbounded.

V. UTILITY-OPTIMAL CONTROL

We solve the general utility maximization problem (2)–(3) with a network control policy similar to the ORA policy except for an additional flow control mechanism.

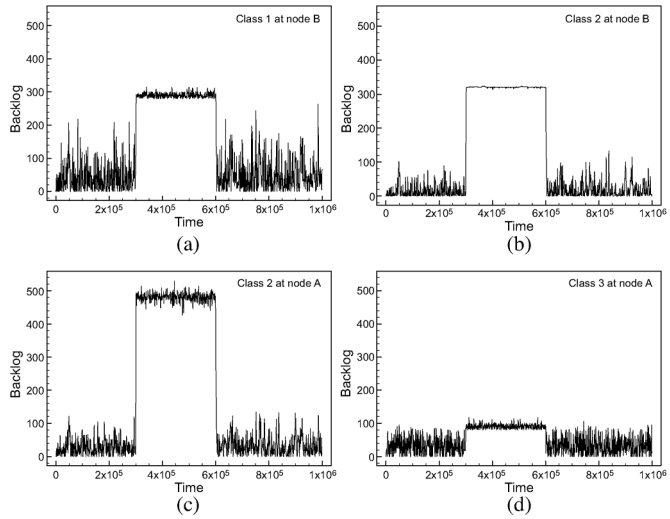


Fig. 5. Queue processes under the ORA policy with time-varying arrival rates that temporarily overload the network. (a) Queue $Q_B^{(1)}(t)$. (b) Queue $Q_B^{(2)}(t)$. (c) Queue $Q_A^{(2)}(t)$. (d) Queue $Q_A^{(3)}(t)$.

A. Virtue Queue

In Section III-C, we explained that solving the equivalent utility maximization problem (18)–(21) needs each receiver σ_c to maximize the new utility function $h_c(\nu_c)$ subject to $r_c = \nu_c$, where ν_c is an auxiliary control variable and r_c is the throughput of class c packets. We transform this task into an optimal queue control problem as follows. Let $\widetilde{\mu}_l^{(c)}(t)$ be the number of class c packets received by the destination σ_c over link $l = (a, \sigma_c)$ in slot t, i.e., $\widetilde{\mu}_l^{(c)}(t) = \min[Q_a^{(c)}(t), \mu_{a\sigma_c}^{(c)}(t)]$, $l \in L_{\text{in}}(\sigma_c)$. Define a virtual queue $Z_c(t)$ that is located at the receiver σ_c and evolves over slots according to

$$Z_c(t+1) = [Z_c(t) - \nu_c(t)]^+ + \sum_{l \in L_{\text{in}}(\sigma_c)} \widetilde{\mu}_l^{(c)}(t) \qquad (30)$$

where $\nu_c(t)$ is the service allocation in slot t. Observe that the throughput r_c of class c packets is the average "data arrival rate" of the virtual queue $Z_c(t)$. Let ν_c be the time average of $\nu_c(t)$ in (30). Then, the task of a receiver σ_c is done by equalizing the arrival rate r_c and the service rate ν_c of the virtual queue $Z_c(t)$ while maximizing $h_c(\nu_c)$. This optimal queue control problem is solved by the Lyapunov-drift-plus-penalty algorithm with a carefully chosen Lyapunov function. Specifically, it is known that if queue $Z_c(t)$ is stable, then $r_c \le \nu_c$. However, we are interested in the stronger relationship that stabilizing $Z_c(t)$ leads to $r_c = \nu_c$. It suffices to guarantee that the queues $Z_c(t)$ do not waste service opportunities, for which we need two conditions.

1) The queues $Z_c(t)$ usually have more than enough (virtual) data to serve and are stable.
2) When $Z_c(t)$ does not have sufficient data, the allocated service rate $\nu_c(t) = 0$.

To attain the first condition, we use an exponential Lyapunov function that forces the virtual backlog processes $Z_c(t)$ away from zero and centers them around a parameter $Q > 0$; the widely used quadratic Lyapunov function does not work in this case. The second condition is attained by choosing the parameters $\theta_c \ge g_c'(0)$ to define the utility functions h_c; see Section V-C for details.

B. Control Policy

The following policy, constructed in Appendixes C and D as an application to the Lyapunov-drift-plus-penalty algorithm, solves the general utility maximization problem (2)–(3).

Utility-Optimal Overload-Resilient Algorithm (UORA)

Parameter Selection: Choose positive parameters $\nu_{\max}$, w, V, Q, and $\{\theta_c, c \in \mathcal{C}\}$ to be discussed in Section V-C. Define $Q_n^{(c)}(0) = 0$ for all n and c, $Z_c(0) = Q$ for all c, and $D_n^{(c)}(0) = V\theta_c$ for all n.

Packet Dropping: Same as the ORA policy.

Backpressure Routing: Same as the ORA policy, except that the differential backlog over each link $l = (a, \sigma_c) \in \mathcal{L}$ connected to a receiver σ_c is modified as

$$W_l^{(c)}(t) = Q_a^{(c)}(t) - Q_{\sigma_c}^{(c)}(t) \tag{31}$$

where we abuse the notation by redefining

$$Q_{\sigma_c}^{(c)}(t) = \begin{cases} we^{w(Z_c(t)-Q)}, & \text{if } Z_c(t) \geq Q \\ -we^{w(Q-Z_c(t))}, & \text{if } Z_c(t) < Q \end{cases} \tag{32}$$

for all classes c. The exponential form of $Q_{\sigma_c}^{(c)}(t)$ is a result of using exponential Lyapunov functions. We emphasize that here $Q_{\sigma_c}^{(c)}(t)$ has nothing to do with real data buffered at the receivers (which must be zero); it is just a function of the virtual queue backlog $Z_c(t)$ that induces the "desired force" in the form of differential backlog in (31) to pull or push back data in the network. Thus, unlike standard back-pressure routing that has $Q_{\sigma_c}^{(c)}(t) = 0$, here we use $Q_{\sigma_c}^{(c)}(t)$ as part of the receiver-based flow control mechanism.

Receiver-Based Flow Control: At a destination σ_c, choose the virtual service rate $\nu_c(t)$ of queue $Z_c(t)$ as the solution to

$$\text{maximize} \quad Vh_c(\nu_c(t)) + \nu_c(t)Q_{\sigma_c}^{(c)}(t) \tag{33}$$

$$\text{subject to} \quad 0 \leq \nu_c(t) \leq \nu_{\max} \tag{34}$$

where $h_c(x) = g_c(x) - \theta_c x$.

Queue Update: Update queues $Q_n^{(c)}(t)$, $D_n^{(c)}(t)$, and $Z_c(t)$ according to (1), (22), and (30), respectively, in every slot.

The maximization problem (33)–(34) comes from the Lyapunov-drift-plus-penalty algorithm; see Appendix C. Intuitively, maximizing the first term of (33) in every slot is useful to maximize the long-term utility $h_c(\nu_c)$. Maximizing the second term in (33) captures the need of stabilizing the virtual queue $Z_c(t)$ while keeping its backlog around Q. Specifically, if the queue backlog $Z_c(t)$ is large so that $Q_{\sigma_c}^{(c)}(t) > 0$, then the second term of (33) is used to allocate a large service rate $\nu_c(t)$ to reduce the value of $Z_c(t)$ for queue stability. If $Z_c(t) < Q$ and $Q_{\sigma_c}^{(c)}(t) < 0$, then the solution to (33)–(34) chooses $\nu_c(t) = 0$ provided that $\theta_c \geq g_c'(0)$; the queue backlog $Z_c(t)$ is allowed to increase. In the latter case, choosing $\nu_c(t) = 0$ conflicts with the job of maximizing $h_c(\nu_c)$; the parameter V in (33) is used to control the relative importance of the two jobs.

We can apply the simplified ORA policy described in Section IV-C to the UORA policy to simplify the packet-dropping mechanism, in which case the virtual queues $D_n^{(c)}(t)$ are not needed. Such simplification does not affect the performance of the UORA policy.

C. Choice of Parameters

We briefly discuss how the parameters in the UORA policy are chosen. Let $\epsilon > 0$ be a small constant that affects the performance of the UORA policy [cf. (36)]. In (33)–(34), we need the parameter $\nu_{\max}$ to satisfy $\nu_{\max} \geq \max_c r_c^* + \epsilon/2$, where (r_c^*) is solution to the utility maximization (2)–(3) for a given exogenous arrival rate vector $(\lambda_n^{(c)})$. This choice of $\nu_{\max}$ ensures that the virtual queue $Z_c(t)$ can be stabilized when its arrival rate is the optimal throughput r_c^*. One feasible choice of $\nu_{\max}$ is the sum of capacities of all links connected to the receivers plus $\epsilon/2$. Due to technical reasons, we define $\delta_{\max} \triangleq \max[\nu_{\max}, \mu_{\max}^{\text{in}}]$ and choose the parameter $w \triangleq (\epsilon/\delta_{\max}^2)e^{-\epsilon/\delta_{\max}}$ in (32). The parameter Q in (32) is used to bound the queues $Z_c(t)$ away from zero and center them around Q; for technical reasons, we need $Q \geq \nu_{\max}$. The parameters θ_c are chosen to satisfy $h_c'(x) = g_c'(x) - \theta_c \leq 0$ for all $x \geq 0$. Any θ_c satisfying $\theta_c \geq g_c'(0)$ is a feasible choice because

$$h_c'(x) = g_c'(x) - \theta_c \leq g_c'(0) - \theta_c \leq 0 \qquad \forall x \geq 0.$$

This value of θ_c ensures that the service rate $\nu_c(t)$ of the virtual queue $Z_c(t)$, as the solution to (33)–(34), is zero whenever $Z_c(t) < Q$. This enforces the second condition described in Section V-A to equalize the arrival rate and the service rate of the queue $Z_c(t)$ (see Lemma 6 in Appendix F for details). The parameter V captures the tradeoff between utility and buffer sizes presented in Section V-D and is chosen sufficiently large; for technical reasons, we need V to satisfy $V\theta_c + 2d_{\max} \geq w$.

D. Performance Analysis

Lemma 3: In the UORA policy, queues $Q_n^{(c)}(t)$, $D_n^{(c)}(t)$, and $Z_c(t)$ are deterministically bounded by

$$Q_n^{(c)}(t) \leq Q_{\max}^{(c)} \quad D_n^{(c)}(t) \leq D_{\max}^{(c)} \quad Z_c(t) \leq Z_{\max}^{(c)}, \qquad \forall t, c, n$$

where $Q_{\max}^{(c)}$ and $D_{\max}^{(c)}$ are defined in (26) and

$$Z_{\max}^{(c)} \triangleq Q + \frac{1}{w}\log\left(\frac{V\theta_c + 2d_{\max}}{w}\right) + \mu_{\max}^{\text{in}}. \tag{35}$$

Proof of Lemma 3: See Appendix E. ∎

Theorem 2: The UORA policy yields the limiting utility that satisfies

$$\sum_c g_c(\overline{r_c}) \geq \sum_c g_c(r_c^*) - \frac{B_1}{V} - \frac{\epsilon}{2}\sum_c (g_c'(0) + \theta_c) \tag{36}$$

where $\overline{r_c}$ is defined in (27), (r_c^*) is the throughput vector that solves the utility maximization problem (2)–(3), and B_1 is a finite constant [defined in (82)].

Proof: See Appendix F. ∎

Theorem 2 shows that the performance gap from the optimal utility can be made arbitrarily small by choosing a large V and a small ϵ. The performance tradeoff of choosing a large V is again on the required finite buffer size $Q_{\max}^{(c)} = V\theta_c + 2d_{\max}$.

E. Simulation of the UORA Policy

We conduct two sets of simulations.

TABLE III
THROUGHPUT PERFORMANCE OF THE UORA POLICY IN THE THREE-NODE NETWORK

V	r_1	r_2	r_3	$\sum \log(r_c)$
10	.522	.478	.522	-2.038
20	.585	.415	.585	-1.952
50	.631	.369	.631	-1.918
100	.648	.352	.647	-1.912
optimal	.667	.333	.667	-1.910

TABLE IV
MAXIMUM BACKLOG IN QUEUES UNDER THE UORA POLICY IN THE THREE-NODE NETWORK

V	$Q_B^{(1)}(t)$	$Q_B^{(2)}(t)$	$Q_A^{(2)}(t)$	$Q_A^{(3)}(t)$	$Q_{\max}^{(c)}$
10	140	97	137	137	142
20	237	187	240	236	242
50	539	441	538	540	542
100	1036	865	1039	1039	1042

1) On the Three-Node Network in Fig. 3: The goal is to provide proportional fairness to the three classes of traffic; equivalently, we maximize the objective function $\log(r_1)+\log(r_2)+\log(r_3)$. Each directed link (A, B) and (B, C) has the capacity of one packet per slot. The arrival process for each class is that, in every slot, 20 packets arrive to the network with probability 0.1, and zero packets arrive otherwise. The arrival rate vector is (2, 2, 2), which overloads the network. In this setting, due to symmetry, the optimal throughput for class 1 is equal to that of class 3, which is the solution to

$$\text{maximize:}\quad 2\log(x)+\log(1-x),\quad \text{subject to:}\quad 0\le x\le 1.$$

The optimal throughput vector is (2/3, 1/3, 2/3), and the optimal utility is -1.91.

As explained in Section V-C, we choose the parameters of the UORA policy as follows. Let $\epsilon = 0.1$ and $\theta_c = 10$ for all classes c. By definition, $\mu_{\max}^{\text{in}} = 1$. The optimal throughput vector satisfies $\max_c r_c^* = 1$ for the given arrival rate vector (2, 2, 2), and we choose $\nu_{\max} = 3$ (any value of $\nu_{\max}$ greater than $\max_c r_c^* + \epsilon/2 = 1.05$ works). By definition, $\delta_{\max} = \max[\nu_{\max}, \mu_{\max}^{\text{in}}] = 3$. In the arrival processes, we have $A_{\max} = 20$. By Assumption 1, we choose $d_{\max} = A_{\max} + \mu_{\max}^{\text{in}} = 21$. Let $Q = 1000$.

We simulate the UORA policy for different values of V (i.e., different buffer sizes). The simulation time is 10^6 slots. Table III demonstrates the near-optimal performance of the UORA policy. Table IV shows the maximum backlog in each queue $Q_n^{(c)}(t)$ during the simulation. Consistent with Lemma 3, the maximum backlog at queue $Q_n^{(c)}(t)$ is bounded by $Q_{\max}^{(c)} = V\theta_c + 2d_{\max} = 10V + 42$.

2) On the Tree Network in Fig. 1(a): Consider providing max-min fairness to the three classes of traffic in Fig. 1(a). Each link has the capacity of one packet per slot. In each of the four arrival processes, 20 packets arrive in a slot with probability 0.1, and zero packets arrive otherwise. The arrival rates are (2, 2, 2, 2), which overload the network. The optimal throughput for the three classes is easily seen to be (2/3, 2/3, 2/3), where each flow of class 1 contributes equally in that class.

TABLE V
THROUGHPUT PERFORMANCE OF THE UORA POLICY IN THE TREE NETWORK

V	$r_{1A}+r_{1C}$	r_{2B}	r_{3D}
10	.200	.100	.100
20	.364	.206	.205
30	.661	.650	.651
50	.667	.667	.667
optimal	.667	.667	.667

We approximate max-min fairness by using the α-fairness functions $g_c(x) = x^{1-\alpha}/(1-\alpha)$ with a large value of $\alpha = 100$. The utility maximization becomes

$$\begin{aligned}&\text{maximize}\quad \frac{-1}{99}\left[(r_{1A}+r_{1C})^{-99}+(r_{2B})^{-99}+(r_{3D})^{-99}\right]\\ &\text{subject to}\quad (r_{1A}, r_{1C}, r_{2B}, r_{3D}) \text{ feasible in Fig. 1(a)}\end{aligned}$$

where r_{1A} is the throughput of class-1 flow originating from node A; the other variables are similarly defined.

According to Section V-C, we choose the parameters of the UORA policy as follows. Let $\theta_c = \epsilon = 1$ for all classes c. The optimal throughput vector satisfies $\max_c r_c^* = 2$ for the given arrival rate vector (2, 2, 2, 2), achieved when the network always serves class 1. We choose $\nu_{\max} = 4$ (any value of $\nu_{\max}$ greater than $\max_c r_c^* + \epsilon/2 = 2.5$ works). We observe from Fig. 1(a) that $\mu_{\max}^{\text{in}} = 2$ and $\delta_{\max} = \max[\nu_{\max}, \mu_{\max}^{\text{in}}] = 4$. Choose $A_{\max} = 20$ in the arrival processes and $d_{\max} = A_{\max} + \mu_{\max}^{\text{in}} = 22$ by Assumption 1. Let $Q = 1000$. We simulate the UORA policy for different values of V, and each simulation takes 10^6 slots. The near-optimal performance of the UORA policy is given in Table V.

VI. CONCLUSION

We developed a receiver-based flow control and threshold-based packet-dropping policy to cope with network overload and achieve optimal utility. Our scheme is robust to uncooperative users who do not employ source-end flow control and to malicious users that intentionally overload the network. A novel feature of our policy is a receiver-based backpressure/push-back mechanism that regulates data flows at the granularity of traffic classes, where packets can be classified based on their types. This is in contrast to source-based schemes that can only differentiate between source–destination pairs. Our control policy may be useful to handle different types of service requests in Internet application servers under overload conditions, or manage multicommodity flows in finite-buffer networks with performance guarantees.

The receiver-based flow control scheme has a wide range of potential applications, including preventing denial-of-service attacks in Web servers, mitigating overload conditions that may arise when the network is experiencing significant degradation due to a disaster or attack, and even regulating traffic flows in the Internet. This framework also gives rise to a number of future research directions, such as accounting for the "cost" of packet dropping (e.g., due to the need to retransmit the dropped packets). A closely related problem involves the interaction between TCP-based flow control and the receiver-based flow control scheme, e.g., TCP's response to the packet-dropping mechanism. In this context, it would also be interesting to develop a

mathematical model to study optimal overload control in a network serving TCP flows. Another interesting future research direction is to use our framework to study traffic offloading problems in wireline and wireless networks, where traffic offloading is analogous to "dropping" data from the overloaded network to an alternative backup network.

APPENDIX A
PROOF OF LEMMA 1

First, we show (6)–(8) are necessary conditions. Given a control policy, let $F_l^{(c)}(t)$ be the class c packets transmitted over link l in the interval $[0,t]$, and $Q_n^{(c)}(t)$ be the class c packets queued at node n at time t. From the fact that the difference between incoming and outgoing packets at a node in $[0,t]$ is equal to the queue backlog at time t, we have

$$\sum_{\tau=0}^{t-1} A_n^{(c)}(\tau) + \sum_{l\in L_{\rm in}(n)} F_l^{(c)}(t) = Q_n^{(c)}(t) + \sum_{l\in L_{\rm out}(n)} F_l^{(c)}(t) \tag{37}$$

which holds for all nodes $n \neq \sigma_c$ for each class $c \in \mathcal{C}$. Taking expectation and time average of (37), we obtain

$$\lambda_n^{(c)} + \frac{1}{t}\mathbb{E}\left[\sum_{l\in L_{\rm in}(n)} F_l^{(c)}(t)\right] = \frac{\mathbb{E}\left[Q_n^{(c)}(t)\right]}{t} + \frac{1}{t}\mathbb{E}\left[\sum_{l\in L_{\rm out}(n)} F_l^{(c)}(t)\right]. \tag{38}$$

The link capacity constraints lead to

$$\frac{1}{t}\sum_c \mathbb{E}\left[F_l^{(c)}(t)\right] \le \mu_l^{\max}, \qquad l \in \mathcal{L}. \tag{39}$$

In (38), the sequence $\{\mathbb{E}[F_l^{(c)}(t)]/t, t \in \mathbb{Z}^+\}$ for each (l,c) pair is bounded because all links have finite capacity. Thus, all sequences $\{\mathbb{E}[Q_n^{(c)}(t)]/t, t \in \mathbb{Z}^+\}$ are bounded as well. There is a subsequence $\{t_k\}$ such that limit points $f_l^{(c)}$ and $q_n^{(c)}$ exist and satisfy, as $k \to \infty$

$$\frac{1}{t_k}\mathbb{E}\left[F_l^{(c)}(t_k)\right] \to f_l^{(c)} \qquad \forall c \in \mathcal{C},\ l \in \mathcal{L} \tag{40}$$

$$\frac{1}{t_k}\mathbb{E}\left[Q_n^{(c)}(t_k)\right] \to q_n^{(c)} \qquad \forall c \in \mathcal{C},\ n \neq \sigma_c. \tag{41}$$

Applying (40)–(41) to (38), (39) yields (6) and (7). Define the aggregate throughput r_c of class c as

$$r_c \triangleq \liminf_{t\to\infty} \frac{1}{t}\sum_{l\in L_{\rm in}(\sigma_c)} \mathbb{E}\left[F_l^{(c)}(t)\right]. \tag{42}$$

The inequality in (8) follows from (42) and (40). The equality in (8) results from summing (6) over $n \neq \sigma_c$.

To show the converse, it suffices to show that every interior point of $\Lambda_{\boldsymbol{\lambda}}$ is achievable. Let $(r_c)_{c\in\mathcal{C}}$ be an interior point of $\Lambda_{\boldsymbol{\lambda}}$, i.e., there exists $\epsilon \in (0,1)$ such that $(r_c+\epsilon)_{c\in\mathcal{C}} \in \Lambda_{\boldsymbol{\lambda}}$. There exist corresponding flow variables $f_l^{(c)}$ and $q_n^{(c)}$ such that

$$\lambda_n^{(c)} + \sum_{l\in L_{\rm in}(n)} f_l^{(c)} = q_n^{(c)} + \sum_{l\in L_{\rm out}(n)} f_l^{(c)} \tag{43}$$

$$\sum_c f_l^{(c)} \le \mu_l^{\max} \quad r_c + \epsilon \le \sum_n \lambda_n^{(c)} - \sum_n q_n^{(c)}. \tag{44}$$

In this system of flows (43)–(44), by removing subflows that contribute to queue overflows, we obtain new flow variables $\hat{\lambda}_n^{(c)}$ and $\hat{f}_l^{(c)}$ such that $0 \le \hat{\lambda}_n^{(c)} \le \lambda_n^{(c)}$, $0 \le \hat{f}_l^{(c)} \le f_l^{(c)}$, and

$$\hat{\lambda}_n^{(c)} + \sum_{l\in L_{\rm in}(n)} \hat{f}_l^{(c)} = \sum_{l\in L_{\rm out}(n)} \hat{f}_l^{(c)} \qquad \forall c,\ n \neq \sigma_c \tag{45}$$

$$\sum_c \hat{f}_l^{(c)} \le \mu_l^{\max}, \qquad l \in \mathcal{L} \tag{46}$$

$$r_c + \epsilon \le \sum_n \hat{\lambda}_n^{(c)}, \qquad c \in \mathcal{C}. \tag{47}$$

Define

$$r_n^{(c)} \triangleq \hat{\lambda}_n^{(c)}\left[1 - \frac{\frac{\epsilon}{2}}{\sum_n \hat{\lambda}_n^{(c)}}\right]\frac{r_c}{\sum_n \hat{\lambda}_n^{(c)} - \epsilon/2}.$$

From (47), we have $\sum_n \hat{\lambda}_n^{(c)} > \epsilon/2$ and $\sum_n \hat{\lambda}_n^{(c)} - \epsilon/2 \ge r_c$. It is not difficult to check that $r_n^{(c)} \ge 0$, $r_n^{(c)} < \hat{\lambda}_n^{(c)}$, and $\sum_n r_n^{(c)} = r_c$. Combining them with (45) and (46) yields

$$r_n^{(c)} + \sum_{l\in L_{\rm in}(n)} \hat{f}_l^{(c)} < \sum_{l\in L_{\rm out}(n)} \hat{f}_l^{(c)}, \quad \sum_c \hat{f}_l^{(c)} \le \mu_l^{\max}.$$

These inequalities show that the rate matrix $(r_n^{(c)})$ is an interior point of the network capacity region Λ in Definition 1, and therefore is achievable by a control policy, such as the back-pressure policy [23]. As a result, the aggregate rate vector (r_c), where $r_c = \sum_n r_n^{(c)}$, is also achievable.

APPENDIX B
PROOF OF LEMMA 2

We prove Lemma 2 by induction. First, we show $D_n^{(c)}(t)$ is deterministically bounded. Assume $D_n^{(c)}(t) \le D_{\max}^{(c)}$ for some t, which holds at $t=0$ since we let $D_n^{(c)}(0) = V\theta_c$. Consider two cases.

1) If $D_n^{(c)}(t) \le V\theta_c$, then from (22) we obtain

$$D_n^{(c)}(t+1) \le D_n^{(c)}(t) + \widetilde{d}_n^{(c)}(t) \le V\theta_c + d_{\max} = D_{\max}^{(c)}$$

where the second inequality uses (23).

2) If $D_n^{(c)}(t) > V\theta_c$, then the ORA policy selects $\varphi_n^{(c)}(t) = d_{\max}$ in the queue $D_n^{(c)}(t)$, and

$$D_n^{(c)}(t+1) \le \left[D_n^{(c)}(t) - d_{\max}\right]^+ + d_{\max} \le \max\left[D_n^{(c)}(t),\ d_{\max}\right] \le D_{\max}^{(c)}$$

where the last inequality uses the induction assumption.

We conclude that $D_n^{(c)}(t+1) \le D_{\max}^{(c)}$.

Next, we show $Q_n^{(c)}(t)$ is bounded. Assume $Q_n^{(c)}(t) \le Q_{\max}^{(c)}$ for some t, which holds at $t=0$ because we let $Q_n^{(c)}(0) = 0$. Consider two cases.

1) If $Q_n^{(c)}(t) \le D_{\max}^{(c)}$, then from (1) we get

$$Q_n^{(c)}(t+1) \le Q_n^{(c)}(t) + A_{\max} + \mu_{\max}^{\rm in} \le D_{\max}^{(c)} + d_{\max} = Q_{\max}^{(c)}$$

where the second inequality uses Assumption 1.

2) If $Q_n^{(c)}(t) > D_{\max}^{(c)} \geq D_n^{(c)}(t)$, the ORA policy selects $d_n^{(c)}(t) = d_{\max}$ at queue $Q_n^{(c)}(t)$, resulting in

$$\begin{aligned} Q_n^{(c)}(t+1) &\leq Q_n^{(c)}(t) - d_{\max} + A_{\max} + \mu_{\max}^{\text{in}} \\ &\leq Q_n^{(c)}(t) \leq Q_{\max}^{(c)} \end{aligned}$$

where the third inequality uses induction assumption.

We conclude that $Q_n^{(c)}(t+1) \leq Q_{\max}^{(c)}$.

Finally, we show $D_n^{(c)}(t) \geq V\theta_c - d_{\max}$ for all slots. Assume this is true at some time t; this holds when $t = 0$ because we define $D_n^{(c)}(0) = V\theta_c$. Consider two cases.

1) If $D_n^{(c)}(t) \leq V\theta_c$, then the ORA policy selects $\varphi_n^{(c)}(t) = 0$ in the queue $D_n^{(c)}(t)$, and we have $D_n^{(c)}(t+1) \geq D_n^{(c)}(t) \geq V\theta_c - d_{\max}$ by induction assumption.
2) If $D_n^{(c)}(t) > V\theta_c$, then the ORA policy selects $\varphi_n^{(c)}(t) = d_{\max}$, and we have $D_n^{(c)}(t+1) \geq D_n^{(c)}(t) - d_{\max} > V\theta_c - d_{\max}$.

The proof is complete.

Appendix C

We construct a proper Lyapunov drift inequality that leads to the UORA policy. Let $H(t) = (Q_n^{(c)}(t); D_n^{(c)}(t); Z_c(t))$ be the vector of all physical and virtual queues in the network. Using the parameters w and Q given in the policy, we define the Lyapunov function

$$\begin{aligned} (H(t)) \triangleq \frac{1}{2} \sum_{c,n \neq \sigma_c} \left[Q_n^{(c)}(t)\right]^2 + \frac{1}{2} \sum_{c,n \neq \sigma_c} \left[D_n^{(c)}(t)\right]^2 \\ + \sum_c \left(e^{w(Z_c(t)-Q)} + e^{w(Q-Z_c(t))} \right). \end{aligned}$$

The last sum is a Lyapunov function whose value grows exponentially large whenever $Z_c(t)$ deviates in both directions from Q. This exponential Lyapunov function [24] is useful for both stabilizing $Z_c(t)$ and guaranteeing there is sufficient backlog in $Z_c(t)$. We define the Lyapunov drift $\Delta(t) \triangleq \mathbb{E}[L(H(t+1)) - L(H(t))|H(t)]$, where the expectation is with respect to all randomness in the system in slot t.

Define $1_c^{\text{R}}(t) = 1_{[Z_c(t) \geq Q]}$ and $1_c^{\text{L}}(t) = 1 - 1_c^{\text{R}}(t)$, where $1_{[E]}$ is an indicator function for the event E. Let $\delta_c(t) = \nu_c(t) - \sum_{l \in L_{\text{in}}(\sigma_c)} \mu_l^{(c)}(t)$. The next lemma is proved in Appendix D.

Lemma 4: The Lyapunov drift $\Delta(t)$ under any control policy satisfies

$$\begin{aligned} \Delta(t) - V \sum_c \mathbb{E}\left[h_c\left(\nu_c(t)\right)|H(t)\right] + V \sum_{n,c} \theta_c \mathbb{E}\left[\varphi_n^{(c)}(t)|H(t)\right] \\ \leq B + \sum_{n,c} Q_n^{(c)}(t) \lambda_n^{(c)} - \sum_{n,c} Q_n^{(c)}(t) \mathbb{E}\left[d_n^{(c)}(t)|H(t)\right] \\ - \sum_{n,c} Q_n^{(c)}(t) \mathbb{E}\left[\sum_{l \in L_{\text{out}}(n)} \mu_l^{(c)}(t) - \sum_{l \in L_{\text{in}}(n)} \mu_l^{(c)}(t) | H(t)\right] \\ - \sum_{n,c} D_n^{(c)}(t) \mathbb{E}\left[\varphi_n^{(c)}(t) - d_n^{(c)}(t) \mid H(t)\right] \\ - V \sum_c \mathbb{E}[h_c\left(\nu_c(t)\right)|H(t)] + V \sum_{n,c} \theta_c \mathbb{E}\left[\varphi_n^{(c)}(t)|H(t)\right] \\ - w \sum_c 1_c^{\text{R}}(t) e^{w(Z_c(t)-Q)} \left(\mathbb{E}\left[\delta_c(t) \mid H(t)\right] - \frac{\epsilon}{2}\right) \\ + w \sum_c 1_c^{\text{L}}(t) e^{w(Q-Z_c(t))} \left(\mathbb{E}\left[\delta_c(t)|H(t)\right] + \frac{\epsilon}{2}\right) \end{aligned} \quad (48)$$

where B is a finite constant defined as

$$\begin{aligned} B = |\mathcal{N}||\mathcal{C}| \left[\left(\mu_{\max}^{\text{out}} + d_{\max}\right)^2 + \left(A_{\max} + \mu_{\max}^{\text{in}}\right)^2\right] \\ + 2|\mathcal{N}||\mathcal{C}| d_{\max}^2 + |\mathcal{C}| \left[w(2\delta_{\max} + \epsilon) + e^{w\left(\nu_{\max} + \mu_{\max}^{\text{in}}\right)} + e^{wQ}\right]. \end{aligned}$$

The conditional expectation in (48) is with respect to the randomness of arrivals and the possibly random control decisions in slot t, given that the network state $H(t)$ summarizing the system history is observed. Equation (48) is the Lyapunov-drift-plus-penalty inequality, which we seek to minimize in every slot t after observing $H(t)$. Intuitively, minimizing the Lyapunov drift $\Delta(t)$ helps to stabilize the physical queues $Q_n^{(c)}(t)$ and the virtual queues $D_n^{(c)}(t)$ and $Z_c(t)$. In order to maximize the objective function (18), it is useful to minimize

$$\sum_{n,c} \theta_c \mathbb{E}\left[\varphi_n^{(c)}(t) \mid H(t)\right] - \sum_c \mathbb{E}\left[h_c\left(\nu_c(t)\right) \mid H(t)\right] \quad (49)$$

in every slot; the expected time average of (49) is closely related to the objective function $\sum_c h_c(\nu_c) - \sum_{n,c} \theta_c \varphi_n^{(c)}$ in (18). Minimizing the Lyapunov drift $\Delta(t)$ and minimizing (49) in a slot, however, conflict with each other; e.g., the former needs large $\varphi_n^{(c)}(t)$, and the latter desires small $\varphi_n^{(c)}(t)$. It is natural to minimize a weighted sum of $\Delta(t)$ and (49) in every slot, which is the left-hand side of (48) weighted by the parameter $V > 0$.

By isolating decision variables in (48), it is easy to verify that the UORA policy observes the current network state $H(t)$ and minimizes the right-hand side of (48) in every slot.

Appendix D

We establish the Lyapunov drift inequality in (48). Applying to (1) the fact that $[(a-b)^+ - c]^+ = (a - b - c)^+$ for nonnegative reals a, b, and c, and using Lemma 7 in Appendix G, we have

$$\begin{aligned} \frac{1}{2}\left(\left[Q_n^{(c)}(t+1)\right]^2 - \left[Q_n^{(c)}(t)\right]^2\right) \leq B_Q \\ - Q_n^{(c)}(t)\left[d_n^{(c)}(t) + \sum_{l \in L_{\text{out}}(n)} \mu_l^{(c)}(t) - A_n^{(c)}(t) - \sum_{l \in L_{\text{in}}(n)} \mu_l^{(c)}(t)\right] \end{aligned} \quad (50)$$

where $B_Q = \left(\mu_{\max}^{\text{out}} + d_{\max}\right)^2 + \left(A_{\max} + \mu_{\max}^{\text{in}}\right)^2$ is a finite constant. Equations (22) and (23) lead to $D_n^{(c)}(t+1) \leq \left[D_n^{(c)}(t) - \varphi_n^{(c)}(t)\right]^+ + d_n^{(c)}(t)$. Similarly, we obtain for queue $D_n^{(c)}(t)$

$$\begin{aligned} \frac{1}{2}\left(\left[D_n^{(c)}(t+1)\right]^2 - \left[D_n^{(c)}(t)\right]^2\right) \\ \leq B_D - D_n^{(c)}(t)\left(\varphi_n^{(c)}(t) - d_n^{(c)}(t)\right) \end{aligned} \quad (51)$$

where $B_D = 2d_{\max}^2 < \infty$.

Lemma 5: Given fixed constants $\epsilon > 0$ and $Q > 0$, we define

$$w \triangleq \left(\frac{\epsilon}{\delta_{\max}^2}\right) e^{-\epsilon/\delta_{\max}}. \quad (52)$$

Then

$$e^{w(Z_c(t+1)-Q)} - e^{w(Z_c(t)-Q)} \le e^{w\left(\nu_{\max}+\mu_{\max}^{\text{in}}\right)} - we^{w(Z_c(t)-Q)}\left[\delta_c(t)-\frac{\epsilon}{2}\right] \quad (53)$$

$$e^{w(Q-Z_c(t+1))} - e^{w(Q-Z_c(t))} \le e^{wQ} + we^{w(Q-Z_c(t))}\left[\delta_c(t)+\frac{\epsilon}{2}\right]. \quad (54)$$

Proof: Let $\delta_c(t) = \nu_c(t) - \sum_{l\in L_{\text{in}}(\sigma_c)} \mu_l^{(c)}(t)$ and $\delta_{\max} = \max(\nu_{\max}, \mu_{\max}^{\text{in}})$; we have $|\delta_c(t)| \le \delta_{\max}$ and $\nu_c(t) \le \delta_{\max}$. Equation (30) yields

$$Z_c(t+1) \le \begin{cases} Z_c(t) - \delta_c(t), & \text{if } Z_c(t) \ge \nu_{\max} \\ \nu_{\max} + \mu_{\max}^{\text{in}}, & \text{if } Z_c(t) < \nu_{\max}. \end{cases}$$

Since $w > 0$, we have

$$e^{wZ_c(t+1)} \le e^{w\left(\nu_{\max}+\mu_{\max}^{\text{in}}\right)} + e^{wZ_c(t)}e^{-w\delta_c(t)} \quad (55)$$

because the first term is bounded by the second term if $Z_c(t) < \nu_{\max}$, and is bounded by the third term otherwise. Multiplying both sides by $\exp(-wQ)$ and using $e^{w(\nu_{\max}+\mu_{\max}^{\text{in}}-Q)} \le e^{w(\nu_{\max}+\mu_{\max}^{\text{in}})}$, we have

$$e^{w(Z_c(t+1)-Q)} \le e^{w\left(\nu_{\max}+\mu_{\max}^{\text{in}}\right)} + e^{w(Z_c(t)-Q)}\left[1 - w\delta_c(t) + \frac{w^2\delta_{\max}^2}{2}e^{w\delta_{\max}}\right] \quad (56)$$

where the last term follows the Taylor expansion of $e^{-w\delta_c(t)}$. If we have

$$\frac{w^2\delta_{\max}^2}{2}e^{w\delta_{\max}} \le \frac{w\epsilon}{2} \quad (57)$$

then plugging (57) into (56) leads to (53). Indeed, by definition of w in (52), we get

$$\frac{w^2\delta_{\max}^2}{2}e^{w\delta_{\max}} = \frac{w\epsilon}{2}e^{w\delta_{\max}-\epsilon/\delta_{\max}} \le \frac{w\epsilon}{2}$$

which uses $w \le \epsilon/\delta_{\max}^2 \Rightarrow \exp(w\delta_{\max}) \le \exp(\epsilon/\delta_{\max})$.

Also, (30) leads to

$$Z_c(t+1) \ge Z_c(t) - \nu_c(t) + \sum_{l\in L_{\text{in}}(\sigma_c)} \widetilde{\mu}_l^{(c)}(t).$$

Define the event E: If $Q_n^{(c)}(t) \ge \mu_l^{\max}$ for all $l = (n, \sigma_c) \in \mathcal{L}$, i.e., all upstream nodes of the receiver σ_c have sufficient data to transmit, which yields $\widetilde{\mu}_l^{(c)}(t) = \mu_l^{(c)}(t)$ for all $l = (n, \sigma_c) \in \mathcal{L}$. It follows

$$Z_c(t+1) \ge \begin{cases} Z_c(t) - \delta_c(t), & \text{if event } E \text{ happens} \\ 0, & \text{otherwise} \end{cases} \quad (58)$$

where the second case follows that queue $Z_c(t)$ is always nonnegative. Similar to (55), using (58) and the Taylor expansion of $e^{w\delta_c(t)}$ we obtain

$$e^{-wZ_c(t+1)} \le 1 + e^{-wZ_c(t)}e^{w\delta_c(t)} \le 1 + e^{-wZ_c(t)}\left[1 + w\delta_c(t) + \frac{w^2\delta_{\max}^2}{2}e^{w\delta_{\max}}\right]. \quad (59)$$

Bounding the last term of (59) by (57) and multiplying the inequality by $\exp(wQ)$ yields (54). ∎

Next, by the definitions $1_c^{\text{R}}(t) = 1_{[Z_c(t)\ge Q]}$ and $1_c^{\text{L}}(t) = 1 - 1_c^{\text{R}}(t)$, for any real number a we have

$$ae^{w(Z_c(t)-Q)} = a\left(1_c^{\text{R}}(t) + 1_c^{\text{L}}(t)\right)e^{w(Z_c(t)-Q)} \le |a| + a1_c^{\text{R}}(t)e^{w(Z_c(t)-Q)}. \quad (60)$$

Similarly, we have

$$ae^{w(Q-Z_c(t))} \le |a| + a1_c^{\text{L}}(t)e^{w(Q-Z_c(t))}. \quad (61)$$

Plugging $a = -w(\delta_c(t)-\epsilon/2)$ into (60) and $a = w(\delta_c(t)+\epsilon/2)$ into (61), inequalities (53) and (54) lead to

$$e^{w(Z_c(t+1)-Q)} - e^{w(Z_c(t)-Q)} \le e^{w\left(\nu_{\max}+\mu_{\max}^{\text{in}}\right)} + w\left(\delta_{\max}+\frac{\epsilon}{2}\right) - w1_c^{\text{R}}(t)e^{w(Z_c(t)-Q)}\left[\delta_c(t)-\frac{\epsilon}{2}\right] \quad (62)$$

$$e^{w(Q-Z_c(t+1))} - e^{w(Q-Z_c(t))} \le e^{wQ} + w\left(\delta_{\max}+\frac{\epsilon}{2}\right) + w1_c^{\text{L}}(t)e^{w(Q-Z_c(t))}\left[\delta_c(t)+\frac{\epsilon}{2}\right]. \quad (63)$$

Summing (50) and (51) over $n \ne \sigma_c$ for each $c \in \mathcal{C}$, summing (62) and (63) over $c \in \mathcal{C}$, and taking conditional expectation, we have

$$\begin{aligned}\Delta(t) \le B &+ \sum_{nc} Q_n^{(c)}(t)\lambda_n^{(c)} - \sum_{nc} Q_n^{(c)}(t)\mathbb{E}\left[d_n^{(c)}(t) \mid H(t)\right] \\ &- \sum_{nc} Q_n^{(c)}(t)\mathbb{E}\left[\sum_{l\in L_{\text{out}}(n)} \mu_l^{(c)}(t) - \sum_{l\in L_{\text{in}}(n)} \mu_l^{(c)}(t) \mid H(t)\right] \\ &- \sum_{nc} D_n^{(c)}(t)\mathbb{E}\left[\varphi_n^{(c)}(t) - d_n^{(c)}(t) \mid H(t)\right] \\ &- w\sum_c 1_c^{\text{R}}(t)e^{w(Z_c(t)-Q)}\left(\mathbb{E}\left[\delta_c(t) \mid H(t)\right] - \frac{\epsilon}{2}\right) \\ &+ w\sum_c 1_c^{\text{L}}(t)e^{w(Q-Z_c(t))}\left(\mathbb{E}\left[\delta_c(t) \mid H(t)\right] + \frac{\epsilon}{2}\right) \end{aligned} \quad (64)$$

where B is a finite constant defined as

$$B = |\mathcal{N}||\mathcal{C}|(B_Q + B_D) + |\mathcal{C}|\left[w(2\delta_{\max}+\epsilon) + e^{w\left(\nu_{\max}+\mu_{\max}^{\text{in}}\right)} + e^{wQ}\right]. \quad (65)$$

The constants B_Q and B_D are defined right after (50) and (51), respectively. Adding to both sides of (64)

$$-V\sum_c \mathbb{E}\left[h_c\left(\nu_c(t)\right) \mid H(t)\right] + V\sum_{nc}\theta_c\mathbb{E}\left[\varphi_n^{(c)}(t) \mid H(t)\right]$$

yields (48).

Appendix E
Proof of Lemma 3

The boundedness of the queues $Q_n^{(c)}(t)$ and $D_n^{(c)}(t)$ follows the proof of Lemma 2. We show the queues $Z_c(t)$ are deterministically bounded by induction. Suppose $Z_c(t) \le Z_{\max}^{(c)}$, which holds at $t = 0$ because we let $Z_c(0) = 0$. Consider two cases.

1) If $Z_c(t) \le Z_{\max} - \mu_{\max}^{\text{in}}$, then by (30) we have $Z_c(t+1) \le Z_c(t) + \mu_{\max}^{\text{in}} \le Z_{\max}^{(c)}$.

2) If $Z_c(t) > Z_{\max}^{(c)} - \mu_{\max}^{\text{in}}$, then from (35) we obtain

$$Z_c(t) - Q > Z_{\max}^{(c)} - \mu_{\max}^{\text{in}} - Q \overset{(a)}{=} \frac{1}{w}\log\left(\frac{V\theta_c + 2d_{\max}}{w}\right) \geq 0 \tag{66}$$

where the last inequality follows our choice of V satisfying $V\theta_c + 2d_{\max} \geq w$ in Section V-C. As a result, in (32) we have

$$Q_{\sigma_c}^{(c)}(t) = we^{w(Z_c(t)-Q)} > V\theta_c + 2d_{\max} = Q_{\max}^{(c)}$$

where the inequality follows (a) in (66). Since all queues $Q_n^{(c)}(t)$ for $n \neq \sigma_c$ are deterministically bounded by $Q_{\max}^{(c)}$, we obtain $Q_{\sigma_c}^{(c)}(t) > Q_n^{(c)}(t)$ for all nodes n such that $(n, \sigma_c) \in \mathcal{L}$. Consequently, the UORA policy does not transmit any class c packets over the links (n, σ_c), and the virtual queue $Z_c(t)$ has no arrivals. Therefore, $Z_c(t+1) \leq Z_c(t) \leq Z_{\max}^{(c)}$.

We conclude that $Z_c(t+1) \leq Z_{\max}^{(c)}$. The proof is complete.

Appendix F
Proof of Theorem 2

Let the throughput vector (r_c^*) and the flow variables $(q_n^{(c)*}; f_l^{(c)*})$ be the optimal solution to the utility maximization problem (11)–(14); this optimal solution exists, but is unknown. Consider a "genie" stationary policy π^* that observes the network state $H(t)$ and chooses deterministically in slot t.

1) $\varphi_n^{(c)*}(t) = q_n^{(c)*}$ and $d_n^{(c)*}(t) = q_n^{(c)*}$ for all $H(t)$.
2) $\mu_l^{(c)*}(t) = f_l^{(c)*}$ for all $H(t)$.
3) $\nu_c^*(t) = r_c^* + (\epsilon/2)$ whenever $Z_c(t) \geq Q$, and $\nu_c^*(t) = r_c^*$ otherwise.

The control variables $\varphi_n^{(c)*}(t)$ and $d_n^{(c)*}(t)$ take values in $[0, d_{\max}]$, which includes the interval $[0, A_{\max} + \mu_{\max}^{\text{in}}]$ by Assumption 1. The optimal queue overflow rate $q_n^{(c)*}$ is at most $A_{\max} + \mu_{\max}^{\text{in}}$ and thus is a feasible choice for $\varphi_n^{(c)*}(t)$ and $d_n^{(c)*}(t)$. The choice of $\mu_l^{(c)*}(t)$ is feasible because $\{f_l^{(c)*}\}$ satisfy the link capacity constraints (7). The choices of $\nu_c^*(t) \in [0, \nu_{\max}]$ are feasible because we assume $\nu_{\max} \geq \max_{c\in\mathcal{C}} r_c^* + \epsilon/2$.

Under the stationary policy π^*, the equalities in (8) and (12) yield $\sum_{l\in L_{\text{in}}(\sigma_c)} \mu_l^{(c)*}(t) = r_c^*$ in every slot. For all network states $H(t)$ that satisfy $Z_c(t) \geq Q$, we have

$$\begin{aligned}
&\mathbb{E}\left[\delta_c^*(t) \mid H(t)\right] \\
&= \mathbb{E}\left[\nu_c^*(t) - \sum_{l\in L_{\text{in}}(\sigma_c)} \mu_l^{(c)*}(t) \mid Z_c(t) \geq Q\right] \\
&= \mathbb{E}\left[\nu_c^*(t) - r_c^* \mid Z_c(t) \geq Q\right] = r_c^* + \frac{\epsilon}{2} - r_c^* = \frac{\epsilon}{2}.
\end{aligned}$$

For all network states $H(t)$ satisfying $Z_c(t) < Q$, we have

$$\mathbb{E}\left[\delta_c^*(t) \mid H(t)\right] = \mathbb{E}\left[\delta_c^*(t) \mid Z_c(t) < Q\right] = r_c^* - r_c^* = 0.$$

Define $\Phi(\pi, H(t))$ as the right-hand side of (48) in slot t evaluated under policy π. Since the UORA policy minimizes the right-hand side of (48) in every slot, we have $\Phi(\text{UORA}, H(t)) \leq \Phi(\pi^*, H(t))$ for all t and all network states $H(t)$. Thus, the inequality (48) under the UORA policy satisfies

$$\begin{aligned}
&\Delta(t) - V\sum_c \mathbb{E}[h_c(\nu_c(t)) \mid H(t)] + V\sum_{n,c}\theta_c\,\mathbb{E}\left[\varphi_n^{(c)}(t) \mid H(t)\right] \\
&\leq \Phi\left(\text{UORA}, H(t)\right) \leq \Phi\left(\pi^*, H(t)\right) \\
&\overset{(a)}{\leq} B + |\mathcal{C}|\frac{w\epsilon}{2}e^{wQ} - V\sum_c h_c\left(\nu_c^*(t)\right) + V\sum_{n,c}\theta_c q_n^{(c)*} \\
&\quad - \sum_{nc} Q_n^{(c)}(t)\left(\sum_{l\in L_{\text{out}}(n)} f_l^{(c)*} + q_n^{(c)*} - \lambda_n^{(c)} - \sum_{l\in L_{\text{in}}(n)} f_l^{(c)*}\right) \\
&\overset{(b)}{=} B_1 - V\sum_c h_c\left(\nu_c^*(t)\right) + V\sum_{n,c}\theta_c q_n^{(c)*}
\end{aligned} \tag{67}$$

where we define

$$B_1 = B + \frac{1}{2}|\mathcal{C}|w\epsilon e^{wQ} \tag{68}$$

and B is given in (65). In (67), the inequality (a) uses that

$$\begin{aligned}
&- w\sum_c 1_c^{\text{R}}(t) e^{w(Z_c(t)-Q)}\left(\mathbb{E}\left[\delta_c^*(t) \mid H(t)\right] - \frac{\epsilon}{2}\right) \\
&+ w\sum_c 1_c^{\text{L}}(t) e^{w(Q-Z_c(t))}\left(\mathbb{E}\left[\delta_c^*(t) \mid H(t)\right] + \frac{\epsilon}{2}\right) \\
&\leq \frac{w\epsilon}{2}\sum_c 1_c^{\text{L}}(t) e^{w(Q-Z_c(t))} \leq \frac{w\epsilon}{2}|\mathcal{C}|e^{wQ}
\end{aligned}$$

and the equality (b) is because the flow variables $f_l^{(c)*}$ and $q_n^{(c)*}$ satisfy the flow conservation constraints (6).

The utility functions g_c are assumed to have bounded derivatives with $g_c'(x) \leq g_c'(0)$ for all $x \geq 0$. Thus, the utility functions h_c have bounded derivatives with $|h_c'(x)| \leq g_c'(0) + \theta_c$ for all $x \geq 0$. From $\nu_c^*(t) \in \{r_c^*, r_c^* + \epsilon/2\}$, we can show

$$h_c\left(\nu_c^*(t)\right) \geq h_c(r_c^*) - \frac{\epsilon}{2}\left(g_c'(0) + \theta_c\right). \tag{69}$$

Indeed, if $\nu_c^*(t) = r_c^*$, then (69) holds. If $\nu_c^*(t) = r_c^* + \epsilon/2$, then by mean-value theorem there exists $x \in (r_c^*, r_c^* + \epsilon/2)$ such that

$$\begin{aligned}
h_c\left(\nu_c^*(t)\right) &= h_c\left(r_c^*\right) + \left(\nu_c^*(t) - r_c^*\right) h_c'(x) \\
&\geq h_c\left(r_c^*\right) - \frac{\epsilon}{2}\left(g_c'(0) + \theta_c\right).
\end{aligned}$$

Using (69) in (67) yields

$$\begin{aligned}
&\Delta(t) - V\sum_c \mathbb{E}[h_c(\nu_c(t)) \mid H(t)] + V\sum_{n,c}\theta_c\,\mathbb{E}\left[\varphi_n^{(c)}(t) \mid H(t)\right] \\
&\leq B_1 - V\sum_c h_c\left(r_c^*\right) + V\sum_{n,c}\theta_c q_n^{(c)*} + \frac{V\epsilon}{2}\sum_c\left(g_c'(0) + \theta_c\right) \\
&= B_1 - V\sum_c g_c\left(r_c^*\right) + V\sum_{n,c}\theta_c\lambda_n^{(c)} + \frac{V\epsilon}{2}\sum_c\left(g_c'(0) + \theta_c\right)
\end{aligned} \tag{70}$$

where the last equality uses $h_c(x) = g_c(x) - \theta_c x$ and $r_c^* = \sum_n(\lambda_n^{(c)} - q_n^{(c)*})$ in (12). The next lemma is useful.

Lemma 6: Define $\widetilde{\nu}_c(t) = \min[Z_c(t), \nu_c(t)]$ as the virtual data departing the queue $Z_c(t)$ in slot t. If $Q \geq \nu_{\max}$ and $\theta_c \geq g_c'(0)$, then $\nu_c(t) = \widetilde{\nu}_c(t)$ for all t under the UORA policy.

Proof of Lemma 6: The value of $\nu_c(t)$ is the solution to the convex program (33)–(34). If $Z_c(t) \geq Q \geq \nu_{\max}$, then we have $Z_c(t) \geq \nu_{\max} \geq \nu_c(t)$ and $\widetilde{\nu}_c(t) = \nu_c(t)$. If $Z_c(t) < Q$, the problem (33)–(34) reduces to

$$\text{maximize} \quad V h_c\left(\nu_c(t)\right) - w\nu_c(t)e^{w(Q-Z_c(t))} \tag{71}$$

$$\text{subject to} \quad 0 \leq \nu_c(t) \leq \nu_{\max}. \tag{72}$$

Since the function $h_c(x) = g_c(x) - \theta_c x$ is decreasing due to $\theta_c \geq g_c'(0)$, the optimal solution to (33)–(34) is $\nu_c(t) = 0$. Since $0 \leq \widetilde{\nu}_c(t) \leq \nu_c(t) = 0$, we have $\nu_c(t) = \widetilde{\nu}_c(t) = 0$. ∎

From Lemma 6, we use $h_c(\nu_c(t)) = h_c(\widetilde{\nu}_c(t))$ in (70) and rearrange terms to get

$$\Delta(t) - V\sum_c \mathbb{E}\left[h_c\left(\widetilde{\nu}_c(t)\right)|H(t)\right] + V\sum_{n,c}\theta_c\mathbb{E}\left[\varphi_n^{(c)}(t)|H(t)\right]$$
$$\leq B_1 + \frac{V\epsilon}{2}\sum_c\left(g_c'(0)+\theta_c\right) - V\sum_c g_c\left(r_c^*\right) + V\sum_{n,c}\theta_c\lambda_n^{(c)}. \tag{73}$$

Define the time average $\overline{\widetilde{\nu}_c(t)} \triangleq (1/t)\sum_{\tau=0}^{t-1}\mathbb{E}[\widetilde{\nu}_c(\tau)]$. Define $\overline{\varphi_n^{(c)}(t)}$, $\overline{\widetilde{d}_n^{(c)}(t)}$, and $\overline{\widetilde{\mu}_l^{(c)}(t)}$ similarly. Taking expectation and time average over $\tau \in \{0,\ldots,t-1\}$ in (73), dividing by V, rearranging terms, and applying Jensen's inequality to the functions h_c, we get

$$\sum_c h_c\left(\overline{\widetilde{\nu}_c(t)}\right) + \sum_{n,c}\theta_c\left(\lambda_n^{(c)} - \overline{\varphi_n^{(c)}(t)}\right)$$
$$\geq \sum_c g_c\left(r_c^*\right) - \frac{B_1}{V} - \frac{\mathbb{E}\left[L\left(H(0)\right)\right]}{Vt} - \frac{\epsilon}{2}\sum_c\left(g_c'(0)+\theta_c\right). \tag{74}$$

Adding and subtracting $\sum_c \theta_c\overline{\widetilde{\nu}_c(t)}$ at the left-hand side of (74) and using the definition of h_c, we get

$$\sum_c g_c\left(\overline{\widetilde{\nu}_c(t)}\right) + \sum_c\theta_c\left[\sum_n\lambda_n^{(c)} - \sum_n\overline{\varphi_n^{(c)}(t)} - \overline{\widetilde{\nu}_c(t)}\right]$$
$$\geq \sum_c g_c\left(r_c^*\right) - \frac{B_1}{V} - \frac{\mathbb{E}\left[L\left(H(0)\right)\right]}{Vt} - \frac{\epsilon}{2}\sum_c\left(g_c'(0)+\theta_c\right). \tag{75}$$

From (22) and (30), we have

$$D_n^{(c)}(t+1) \geq D_n^{(c)}(t) - \varphi_n^{(c)}(t) + \widetilde{d}_n^{(c)}(t)$$
$$Z_c(t+1) = Z_c(t) - \widetilde{\nu}_c(t) + \sum_{l\in L_{\text{in}}(\sigma_c)}\widetilde{\mu}_l^{(c)}(t).$$

Taking expectation and time average yields

$$\overline{\varphi_n^{(c)}(t)} \geq \overline{\widetilde{d}_n^{(c)}(t)} - \mathbb{E}\left[D_n^{(c)}(t)\right]/t \tag{76}$$

$$\overline{\widetilde{\nu}_c(t)} = \sum_{l\in L_{\text{in}}(\sigma_c)}\overline{\widetilde{\mu}_l^{(c)}(t)} + \mathbb{E}\left[Z_c(0) - Z_c(t)\right]/t. \tag{77}$$

By the law of flow conservation, the sum of exogenous arrival rates is equal to the sum of delivered throughput, time averages of dropped packets, and queue growth rates. In other words, we have for each class $c \in \mathcal{C}$ and for all slots t

$$\sum_n\lambda_n^{(c)} = \sum_n\overline{\widetilde{d}_n^{(c)}(t)} + \sum_{l\in L_{\text{in}}(\sigma_c)}\overline{\widetilde{\mu}_l^{(c)}(t)} + \frac{1}{t}\sum_n\mathbb{E}\left[Q_n^{(c)}(t)\right]. \tag{78}$$

Combining (76)–(78) yields

$$\sum_n\lambda_n^{(c)} - \sum_n\overline{\varphi_n^{(c)}(t)} - \overline{\widetilde{\nu}_c(t)}$$
$$\leq \frac{1}{t}\mathbb{E}\left[Z_c(t) - Z_c(0)\right] + \frac{1}{t}\sum_n\mathbb{E}\left[D_n^{(c)}(t) + Q_n^{(c)}(t)\right]. \tag{79}$$

Using the boundedness of queues $Q_n^{(c)}(t)$, $D_n^{(c)}(t)$, and $Z_c(t)$ in Lemma 3 and the continuity of g_c, we obtain from (77) and (79) that

$$\lim_{t\to\infty} g_c\left(\overline{\widetilde{\nu}_c(t)}\right) = g_c\left(\lim_{t\to\infty}\sum_{l\in L_{\text{in}}(\sigma_c)}\overline{\widetilde{\mu}_l^{(c)}(t)}\right) = g_c(\overline{r_c}) \tag{80}$$

$$\lim_{t\to\infty}\left(\sum_n\lambda_n^{(c)} - \sum_n\overline{\varphi_n^{(c)}(t)} - \overline{\widetilde{\nu}_c(t)}\right) \leq 0 \tag{81}$$

where the last equality of (80) uses the definition in (27). Taking a limit of (75) as $t\to\infty$ and using (80) and (81), we obtain

$$\sum_c g_c(\overline{r_c}) \geq \sum_c g_c\left(r_c^*\right) - \frac{B_1}{V} - \frac{\epsilon}{2}\sum_c\left(g_c'(0)+\theta_c\right)$$

where the constant B_1, defined in (68), is

$$B_1 = |\mathcal{N}|\mathcal{C}|\left[\left(\mu_{\max}^{\text{out}} + d_{\max}\right)^2 + \left(A_{\max} + \mu_{\max}^{\text{in}}\right)^2 + 2d_{\max}^2\right]$$
$$+|\mathcal{C}|\left[w(2\delta_{\max}+\epsilon) + e^{w\left(\nu_{\max}+\mu_{\max}^{\text{in}}\right)} + \frac{w\epsilon}{2}e^{wQ} + e^{wQ}\right]. \tag{82}$$

The proof is complete.

Appendix G

Lemma 7: If a queue process $\{Q(t)\}$ satisfies

$$Q(t+1) \leq \left(Q(t) - b(t)\right)^+ + a(t) \tag{83}$$

where $a(t)$ and $b(t)$ are nonnegative bounded random variables with $a(t) \leq a_{\max}$ and $b(t) \leq b_{\max}$, then there exists a positive constant B such that $(1/2)(Q^2(t+1) - Q^2(t)) \leq B - Q(t)(b(t) - a(t))$.

Proof of Lemma 7: Squaring both sides of (83) yields

$$Q^2(t+1) \leq \left(Q(t) - b(t)\right)^2 + a^2(t) + 2Q(t)a(t)$$
$$\leq Q^2(t) + B' - 2Q(t)\left(b(t) - a(t)\right)$$

where B' is a finite constant satisfying $B' \geq a_{\max}^2 + b_{\max}^2$. Dividing the above by two and defining $B = B'/2$ complete the proof. ∎

References

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

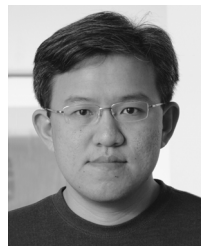
[2] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res.*, vol. 49, pp. 237–252, 1998.

[3] F. P. Kelly, "Charging and rate control for elastic traffic," *Eur. Trans. Telecommun.*, vol. 8, pp. 33–37, 1997.

[4] S. H. Low and D. E. Lapsley, "Optimization flow control—I: Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

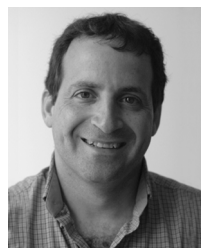
[5] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Syst.*, vol. 50, no. 4, pp. 401–457, 2005.

[6] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.
[7] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1514–1524, Aug. 2006.
[8] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1333–1344, Dec. 2007.
[9] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. IEEE CDC*, Dec. 2004, pp. 1484–1489.
[10] R. K. C. Chang, "Defending against flooding-based distributed denial-of-service attacks: A tutorial," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 42–51, Oct. 2002.
[11] A. Srivastava, B. B. Gupta, A. Tyagi, A. Sharma, and A. Mishra, "A recent survey on DDoS attacks and defense mechanisms," in *Advances in Parallel Distributed Computing*, ser. Communications in Computer and Information Science. Berlin, Germany: Springer, 2011, vol. 203, pp. 570–580.
[12] J. Borland, "Net video not yet ready for prime time," Feb. 1999 [Online]. Available: http://news.cnet.com/2100-1033-221271.html
[13] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *Comput. Commun. Rev.*, vol. 32, pp. 62–73, 2002.
[14] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.
[15] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Francisco, CA, USA: Morgan & Claypool, 2010.
[16] L. Georgiadis and L. Tassiulas, "Optimal overload response in sensor networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2684–2696, Jun. 2006.
[17] C. W. Chan, M. Armony, and N. Bambos, "Fairness in overloaded parallel queues," working paper, 2011.
[18] D. Shah and D. Wischik, "Fluid models of congestion collapse in overloaded switched networks," *Queueing Syst.*, vol. 69, pp. 121–143, 2011.
[19] R. Egorova, S. Borst, and B. Zwart, "Bandwidth-sharing in overloaded networks," in *Proc. CISS*, Princeton, NJ, USA, Mar. 2008, pp. 36–41.
[20] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 29–42, Feb. 2005.
[21] C. W. Tan, D.-M. Chiu, J. C. S. Lui, and D. K. Y. Yau, "A distributed throttling approach for handling high bandwidth aggregates," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 7, pp. 983–995, Jul. 2007.
[22] S. Chen and Q. Song, "Perimeter-based defense against high bandwidth DDoS attacks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 6, pp. 526–537, Jun. 2005.
[23] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.
[24] M. J. Neely, "Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1489–1501, Aug. 2006.



Chih-ping Li received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 2001, and the M.S. and Ph.D. degrees from the University of Southern California, Los Angeles, CA, USA, in 2005 and 2011, respectively, all in electrical engineering.

Since 2011, he has been a Postdoctoral Associate with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include stochastic control and resource allocation in communication networks, wireless networks, and queueing systems.



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

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

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