Receiver-Based Flow Control for Networks in Overload

Chih-ping Li and Eytan Modiano

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

***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 novel 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.**

I. Introduction

The idea behind flow control in data networks is to regulate the source rates in order to prevent network overload, and provide fair allocation of resources. In recent years, extensive research has been devoted to the problem of network utility maximization, with the objective of optimizing network resource allocation through a combination of source-based flow control, routing, and scheduling. The common theme is to assign a utility, as a function of the source rate, to a flow specified by a source-destination pair, and formulate an optimization problem that maximizes the sum of utilities (e.g., see [1]–[9]). The optimal network control policy is revealed as the algorithm that solves the utility maximization problem.

Source-based flow control implicitly requires all sources to react properly to congestion signals such as packet loss or delay. Congestion-insensitive traffic, however, is pervasive in modern networks. UDP-based applications such as video streaming or VoIP are increasingly popular and do not respond to congestion. Moreover, greedy or malicious users can inject excessive data into the network to either boost self-performance or bring down high-profile websites. In these circumstances, the network can be temporarily overloaded and congestion-aware applications, e.g., TCP-based traffic, may be adversely affected or even starved. In this context, source-based flow control may not be adequate.

There are other scenarios in which optimizing the source rates of data flows on a per-flow basis is ineffective. The Internet nowadays is vulnerable to distributed denial of service (DDoS) attacks [10], [11], in which an attacker creates a large number of flows with different source addresses to overwhelm prominent websites. Similarly, in a multi-player online game, thousands of users require continuous access to the game servers. There are also occasions in which a large number of users seek access to a website that broadcasts live events [12]; this is the so-called *flash crowd* phenomenon [13]. In these situations, source-based flow control is ineffective because each individual flow uses little bandwidth, but their aggregate traffic can lead to severe link congestion near the service provider, and may starve other unrelated users in the network. A flow control scheme that can optimize a utility assigned to a collection of flows as opposed to optimizing the sum of per-flow utilities can be used to cope with such scenarios. Moreover, in order to cope with uncooperative flows, it is necessary to relocate the flow control functionality from untrusted network hosts to more secure ones, such as the web servers that provide service at the receiver end.

In this paper, we develop such *receiver-based flow control* policies using tools from stochastic network optimization [14], [15]. Our main contributions are three-fold. First, we formulate a utility maximization problem that can assign utilities to an aggregate 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 corresponding achievable throughput region in terms of queue overflow rates. Third, using a novel decomposition of the utility functions, we design a network control policy consisting of: (i) a set of flow controllers at the receivers; (ii) packet dropping mechanism at internal nodes; and (iii) back-pressure routing at 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 receiver. To deal with data that cannot be delivered due to network overload, we design a threshold-based packet dropping mechanism that discards data whenever

This work was supported by DTRA grants HDTRA1-07-1-0004 and HDTRA-09-1-005, ARO Muri grant number W911NF-08-1-0238, and NSF grant CNS-1116209.

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

queues grow beyond certain thresholds. Surprisingly, we show that our threshold-based packet dropping scheme, without the use of flow control, is sufficient to maximize the weighted sum throughput. Moreover, the combined flow control and packet dropping mechanism has the following properties: (i) It works with finite-size buffers. (ii) 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. (iii) It does not require the knowledge of arrival rates and therefore is robust to time-varying arrival rates that can go far beyond the network's stability region. In addition, our control policy can be implemented only in parts of the network that include the receivers, treating the rest of the network as exogenous data sources (see Fig. 1 for an example), and thus might be an attractive flow control solution for web servers.

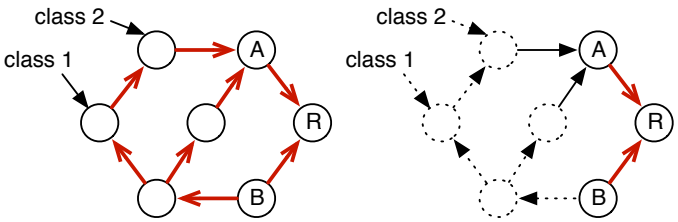


Fig. 1. Our receiver-based policy can be implemented in the whole network on the left, or implemented only at nodes A, B, and R on the right, where R is the only receiver. The rest of the network on the right may be controlled by another network operator or follow a different network control scheme.

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 lossless networks in 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], [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 DDoS attacks have been proposed in [20]–[22].

An outline of the paper is as follows. The network model is given in Section II. We formulate the utility maximization problem and characterize the achievable throughput region 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 general utility maximization problem. 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 for service and are categorized into a collection $\mathcal{C}$ of classes. The definition of a data class is quite flexible except that we assume packets in a class $c \in \mathcal{C}$ have a shared destination d_c. For example, each class can simply be a flow specified by a source-destination pair. Alternatively, computing-on-demand services in the cloud such as Amazon (Elastic Compute Cloud; EC2) or Google (App Engine) can 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 to the service provider. 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 $A_{d_c}^{(c)}(t) = 0$ for all t. We assume $A_n^{(c)}(t)$ are independent across classes c and nodes $n \neq d_c$, and are i.i.d. over slots with mean $\mathbb{E}\big[A_n^{(c)}(t)\big] = \lambda_n^{(c)}$.

In the network, packets are relayed toward the destinations via dynamic routing and link rate allocation decisions. Each link $(n, m) \in \mathcal{L}$ is used to transmit data from node n to node m and has a fixed capacity $\mu_{\max}^{nm}$ (in units of packets/slot).[1] Under a given control policy, let $\mu_{nm}^{(c)}(t)$ be the service rate allocated to class c data over link (n, m) in slot t. The service rates must satisfy the link capacity constraints

$$\sum_{c \in \mathcal{C}} \mu_{nm}^{(c)}(t) \leq \mu_{\max}^{nm}, \quad \text{for all } t \text{ and all links } (n, m).$$

At a node $n \neq d_c$, class c packets that arrive but not yet transmitted are stored in a queue; we let $Q_n^{(c)}(t)$ be the backlog of class c packets at node n at time t. We assume initially $Q_n^{(c)}(0) = 0$ for all n and c. Destinations do not buffer packets and we have $Q_{d_c}^{(c)}(t) = 0$ 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 buffers. To resolve potential network congestion due to traffic overload, a queue $Q_n^{(c)}(t)$, after transmitting data to neighboring nodes in a slot, discards $d_n^{(c)}(t)$ packets from the remaining backlog at the end of the slot. The drop rate $d_n^{(c)}(t)$ is a function of the control policy to be described later and takes values in $[0, d_{\max}]$ for some finite $d_{\max}$. The queue $Q_n^{(c)}(t)$ evolves over

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

slots according to

$$Q_n^{(c)}(t+1) \le \left[\left(Q_n^{(c)}(t) - \sum_b \mu_{nb}^{(c)}(t)\right)^+ - d_n^{(c)}(t)\right]^+ + A_n^{(c)}(t) + \sum_a \mu_{an}^{(c)}(t), \quad \forall c,\ \forall n \neq d_c, \tag{1}$$

where $(\cdot)^+ \triangleq \max(\cdot, 0)$. This inequality is due to the fact that endogenous arrivals may be less than the allocated rate $\sum_a \mu_{an}^{(c)}(t)$ 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

$$\mu_{\max}^{\text{in}} \triangleq \max_{n\in\mathcal{N}} \sum_{a:(a,n)\in\mathcal{L}} \mu_{\max}^{an}, \quad \mu_{\max}^{\text{out}} \triangleq \max_{n\in\mathcal{N}} \sum_{b:(n,b)\in\mathcal{L}} \mu_{\max}^{nb}. \tag{2}$$

Throughout the paper, we use the following assumption.

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

From (1), the sum $A_{\max} + \mu_{\max}^{\text{in}}$ is the largest amount of data that can arrive at a node in a slot; therefore it 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 prevent the queues from blowing up.

III. Problem Formulation

We assign to each class $c \in \mathcal{C}$ a utility function $g_c(\cdot)$. 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), where r_c is the aggregate throughput of class c received by the destination d_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

$$\text{maximize} \quad \sum_{c\in\mathcal{C}} g_c(r_c) \tag{3}$$

$$\text{subject to} \quad (r_c) \in \Lambda_{\boldsymbol{\lambda}}, \tag{4}$$

where the region $\Lambda_{\boldsymbol{\lambda}}$ is presented later in Lemma 1. We assume all $g_c(\cdot)$ functions are concave, increasing, and continuously differentiable. For ease of exposition, we also assume the functions $g_c(\cdot)$ have bounded derivatives such that $|g_c'(x)| \le m_c$ for all $x \ge 0$, where m_c are finite constants.[2]

As an example, consider the tree network in Fig. 2 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 sending or requesting information from node R. If class 1 traffic is congestion-insensitive and overloads the network, without proper flow control class 2 and 3 will be starved due to the presence of class 1. A utility maximization problem here is to solve

$$\text{maximize} \quad g_1(r_{1A} + r_{1C}) + g_2(r_{2B}) + g_3(r_{3D}) \tag{5}$$

$$\text{subject to} \quad (r_{1A}, r_{1C}, r_{2B}, r_{3D}) \text{ feasible}, \tag{6}$$

where r_{1A} denotes the throughput of class 1 data originating from A; r_{1C}, r_{2B}, and r_{3D} are defined similarly. Note that this utility maximization (5)-(6) is very different from, and generalizes, the traditional per-flow-based utility maximization.

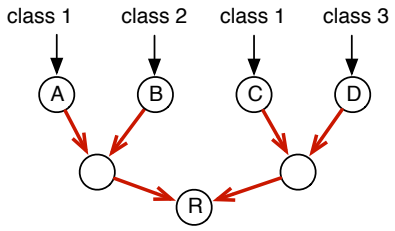


Fig. 2. A tree network with three classes of traffic.

A. Achievable Throughput Region

The next lemma characterizes the set $\Lambda_{\boldsymbol{\lambda}}$ of all achievable throughput vectors in (4).

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). Then $(r_c) \in \Lambda_{\boldsymbol{\lambda}}$ if and only if there exist flow variables $\{f_{ab}^{(c)} \ge 0 \mid c \in \mathcal{C}, (a,b) \in \mathcal{L}\}$ and queue overflow variables $\{q_n^{(c)} \ge 0 \mid c \in \mathcal{C},\ n \neq d_c\}$ such that

$$\lambda_n^{(c)} + \sum_a f_{an}^{(c)} = q_n^{(c)} + \sum_b f_{nb}^{(c)} \quad \forall\, c \in \mathcal{C},\ n \neq d_c, \tag{7}$$

$$\sum_c f_{ab}^{(c)} \le \mu_{\max}^{ab} \quad \forall\, (a,b) \in \mathcal{L}, \tag{8}$$

$$r_c \le \sum_a f_{ad_c}^{(c)} = \sum_n \lambda_n^{(c)} - \sum_n q_n^{(c)} \quad \forall\, c \in \mathcal{C}. \tag{9}$$

In other words,

$$\Lambda_{\boldsymbol{\lambda}} = \left\{ (r_c) \;\middle|\; \text{(7)-(9) hold and } f_{ab}^{(c)} \ge 0, q_n^{(c)} \ge 0 \right\}.$$

Proof of Lemma 1: See technical report [23]. ■

In Lemma 1, equation (7) 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 (8) is the link capacity constraint. The equality in (9) 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 (Theorem 1, [24])**.** The capacity region Λ of the network is the set of all arrival rate matrices $(\lambda_n^{(c)})$ for which there exists nonnegative flow variables $f_{ab}^{(c)}$ such that

$$\lambda_n^{(c)} + \sum_a f_{an}^{(c)} \le \sum_b f_{nb}^{(c)}, \quad \forall\, c \in \mathcal{C},\ n \neq d_c, \tag{10}$$

[2]Utility functions $g_c(x)$ that have unbounded derivatives as $x \to 0$ can be approximated by those with bounded derivatives. For example, we may approximate the proportional fairness function $\log(x)$ by $\log(x+\xi)$ for some small $\xi > 0$.

$$\sum_c f_{ab}^{(c)} \le \mu_{\max}^{ab}, \quad \forall\,(a,b) \in \mathcal{L}. \tag{11}$$

Corollary 1. An arrival rate matrix $(\lambda_n^{(c)})$ lies in the network capacity region Λ if there exist flow variables $f_{ab}^{(c)} \ge 0$ such that flow conservation constraints (7) and link capacity constraints (8) hold with $q_n^{(c)} = 0$ for all n and c.

Corollary 1 shows that $(\lambda_n^{(c)})$ is achievable if and only if there exists a control policy yielding zero queue overflow rates. In this case the throughput of class c is $r_c = \sum_n \lambda_n^{(c)}$.

We remark that the solution to the utility maximization (3)-(4) activates the linear constraints (9); thus the problem (3)-(4) is equivalent to

$$\text{maximize} \quad \sum_{c\in\mathcal{C}} g_c(r_c) \tag{12}$$

$$\text{subject to} \quad r_c = \sum_n \lambda_n^{(c)} - \sum_n q_n^{(c)}, \quad \forall\, c \in \mathcal{C} \tag{13}$$

$$\text{(7) and (8) hold} \tag{14}$$

$$f_{ab}^{(c)} \ge 0,\ q_n^{(c)} \ge 0,\ \forall (a,b) \in \mathcal{L},\ \forall n, c. \tag{15}$$

Let (r_c^*) be the optimal throughput vector that solves (12)-(15). If the arrival rate matrix $(\lambda_n^{(c)})$ is in the network capacity region Λ, the optimal throughput is $r_c^* = \sum_n \lambda_n^{(c)}$ from Corollary 1. Otherwise, we have $r_c^* = \sum_n \lambda_n^{(c)} - \sum_n q_n^{(c)*}$, where $q_n^{(c)*}$ is the optimal queue overflow rate.

B. Features of Network Control

Our control policy that solves (12)-(15) has two main features. First, we have a packet dropping mechanism discarding data from the network when queues build up. An observation here 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 (12) into these two control features, we define, for each class $c \in \mathcal{C}$, a utility function $h_c(\cdot)$ related to $g_c(\cdot)$ as

$$h_c(r_c) \triangleq g_c(r_c) - \theta_c\, r_c, \tag{16}$$

where $\theta_c \ge 0$ are control parameters to be decided later. Using (13), we have $g_c(r_c) = h_c(r_c) + \theta_c\big[\sum_n \lambda_n^{(c)} - \sum_n q_n^{(c)}\big]$. Since $\lambda_n^{(c)}$ are unknown constants, maximizing $\sum_{c\in\mathcal{C}} g_c(r_c)$ is the same as maximizing

$$\sum_{c\in\mathcal{C}} \Big[h_c(r_c) - \theta_c \sum_n q_n^{(c)}\Big]. \tag{17}$$

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

Optimizing the throughput at the receivers amounts to controlling the amount of data *actually delivered*. This is difficult because the data available to the receivers at their upstream nodes is highly correlated with control decisions taken in the rest of the network. Optimizing the packet dropping rates depends on the data available at each network node, which has similar difficulties. To get around these difficulties, we introduce auxiliary control variables $\varphi_n^{(c)} \ge 0$ and $\nu_c \ge 0$ and consider the optimization problem

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

$$\text{subject to} \quad r_c = \nu_c, \quad \forall\, c \in \mathcal{C}, \tag{19}$$

$$q_n^{(c)} \le \varphi_n^{(c)}, \quad \forall\, c \in \mathcal{C},\ n \ne d_c. \tag{20}$$

$$\text{(13)-(15) hold.} \tag{21}$$

This is an equivalent problem to (12)-(15). The constraints (19) and (20) can be enforced by stabilizing virtual queues that will appear in our control policy. The new control variables ν_c and $\varphi_n^{(c)}$ to be optimized can now be chosen freely unconstrained by past control actions in the network. Introducing auxiliary variables and setting up virtual queues are at the heart of using Lyapunov drift theory to solve network optimization problems.

IV. Maximizing the Weighted Sum Throughput

For ease of exposition, we first consider the special case of maximizing the weighted sum throughput in the network. For each class $c \in \mathcal{C}$, we let $g_c(r_c) = a_c\, r_c$ for some $a_c > 0$. We present a threshold-based packet dropping policy that, together with back-pressure routing, solves this problem. Surprisingly, flow control is not needed here. This is because maximizing the weighted sum throughput is equivalent to minimizing the weighted packet dropping rate $\sum_{n,c} a_c\, q_n^{(c)}$. Indeed, choosing $\theta_c = a_c$ in (16), we have $h_c = 0$ for all classes c, under which maximizing the equivalent objective (17) is the same as minimizing $\sum_{n,c} a_c\, q_n^{(c)}$. In the next section, we will combine the threshold-based packet dropping policy with receiver-based flow control to solve the general utility maximization problem.

A. Control Policy

To optimize packet dropping rates, we set up a *drop queue* $D_n^{(c)}(t)$ associated with each queue $Q_n^{(c)}(t)$. The packets that are dropped from $Q_n^{(c)}(t)$ in a slot, denoted by $\widetilde{d}_n^{(c)}(t)$, are first stored in $D_n^{(c)}(t)$ for eventual deletion. From (1), we have

$$\widetilde{d}_n^{(c)}(t) = \min\left[\Big(Q_n^{(c)}(t) - \sum_b \mu_{nb}^{(c)}(t)\Big)^+,\ d_n^{(c)}(t)\right]. \tag{22}$$

Note that the quantity $\widetilde{d}_n^{(c)}(t)$ is the *actual* packets dropped from $Q_n^{(c)}(t)$, which is strictly less than the allocated drop rate $d_n^{(c)}(t)$ if queue $Q_n^{(c)}(t)$ does not have sufficient data. Packets are permanently deleted from $D_n^{(c)}(t)$ at the rate of $\varphi_n^{(c)}(t) \in [0, d_{\max}]$ in slot t. The queue $D_n^{(c)}(t)$ evolves according to

$$D_n^{(c)}(t+1) = \big[D_n^{(c)}(t) - \varphi_n^{(c)}(t)\big]^+ + \widetilde{d}_n^{(c)}(t). \tag{23}$$

Assume initially $D_n^{(c)}(0) = V\theta_c = V a_c$ for all n and c, where $V > 0$ is a control parameter.[3] If queue $D_n^{(c)}(t)$

[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.

is stabilized, then minimizing the service rate of $D_n^{(c)}(t)$ effectively minimizes the time average of dropped packets at $Q_n^{(c)}(t)$. 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 $V > 0$.

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 $Q_{d_c}^{(c)}(t) = 0$ at the receiver d_c. Define

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

We allocate the service rates

$$\mu_{nm}^{(c_l^*(t))}(t) = \begin{cases} \mu_{\max}^{nm} & \text{if } W_l^{(c)*}(t) > 0, \\ 0 & \text{if } W_l^{(c)*}(t) \le 0. \end{cases}$$

Let $\mu_{nm}^{(c)}(t) = 0$ for all classes $c = \mathcal{C}_l \setminus \{c_l^*(t)\}$.

Packet Dropping: At queue $Q_n^{(c)}(t)$, allocate the packet dropping rate $d_n^{(c)}(t)$ (see (1)) according to

$$d_n^{(c)}(t) = \begin{cases} d_{\max} & \text{if } Q_n^{(c)}(t) > D_n^{(c)}(t), \\ 0 & \text{if } Q_n^{(c)}(t) \le D_n^{(c)}(t), \end{cases}$$

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

$$\varphi_n^{(c)}(t) = \begin{cases} d_{\max} & \text{if } D_n^{(c)}(t) > V\theta_c, \\ 0 & \text{if } D_n^{(c)}(t) \le V\theta_c. \end{cases}$$

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

The packet dropping subroutine in this policy is threshold-based. The ORA policy uses local queueing information and does not require the knowledge of exogenous arrival rates. It is notable that network overload is autonomously resolved by each node making local decisions of routing 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

$$D_{\max}^{(c)} \triangleq V\theta_c + d_{\max}, \quad Q_{\max}^{(c)} \triangleq V\theta_c + 2d_{\max}. \tag{24}$$

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

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

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

Proof of Lemma 2: See technical report [23]. ■

In Lemma 2, the value of $Q_{\max}^{(c)}$ is the finite buffer size sufficient at queue $Q_n^{(c)}(t)$. The parameter V controls when queue $Q_n^{(c)}(t)$ starts dropping packets. Indeed, due to $D_n^{(c)}(t) \ge V\theta_c - d_{\max}$, the ORA policy discards packets from $Q_n^{(c)}(t)$ only if $Q_n^{(c)}(t) \ge V\theta_c - d_{\max}$. The quantity $V\theta_c - d_{\max}$ is a controllable threshold beyond which we say queue $Q_n^{(c)}(t)$ is overloaded and should start dropping packets. As we see next, the performance of the ORA policy approaches optimality as the buffer sizes increase.

Theorem 1. Define the limiting throughput of class c as

$$\overline{r_c} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\Big[\sum_{a:(a,d_c)\in\mathcal{L}} \widetilde{\mu}_{ad_c}^{(c)}(\tau) \Big], \tag{25}$$

where $\widetilde{\mu}_{ad_c}^{(c)}(\tau)$ denotes the class c packets received by node d_c over link (a, d_c). The ORA policy yields the limiting weighted sum throughput satisfying

$$\sum_{c\in\mathcal{C}} a_c\, \overline{r_c} \ge \sum_{c\in\mathcal{C}} a_c\, r_c^* - \frac{B}{V}, \tag{26}$$

where (r_c^*) is the optimal throughput vector that solves (12)-(15) under the linear objective function $\sum_{c\in\mathcal{C}} a_c\, r_c$, $V > 0$ is a control parameter, and B is a finite constant defined as

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

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 (26), the ORA policy yields near-optimal performance by choosing the parameter V sufficiently large. Correspondingly, a large V implies a large buffer size of $Q_{\max}^{(c)} = V\theta_c + 2d_{\max}$.

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 (26) reduces to

$$\sum_{c\in\mathcal{C}} a_c\, \overline{r_c} \ge \sum_{c\in\mathcal{C}} a_c \Big(\sum_n \lambda_n^{(c)} \Big) - \frac{B}{V}.$$

That we can choose V arbitrarily large leads to the next corollary.

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

C. 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

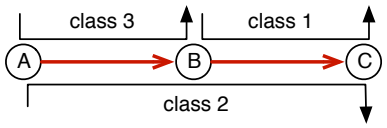


Fig. 3. A 3-node network with three classes of traffic.

the capacity of 1 packet/slot. There are three classes of traffic

TABLE I
THE THROUGHPUT PERFORMANCE OF THE ORA POLICY UNDER FIXED ARRIVAL RATES.

(a) Maximizing $3\,r_1 + 2\,r_2 + r_3$

V	r_1	r_2	r_3
10	.787	.168	.099
20	.867	.133	.410
50	.992	.008	.967
100	.999	0	.999
opt	1	0	1

(b) Maximizing $3\,r_1 + 5\,r_2 + r_3$

V	r_1	r_2	r_3
10	.185	.815	.083
20	.107	.893	.095
50	.031	.969	.031
100	.002	.998	.001
opt	0	1	0

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); classes 2 and 3 compete for service 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.

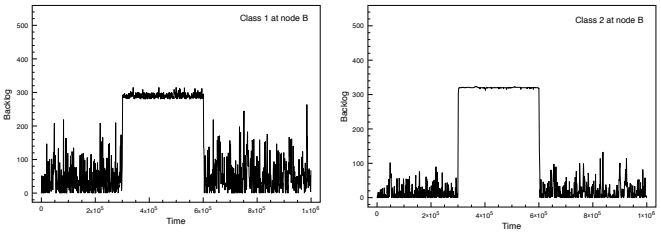
Let r_c be the long-term throughput of class c. Consider the objective of maximizing the weighted sum throughput $3\,r_1 + 2\,r_2 + r_3$; the weights are rewards obtained by serving a packet in a class. The optimal solution is: (i) Always serve class 1 at node B because it yields better rewards than serving class 2. (ii) 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 $3\,r_1 + 5\,r_2 + r_3$. 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 $(0, 1, 0)$. We simulate the ORA policy, and Table I shows its near-optimal performance in both cases as V increases.

2) Time-varying arrival rates: We show that the ORA policy is robust to time-varying arrival rates. Suppose class 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 $\mathcal{T} = [3\times10^5, 6\times10^5)$ and is 0.1 packets/slot elsewhere. We consider the objective of maximizing $3\,r_1 + 5\,r_2 + r_3$. The network is temporarily overloaded in the interval $\mathcal{T}$; the optimal time-average 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 $(0.8, 0.1, 0.8)$.

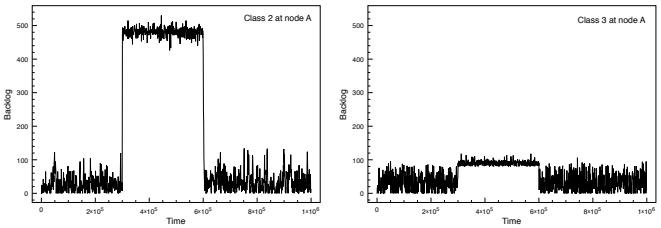
We use the following parameters here: $V = 100$, $A_{\max} = 20$, $d_{\max} = A_{\max} + \mu^{\text{in}}_{\max} = 21$, and $(\theta_1, \theta_2, \theta_3) = (3, 5, 1)$. Table II shows the near-optimal throughput performance of the ORA policy. Figure 4 shows the sample paths of the queue processes $Q_B^{(1)}(t)$, $Q_B^{(2)}(t)$, $Q_A^{(2)}(t)$, and $Q_A^{(3)}(t)$ in the simulation. Clearly the queues suddenly build up when the network enters the overload interval $\mathcal{T}$, but the backlogs are kept close to the upper bound $Q^{(c)}_{\max} = V\theta_c + 2d_{\max}$ without blowing up.

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

time interval	throughput in this interval	optimal value
$[0, 3\cdot10^5)$	$(.797, .097, .771)$	$(.8, .1, .8)$
$[3\cdot10^5, 6\cdot10^5)$	$(.001, .998, 0)$	$(0, 1, 0)$
$[6\cdot10^5, 10^6)$	$(.798, .102, .772)$	$(.8, .1, .8)$



(a) Queue $Q_B^{(1)}(t)$ (b) Queue $Q_B^{(2)}(t)$



(c) Queue $Q_A^{(2)}(t)$ (d) Queue $Q_A^{(3)}(t)$

Fig. 4. The queue processes under the ORA policy with time-varying arrival rates that temporarily overload the network.

V. UTILITY-OPTIMAL CONTROL

We solve the general utility maximization problem (3)-(4) with a network control policy very similar to the ORA policy in the previous section except for an additional flow control mechanism.

A. Virtue Queue

In Section III-B we formulate the equivalent optimization problem (18)-(21) that involves maximizing $\sum_{c\in\mathcal{C}} h_c(\nu_c)$ subject to $r_c = \nu_c$ for all classes c, where ν_c are auxiliary control variables and r_c is the throughput of class c. To enforce the constraint $r_c = \nu_c$, we construct a virtual queue $Z_c(t)$ in which r_c is the virtual arrival rate and ν_c is the time-average service rate. Let $\widetilde{\mu}^{(c)}_{ad_c}(t)$ be the class c packets received by node d_c over link (a, d_c); we have

$$\widetilde{\mu}^{(c)}_{ad_c}(t) = \min\left[Q^{(c)}_a(t), \mu^{(c)}_{ad_c}(t)\right].$$

The arrivals to the virtual queue $Z_c(t)$ in a slot are the total class c packets delivered in that slot, namely, $\sum_a \widetilde{\mu}^{(c)}_{ad_c}(t)$. Let $\nu_c(t)$ be the allocated virtual service rate at $Z_c(t)$ in slot t. The virtual queue $Z_c(t)$ is located at the receiver d_c and evolves according to

$$Z_c(t+1) = \left[Z_c(t) - \nu_c(t)\right]^+ + \sum_a \widetilde{\mu}^{(c)}_{ad_c}(t). \qquad (27)$$

Assume initially $Z_c(0) = 0$ for all classes c. It is well known that if queue $Z_c(t)$ is stable then $r_c \le \nu_c$. But we are interested in the stronger relationship that stabilizing $Z_c(t)$ leads to $r_c = \nu_c$. To make it happen, it suffices to guarantee that queue $Z_c(t)$ wastes as few service opportunities as possible, so that the time-average allocated service rate ν_c is approximately equal to the throughput out of queue $Z_c(t)$. For this, we need two conditions:

1) The queues $Z_c(t)$ usually have more than enough (virtual) data to serve.
2) When $Z_c(t)$ does not have sufficient data, the allocated service rate $\nu_c(t)$ is made arbitrarily small.

To attain the first condition, we use an exponential-type Lyapunov function that bounds the virtual backlog process $Z_c(t)$ away from zero (and centers it around a parameter $Q > 0$). The second condition is attained by a proper choice of the parameters θ_c to be decided later.

B. Control Policy

The following policy, whose construction is sketched in Appendix A, solves the utility maximization problem (3)-(4).

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 shortly. Assume initially $Q_n^{(c)}(0) = Z_c(0) = 0$ and $D_n^{(c)}(0) = V\theta_c$.

Packet Dropping: Same as the ORA policy.

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

$$W_l^{(c)}(t) = Q_a^{(c)}(t) - Q_{d_c}^{(c)}(t), \tag{28}$$

where we abuse the notation by redefining

$$Q_{d_c}^{(c)}(t) = \begin{cases} w\, e^{w(Z_c(t)-Q)} & \text{if } Z_c(t) \ge Q \\ -w\, e^{w(Q-Z_c(t))} & \text{if } Z_c(t) < Q \end{cases} \tag{29}$$

for all classes c. The exponential form of $Q_{d_c}^{(c)}(t)$ is a result of using exponential-type Lyapunov functions. We emphasize that here $Q_{d_c}^{(c)}(t)$ 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 gives us the "desired force" in the form of differential backlog in (28) to pull or push-back data in the network. Thus, unlike standard back-pressure routing that has $Q_{d_c}^{(c)}(t) = 0$, here we use $Q_{d_c}^{(c)}(t)$ as part of the receiver-based flow control mechanism.

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

$$\text{maximize} \quad V h_c\big(\nu_c(t)\big) + \nu_c(t)\, Q_{d_c}^{(c)}(t) \tag{30}$$

$$\text{subject to} \quad 0 \le \nu_c(t) \le \nu_{\max} \tag{31}$$

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), (23), and (27), respectively, in every slot.

C. Choice of Parameters

We briefly discuss how the parameters in the UORA policy are chosen. Let $\epsilon > 0$ be a small constant which affects the performance of the UORA policy (cf. (33)). In (30)-(31), we need the parameter $\nu_{\max}$ to satisfy $\nu_{\max} \ge \max_{c\in\mathcal{C}} r_c^* + \epsilon/2$, where (r_c^*) is solution to the utility maximization (3)-(4) (one feasible choice of $\nu_{\max}$ is the sum of capacities of all links connected to the receivers plus $\epsilon/2$). This choice of $\nu_{\max}$ ensures that queue $Z_c(t)$ can be stabilized when its virtual arrival rate is the optimal throughput r_c^*. Due to technical reasons, we define $\delta_{\max} \triangleq \max[\nu_{\max}, \mu_{\max}^{\text{in}}]$ and choose the parameter

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

in (29). The parameter Q (see (29)) is used to bound the queues $Z_c(t)$ away from zero and center them around Q; for technical reasons, we need $Q \ge \nu_{\max}$. The parameters θ_c are chosen to satisfy $h_c'(x) = g_c'(x) - \theta_c \le 0$ for all $x \ge \epsilon$. This ensures that, when $Z_c(t) < Q$, its virtual service rate $\nu_c(t)$ as the solution to (30)-(31) is less than or equal to ϵ, attaining the second condition mentioned in Section V-A to equalize the arrival rate and the time-average service rate of the virtual queue $Z_c(t)$ (see [23, Lemma 6]). The parameter V captures the tradeoff between utility and buffer sizes to be shown shortly and should be chosen large; for technical reasons, we need V to satisfy $V\theta_c + 2d_{\max} \ge w$.

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

$$Q_n^{(c)}(t) \le Q_{\max}^{(c)}, \quad D_n^{(c)}(t) \le D_{\max}^{(c)}, \quad Z_c(t) \le Z_{\max}^{(c)}$$

for all t, c, and n, where $Q_{\max}^{(c)}$ and $D_{\max}^{(c)}$ are defined in (24) and $Z_{\max}^{(c)}$ is defined as

$$Z_{\max}^{(c)} \triangleq Q + \frac{1}{w} \log\left(\frac{V\theta_c + 2d_{\max}}{w}\right) + \mu_{\max}^{\text{in}}. \tag{32}$$

Proof of Lemma 3: See technical report [23]. ∎

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

$$\sum_c g_c(\overline{r_c}) \ge \sum_c g_c(r_c^*) - \frac{B_1}{V} - \frac{3\epsilon}{2} \sum_c (m_c + \theta_c), \tag{33}$$

where $\overline{r_c}$ is defined in (25), (r_c^*) is the throughput vector that solves the utility maximization problem (3)-(4), and B_1 is a finite constant defined as

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

Proof of Theorem 2: See technical report [23]. ∎

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 $Q_{\max}^{(c)} = V\theta_c + 2d_{\max}$.

TABLE III
THE THROUGHPUT PERFORMANCE OF THE UORA POLICY IN THE 3-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 3-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

E. Simulation of the UORA *Policy*

We conduct two sets of simulations.

1) On the 3-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 network setting, due to symmetry the optimal throughput for class 1 is equal to that of class 3, which is the solution to the simple convex program

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

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 $\epsilon = 0.1$. To satisfy $\theta_c \ge 1/x$ for all $x \ge \epsilon$, we choose $\theta_c = 1/\epsilon = 10$ for all classes c. The value of $\mu_{\max}^{\text{in}}$ in the 3-node network is one. The optimal throughput vector satisfies $\max_c r_c^* = 1$ 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. The simulation time is 10^6 slots. The near-optimal throughput performance is given in Table III. Table IV shows the maximum backlog in each queue $Q_n^{(c)}(t)$ during the simulation. Consistent with Lemma 3, the maximum backlog is bounded by $Q_{\max}^{(c)} = V\theta_c + 2d_{\max} = 10V + 42$.

2) On the tree network in Fig. 2: Consider providing max-min fairness to the three classes of traffic in Fig. 2. Each link has the capacity of one packet per slot. Each one of the four arrival processes has 20 packets arriving in a slot with probability 0.1 and zero packets arrive otherwise. The arrival rates are $(2, 2, 2, 2)$, which overloads 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
THE 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:

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

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. We require $\theta_c \ge x^{-100}$ for all $x \ge \epsilon$. For convenience, let us choose $\theta_c = \epsilon = 1$ for all classes c. The optimal throughput vector satisfies $\max_c r_c^* = 2$, 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. 2 that $\mu_{\max}^{\text{in}} = 2$, and we have $\delta_{\max} = \max[\nu_{\max}, \mu_{\max}^{\text{in}}] = 4$. We have $A_{\max} = 20$ in the arrival processes and by Assumption 1 we choose $d_{\max} = A_{\max} + \mu_{\max}^{\text{in}} = 22$. Let $Q = 100$.

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

In this paper we develop a receiver-based flow control and an in-network packet dropping strategy to cope with network overload. Our scheme is robust to uncooperative users who do not employ source-based flow control and malicious users that intentionally overload the network. A novel feature of our scheme 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 aggregates of data flows. This is in contrast to source-based schemes that can only differentiate between source-destination pairs. We show that when the receiver-based flow control scheme is combined with a threshold-based packet dropping policy at internal network nodes, optimal utility can be achieved.

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," *Journal of the Oper. Res.*, vol. 49, pp. 237–252, 1998.

[3] F. P. Kelly, "Charging and rate control for elastic traffic," *European Trans. Telecommunications*, 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. 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] ——, "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 *IEEE Conf. Decision and Control (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. Springer Berlin Heidelberg, 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," *ACM Computer Communication Review*, vol. 32, pp. 62–73, 2002.

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

[15] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. 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," 2011, working paper.

[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 *Conf. Information Science and Systems (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] C. Li and E. Modiano, "Receiver-based flow control for networks in overload," arXiv report, 2012. [Online]. Available: http://arxiv.org/abs/1207.6354

[24] 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.

[25] 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.

APPENDIX A

We construct a proper Lypuanov drift inequality that leads to the UORA policy. Let $H(t) = \big(Q_n^{(c)}(t); D_n^{(c)}(t); Z_c(t)\big)$ 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

$$L\big(H(t)\big) \triangleq \frac{1}{2}\sum_{c,n\neq d_c}\big[Q_n^{(c)}(t)\big]^2 + \frac{1}{2}\sum_{c,n\neq d_c}\big[D_n^{(c)}(t)\big]^2 + \sum_{c\in\mathcal{C}}\Big(e^{w\left(Z_c(t)-Q\right)} + e^{w\left(Q-Z_c(t)\right)}\Big).$$

The last sum is an exponential-type Lyapunov function whose value grows exponentially when $Z_c(t)$ deviates in both directions from the constant Q. This is useful for both stabilizing $Z_c(t)$ and ensuring there is sufficient data in $Z_c(t)$. Such Lyapunov functions are previously used in [25] to study the optimal utility-delay tradeoff in wireless networks. We define the Lyapunov drift $\Delta(t) \triangleq \mathbb{E}\left[L\big(H(t+1)\big) - L\big(H(t)\big)|H(t)\right]$, where the expectation is with respect to all randomness in the system in slot t.

Define the indicator function $1_c^{\mathrm{R}}(t) = 1$ if $Z_c(t) \geq Q$ and 0 otherwise; define $1_c^{\mathrm{L}}(t) = 1 - 1_c^{\mathrm{R}}(t)$. Define $\delta_c(t) = \nu_c(t) - \sum_{a:(a,d_c)\in\mathcal{L}}\mu_{ad_c}^{(c)}(t)$. The next lemma shows the Lyapunov drift inequality (proof is given in [23]).

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

$$\begin{aligned}
&\Delta(t) - V\sum_c \mathbb{E}\left[h_c\big(\nu_c(t)\big) \mid H(t)\right] \\
&+ V\sum_{nc}\theta_c\,\mathbb{E}\left[\varphi_n^{(c)}(t) \mid H(t)\right] \leq B + \sum_{nc}Q_n^{(c)}(t)\lambda_n^{(c)} \\
&- \sum_{nc}Q_n^{(c)}(t)\mathbb{E}\left[\sum_b \mu_{nb}^{(c)}(t) + d_n^{(c)}(t) - \sum_a \mu_{an}^{(c)}(t) \mid H(t)\right] \\
&- \sum_{nc}D_n^{(c)}(t)\mathbb{E}\left[\varphi_n^{(c)}(t) - d_n^{(c)}(t) \mid H(t)\right] \\
&- V\sum_c \mathbb{E}\left[h_c\big(\nu_c(t)\big) \mid H(t)\right] + V\sum_{nc}\theta_c\,\mathbb{E}\left[\varphi_n^{(c)}(t) \mid H(t)\right] \\
&- w\sum_c 1_c^{\mathrm{R}}(t)e^{w(Z_c(t)-Q)}\Big(\mathbb{E}\left[\delta_c(t) \mid H(t)\right] - \frac{\epsilon}{2}\Big) \\
&+ w\sum_c 1_c^{\mathrm{L}}(t)e^{w(Q-Z_c(t))}\Big(\mathbb{E}\left[\delta_c(t) \mid H(t)\right] + \frac{\epsilon}{2}\Big), \qquad (34)
\end{aligned}$$

where B is a finite constant defined by

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

By isolating decisions variables in (34), it is not difficult to verify that the UORA policy observes the current network state $H(t)$ and minimizes the right-hand side of (34) in every slot.