Dual-resource TCP/AQM for Processing-constrained Networks

Minsu Shin
Department of Electrical Engineering and Computer Science
KAIST, Korea
msshin@netsys.kaist.ac.kr

Song Chong
Department of Electrical Engineering and Computer Science
KAIST, Korea
song@ee.kaist.ac.kr

Injong Rhee
Department of Computer Science
North Carolina State University, USA
rhee@csc.ncsu.edu

***Abstract*— This paper examines congestion control issues for TCP flows that require in-network processing on the fly in network elements such as gateways, proxies, firewalls and even routers. Applications of these flows are increasingly abundant in the future as the Internet evolves. Since these flows require use of CPUs in network elements, both bandwidth and CPU resources can be a bottleneck and thus congestion control must deal with "congestion" on both of these resources. In this paper, we show that conventional TCP/AQM schemes can significantly lose throughput and suffer harmful unfairness in this environment, particularly when CPU cycles become more scarce (which is likely the trend given the recent explosive growth rate of bandwidth). As a solution to this problem, we establish a notion of dual-resource proportional fairness and propose an AQM scheme, called Dual-Resource Queue (DRQ), that can closely approximate proportional fairness for TCP Reno sources with in-network processing requirements. DRQ is scalable because it does not maintain per-flow states while minimizing communication among different resource queues, and is also incrementally deployable because of no required change in TCP stacks. The simulation study shows that DRQ approximates proportional fairness without much implementation cost and even an incremental deployment of DRQ at the edge of the Internet improves the fairness and throughput of these TCP flows. Our work is at its early stage and might lead to an interesting development in congestion control research.**

I. Introduction

Advances in optical network technology enable fast pace increase in physical bandwidth whose growth rate has far surpassed that of other resources such as CPU and memory bus. This phenomenon causes network bottlenecks to shift from bandwidth to other resources. The rise of new applications that require in-network processing hastens this shift, too. For instance, a voice-over-IP call made from a cell phone to a PSTN phone must go through a media gateway that performs audio transcoding "on the fly" as the two end points often use different audio compression standards. Examples of in-network processing services are increasingly abundant from security, performance-enhancing proxies (PEP), to media translation. These services add additional loads to processing capacity in the network components. New router technologies such as extensible routers [1] or programmable routers [2] also need to deal with scheduling of CPU usage per packet as well as bandwidth usage per packet.

In this paper, we examine congestion control issues for an environment where both bandwidth and CPU resources can be a bottleneck. We call this environment *dual-resource* environment. In the dual-resource environment, different flows could have different processing demands per byte.

Traditionally, congestion control research has focused on managing only bandwidth. However, we envision (also it is indeed happening now to some degree) that diverse network services reside somewhere inside the network, most likely at the edge of the Internet, processing, storing or forwarding data packets on the fly. As the in-network processing is likely to be popular in the future, our work that examines whether the current congestion control theory can be applied without modification, or if not, then what scalable solutions can be applied to fix the problem, is highly timely.

In our earlier work [3], we extended proportional fairness to the dual-resource environment and proposed a distributed congestion control protocol for the same environment where end-hosts are cooperative and explicit signaling is available for congestion control. In this paper, we propose a scalable active queue management (AQM) strategy, called *Dual-Resource Queue* (DRQ), that can be used by network routers to approximate proportional fairness, without requiring any change in end-host TCP stacks. Since it does not require any change in TCP stacks, our solution is incrementally deployable in the current Internet. Furthermore, DRQ is highly scalable in the number of flows it can handle because it does not maintain per-flow states or queues. DRQ maintains only one queue per resource and works with classes of application flows whose processing requirements are a priori known or measurable.

Resource scheduling and management of one resource type in network environments where different flows could have different demands are a well-studied area of research. Weighted-fair queuing (WFQ) [4] and its variants such as deficit round robin (DRR) [5] are well known techniques to achieve fair and efficient resource allocation. However, the solutions are not scalable and implementing them in a high-speed router with many flows is difficult since they need to maintain per-flow queues and states. Another extreme is to have routers maintain simpler queue management schemes

This work was supported by the center for Broadband OFDM Mobile Access (BrOMA) at POSTECH through the ITRC program of the Korean MIC, supervised by IITA. (IITA-2005-C1090-0502-0008)

1-4244-0222-0/06/$20.00 (c)2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the Proceedings IEEE Infocom.

such as RED [6], REM [7] or PI [8]. Our study finds that these solutions may yield extremely unfair allocation of CPU and bandwidth and sometimes lead to very inefficient resource usages.

Our study, to the best of our knowledge, is the first in examining the issues of TCP and AQM under the dual-resource environment and we show that by simulation DRQ achieves fair and efficient resource allocation without imposing much implementation cost. The remainder of this paper is organized as follows. In Section II, we define the problem and fairness in the dual-resource environment, in Sections III and IV, we present DRQ and its simulation study, and in Section V, we conclude our paper.

II. Preliminaries: Network Model and Dual-resource Proportional Fairness

A. Network model

We consider a network that consists of a set of unidirectional links, $L = \{1, \cdots, L\}$, and a set of CPUs, $K = \{1, \cdots, K\}$. The transmission capacity (or bandwidth) of link l is B_l (bits/sec) and the processing capacity of CPU k is C_k (cycles/sec). These network resources are shared by a set of flows (or data sources), $S = \{1, \cdots, S\}$. Each flow s is associated with its data rate r_s (bits/sec) and its end-to-end route (or path) which is defined by a set of links, $L(s) \subset L$, and a set of CPUs, $K(s) \subset K$, that flow s travels through. Let $S(l) = \{s \in S | l \in L(s)\}$ be the set of flows that travel through link l and let $S(k) = \{s \in S | k \in K(s)\}$ be the set of flows that travel through CPU k. Note that this model is general enough to include various types of router architecture and network element with multiple CPUs and transmission links.

Flows can have different CPU demands. We represent this notion by *processing density* w_s^k, $k \in K$, of each flow s, which is defined to be the average number of CPU cycles required per bit when flow s is processed by CPU k. w_s^k depends on k since different processing platforms (CPU, OS, and software) would require a different number of CPU cycles to process the same flow s. The processing demand of flow s at CPU k is then $w_s^k r_s$ (cycles/sec).

Since there are limits on CPU and bandwidth capacities, the amount of processing and bandwidth usage by all flows sharing these resources must be less than or equal to the capacities at anytime. We represent this notion by the following two constraints: for each CPU $k \in K$, $\sum_{s \in S(k)} w_s^k r_s \leq C_k$ (*processing constraint*) and for each link $l \in L$, $\sum_{s \in S(l)} r_s \leq B_l$ (*bandwidth constraint*). These constraints are called *dual-resource constraints* and a nonnegative rate vector $r = [r_1, \cdots, r_S]^T$ satisfying these dual constraints for all CPUs $k \in K$ and all links $l \in L$ is said to be *feasible*.

A1: We assume that each CPU $k \in K$ knows the processing densities w_s^k's for all the flows $s \in S(k)$.

This assumption is reasonable because a majority of Internet applications are known and their processing requirements can be measured either off-line or on-line as discussed below. In practice, network flows could be readily classified into a small number of application types [9]–[12]. That is, there is a finite set of application types, a flow is an instance of an application type, and flows will have different processing densities only if they belong to different application types. In [9], applications have been divided into two categories: header-processing applications and payload-processing applications, and each category has been further divided into a set of benchmark applications. In particular, [12] proposes an empirical model for the per-packet processing time of these benchmark applications for a given processing platform, which is, interesting enough, a simple affine function of packet size M, i.e., $\mu_a^k + \nu_a^k M$ where μ_a^k and ν_a^k are the parameters specific to each benchmark application a for a given processing platform k. Thus, the processing density (in cycles/bit) of a packet of size M from application a at platform k can be modelled as $\frac{\mu_a^k}{M} + \nu_a^k$. Therefore, the average processing density w_a^k of application a at platform k can be computed upon arrival of a packet using an exponentially weighted moving average (EWMA) filter:

$$w_a^k \leftarrow (1-\lambda) w_a^k + \lambda (\frac{\mu_a^k}{M} + \nu_a^k), \quad 0 < \lambda < 1. \tag{1}$$

One could also directly measure the quantity $\frac{\mu_a^k}{M} + \nu_a^k$ in Eq. (1) as a whole instead of relying on the empirical model by counting the number of CPU cycles actually consumed by a packet while the packet is being processed. Lastly, determining the application type an arriving packet belongs to is an easy task in many commercial routers today since L3/L4 packet classification is a default functionality.

B. Proportional fairness in the dual-resource environment

Fairness and efficiency are two main objectives in resource allocation. The notion of fairness and efficiency has been extensively studied and well understood with respect to bandwidth sharing. In particular, proportionally fair (PF) rate allocation has been considered as the bandwidth sharing strategy that can provide a good balance between fairness and efficiency [13], [14].

In our recent work [3], we extended the notion of proportional fairness to the dual-resource environment where processing and bandwidth resources are jointly constrained. In the following, we present this notion and its potential advantages for the dual-resource environment to define our goal for our main study of this paper on TCP/AQM.

Consider an aggregate log utility maximization problem ($\mathbf{P}$) with dual constraints:

$$\mathbf{P}: \quad \max_r \sum_{s \in S} \alpha_s \log r_s \tag{2}$$

$$\text{subject to} \quad \sum_{s \in S(k)} w_s^k r_s \leq C_k, \quad \forall\, k \in K \tag{3}$$

$$\sum_{s \in S(l)} r_s \leq B_l, \quad \forall\, l \in L \tag{4}$$

$$r_s \geq 0, \quad \forall\, s \in S \tag{5}$$

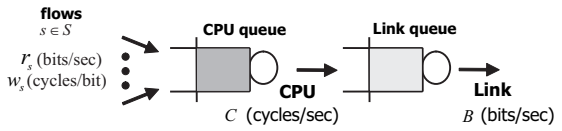


Fig. 1. Single-CPU and single-link network

where α_s is the weight (or willingness to pay) of flow s. The solution r^* of this problem is unique since it is a strictly concave maximization problem over a convex set [15]. Furthermore, r^* is weighted proportionally fair since $\sum_{s\in S} \alpha_s \frac{r_s - r_s^*}{r_s^*} \leq 0$ holds for all feasible rate vectors r by the optimality condition of the problem. We define this allocation to be *(dual-resource) PF rate allocation*. Note that this allocation can be different from Kelly's PF allocation [13] since the set of feasible rate vectors can be different from that of Kelly's formulation due to the extra processing constraint (3).

From the duality theory [15], r^* satisfies that

$$r_s^* = \frac{\alpha_s}{\sum_{k\in K(s)} w_s^k \theta_k^* + \sum_{l\in L(s)} \pi_l^*}, \quad \forall\ s \in S, \qquad (6)$$

where θ^*=$[\theta_1^*, \cdots, \theta_K^*]^T$ and π^*=$[\pi_1^*, \cdots, \pi_L^*]^T$ are Lagrange multiplier vectors for Eqs. (3) and (4), respectively, and θ_k^* and π_l^* can be interpreted as congestion prices of CPU k and link l, respectively. Eq. (6) reveals an interesting property that the PF rate of each flow is inversely proportional to the aggregate congestion price of its route with the contribution of each θ_k^* being weighted by w_s^k. The congestion price θ_k^* or π_l^* is positive only when the corresponding resource becomes a bottleneck, and is zero, otherwise.

To illustrate the characteristics of PF rate allocation in the dual-resource environment, let us consider a limited case where there are only one CPU and one link in the network, as shown in Figure 1. For now, we drop k and l in the notation for simplicity. Let $\bar{w}_a$ and $\bar{w}_h$ be the weighted arithmetic and harmonic means of the processing densities of flows sharing the CPU and link, respectively. So, $\bar{w}_a = \sum_{s\in S} \frac{w_s \alpha_s}{\sum_{s\in S} \alpha_s}$ and $\bar{w}_h = \left(\sum_{s\in S} \frac{\alpha_s}{w_s \sum_{s\in S} \alpha_s}\right)^{-1}$. There exist three cases as below.

- *CPU-limited case* ($\theta^* > 0$ and $\pi^* = 0$): $r_s^* = \frac{\alpha_s}{w_s \theta^*}$, $\forall s \in S$, $\sum_{s\in S} w_s r_s^* = C$ and $\sum_{s\in S} r_s^* \leq B$. From these, we know that this case occurs when $\frac{C}{B} \leq \bar{w}_h$ and PF rate allocation becomes $r_s^* = \frac{\alpha_s C}{w_s \sum_{s\in S} \alpha_s}$, $\forall s \in S$.
- *Bandwidth(BW)-limited case* ($\theta^* = 0$ and $\pi^* > 0$): $r_s^* = \frac{\alpha_s}{\pi^*}$, $\forall s \in S$, $\sum_{s\in S} w_s r_s^* \leq C$ and $\sum_{s\in S} r_s^* = B$. From these, we know that this case occurs when $\frac{C}{B} \geq \bar{w}_a$ and PF rate allocation becomes $r_s^* = \frac{\alpha_s B}{\sum_{s\in S} \alpha_s}$, $\forall s \in S$.
- *Jointly-limited case* ($\theta^* > 0$ and $\pi^* > 0$): This case occurs when $\bar{w}_h < \frac{C}{B} < \bar{w}_a$. By plugging $r_s^* = \frac{\alpha_s}{w_s \theta^* + \pi^*}$, $\forall s \in S$, into $\sum_{s\in S} w_s r_s^* = C$ and $\sum_{s\in S} r_s^* = B$, we can obtain θ^*, π^* and consequently r_s^*, $\forall s \in S$.

Figures 2 (a) and (b) illustrate the bandwidth and CPU allocations enforced by PF rate allocation in the single-CPU and single-link case using an example of four flows with identical weights (α_s=1, $\forall s$) and different processing densities (w_1, w_2, w_3, w_4) = (1, 2, 4, 8) where $\bar{w}_h$=2.13 and $\bar{w}_a$=3.75. For comparison, we also consider a rate allocation in which flows with an identical end-to-end path get an equal share of the maximally achievable throughput of the path and call it *TCP-like rate allocation*. That is, if TCP flows run on the example network in Figure 1 with ordinary AQM schemes such as RED on both CPU and link queues, they would have the same long-term throughput. Thus, in our example, TCP-like rate allocation is defined to be the maximum equal rate vector satisfying the dual constraints, which is $r_s = \frac{B}{S}$, $\forall s$, if $\frac{C}{B} \geq \bar{w}_a$, and $r_s = \frac{C}{\bar{w}_a S}$, $\forall s$, otherwise. The bandwidth and CPU allocations enforced by TCP-like rate allocation are shown in Figures 2 (c) and (d).

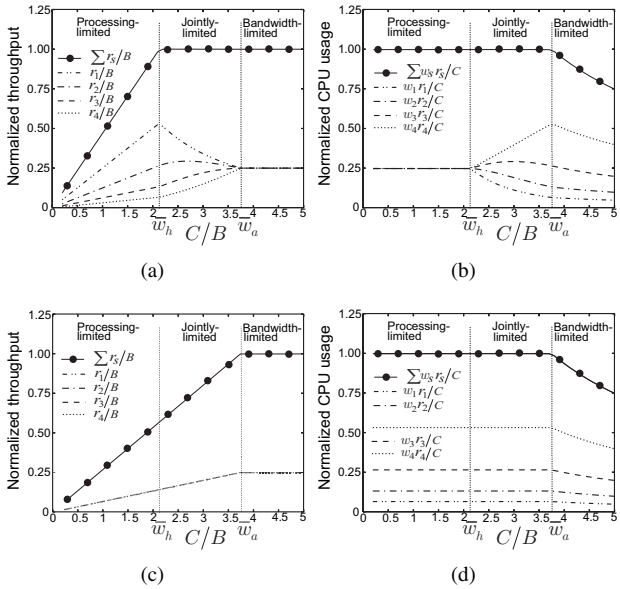


Fig. 2. Fairness and efficiency in the dual-resource environment (single-CPU and single-link network): (a) and (b) respectively show the normalized bandwidth and CPU allocations enforced by PF rate allocation, and (c) and (d) respectively show the normalized bandwidth and CPU allocations enforced by TCP-like rate allocation. When $C/B < \bar{w}_a$, TCP-like rate allocation gives lower bandwidth utilization than PF rate allocation (shown in (a) and (c)) and has an unfair allocation of CPU cycles (shown in (d)).

From Figure 2, we observe that TCP-like rate allocation yields far less aggregate throughput than PF rate allocation when $C/B < \bar{w}_a$, i.e., in both CPU-limited and jointly-limited cases. Intuitively, this is because TCP-like allocation which finds an equal rate allocation yields unfair sharing of CPU cycles as CPU becomes a bottleneck (see Figure 2 (d)), which causes the severe aggregate throughput drop. In contrast, PF allocation yields equal sharing of CPU cycles, i.e., $w_s r_s$ become equal for all $s \in S$, as CPU becomes a bottleneck (see Figure 2 (b)), which mitigates the aggregate throughput drop. This problem in TCP-like allocation would get more severe when the processing densities of flows have a more skewed distribution.

In summary, in a single-CPU and single-link network,

PF rate allocation achieves equal bandwidth sharing when bandwidth is a bottleneck, equal CPU sharing when CPU is a bottleneck, and a good balance between equal bandwidth sharing and equal CPU sharing when bandwidth and CPU form a joint bottleneck. Moreover, in comparison to TCP-like rate allocation, such consideration of CPU fairness in PF rate allocation can increase aggregate throughput significantly when CPU forms a bottleneck either alone or jointly with bandwidth.

III. Main Result: Scalable TCP/AQM Algorithm

In this section, we present a scalable AQM scheme, called Dual-Resource Queue (DRQ), that can approximately implement dual-resource PF rate allocation described in Section II for TCP-Reno flows. DRQ modifies RED [6] to achieve PF allocation without incurring per-flow operations (queueing or state management). DRQ does not require any change in TCP stacks.

A. DRQ objective and optimality

We describe a TCP/AQM network using a fluid model, as customary in the literature [16]–[21]. Let $x_s(t)$ (bits/sec) be the average data rate of TCP source s at time t where the average is taken over the time interval Δ (seconds) and Δ is assumed to be on the order of several round-trip times (RTTs), i.e., large enough to average out the additive-increase and multiplicative decrease (AIMD) oscillation of TCP. Define the RTT τ_s of source s by $\tau_s = \tau_{si} + \tau_{is}$ where τ_{si} denotes forward-path delay from source s to resource i and τ_{is} denotes backward-path delay from resource i to source s.

A2: We assume that each TCP flow s has a constant RTT τ_s, as customary in the fluid modelling of TCP dynamics [16]–[21].

Let $y_l(t)$ be the average queue length at link l at time t, measured in bits. Then,

$$\dot{y}_l(t) = \begin{cases} \sum_{s\in S(l)} x_s(t-\tau_{sl}) - B_l & y_l(t) > 0 \\ \left[\sum_{s\in S(l)} x_s(t-\tau_{sl}) - B_l\right]^+ & y_l(t) = 0. \end{cases} \quad (7)$$

Similarly, let $z_k(t)$ be the average queue length at CPU k at time t, measured in CPU cycles. Then,

$$\dot{z}_k(t) = \begin{cases} \sum_{s\in S(k)} w_s^k x_s(t-\tau_{sk}) - C_k & z_k(t) > 0 \\ \left[\sum_{s\in S(k)} w_s^k x_s(t-\tau_{sk}) - C_k\right]^+ & z_k(t) = 0. \end{cases} \quad (8)$$

Let $p_s(t)$ be the end-to-end marking (or loss) probability at time t to which TCP source s reacts. Then, the rate-adaptation dynamics of TCP Reno, particularly in the timescale of several RTTs, can be readily described by [19]

$$\dot{x}_s(t) = \begin{cases} \frac{M_s(1-p_s(t))}{N_s\tau_s^2} - \frac{2}{3}\frac{x_s^2(t)p_s(t)}{N_sM_s} & x_s(t) > 0 \\ \left[\frac{M_s(1-p_s(t))}{N_s\tau_s^2} - \frac{2}{3}\frac{x_s^2(t)p_s(t)}{N_sM_s}\right]^+ & x_s(t) = 0 \end{cases} \quad (9)$$

where M_s is the average packet size in bits of TCP flow s and N_s is the number of consecutive data packets that are acknowledged by an ACK packet in TCP flow s (N_s is typically 2).

In DRQ, we employ one RED queue per one resource. Each RED queue computes a probability (we refer to it as *pre-marking probability*) in the same way as an ordinary RED queue computes its marking probability.

That is, the RED queue at link l computes a pre-marking probability $\rho_l(t)$ at time t by

$$\rho_l(t) = \begin{cases} 0 & \hat{y}_l(t) \le \underline{b}_l \\ \frac{m_l}{\overline{b}_l - \underline{b}_l}(\hat{y}_l(t) - \underline{b}_l) & \underline{b}_l \le \hat{y}_l(t) \le \overline{b}_l \\ \frac{1-m_l}{\overline{b}_l}(\hat{y}_l(t) - \overline{b}_l) + m_l & \overline{b}_l \le \hat{y}_l(t) \le 2\overline{b}_l \\ 1 & \hat{y}_l(t) \ge 2\overline{b}_l \end{cases} \quad (10)$$

$$\dot{\hat{y}}_l(t) = \frac{\log_e(1-\lambda_l)}{\eta_l}\hat{y}_l(t) - \frac{\log_e(1-\lambda_l)}{\eta_l}y_l(t) \quad (11)$$

where $m_l \in (0,1]$, $0 \le \underline{b}_l < \overline{b}_l$ and Eq. (11) is the continuous-time representation of the EWMA filter [17] used by the RED, i.e.,

$$\hat{y}_l((k+1)\eta_l) = (1-\lambda_l)\hat{y}_l(k\eta_l) + \lambda_l y_l(k\eta_l),\ \lambda_l \in (0,1). \quad (12)$$

Eq. (11) does not model the case where the averaging timescale of the EWMA filter is smaller than the averaging timescale Δ on which $y_l(t)$ is defined. In this case, Eq. (11) must be replaced by $\hat{y}_l(t) = y_l(t)$.

Similarly, the RED queue at CPU k computes a pre-marking probability $\sigma_k(t)$ at time t by

$$\sigma_k(t) = \begin{cases} 0 & \hat{v}_k(t) \le \underline{b}_k \\ \frac{m_k}{\overline{b}_k - \underline{b}_k}(\hat{v}_k(t) - \underline{b}_k) & \underline{b}_k \le \hat{v}_k(t) \le \overline{b}_k \\ \frac{1-m_k}{\overline{b}_k}(\hat{v}_k(t) - \overline{b}_k) + m_k & \overline{b}_k \le \hat{v}_k(t) \le 2\overline{b}_k \\ 1 & \hat{v}_k(t) \ge 2\overline{b}_k \end{cases} \quad (13)$$

$$\dot{\hat{v}}_k(t) = \frac{\log_e(1-\lambda_k)}{\eta_k}\hat{v}_k(t) - \frac{\log_e(1-\lambda_k)}{\eta_k}v_k(t) \quad (14)$$

where $v_k(t)$ is the translation of $z_k(t)$ in bits.

Given these pre-marking probabilities, the objective of DRQ is to mark (or discard) packets in such a way that the end-to-end marking (or loss) probability $p_s(t)$ seen by each TCP flow s at time t becomes

$$p_s(t) = \frac{\left(\sum_{k\in K(s)} w_s^k \sigma_k(t-\tau_{ks}) + \sum_{l\in L(s)} \rho_l(t-\tau_{ls})\right)^2}{1 + \left(\sum_{k\in K(s)} w_s^k \sigma_k(t-\tau_{ks}) + \sum_{l\in L(s)} \rho_l(t-\tau_{ls})\right)^2}. \quad (15)$$

The actual marking scheme that can closely approximate this objective function will be given in Section III-B.

The Reno/DRQ network model given by Eqs. (7)-(15) is called *average Reno/DRQ network* as the model describes the interaction between DRQ and Reno dynamics in long-term average rates rather than explicitly capturing instantaneous TCP rates in the AIMD form. This average network model enables us to study fixed-valued equilibrium and consequently establish *in an average sense* the *equilibrium equivalence* of a Reno/DRQ network and a network with the same configuration but under dual-resource PF congestion control.

Let $x = [x_1, \cdots, x_S]^T$, $\sigma = [\sigma_1, \cdots, \sigma_K]^T$, $\rho = [\rho_1, \cdots, \rho_L]^T$, $p = [p_1, \cdots, p_S]^T$, $y = [y_1, \cdots, y_L]^T$, $z = [z_1, \cdots, z_K]^T$, $v = [v_1, \cdots, v_K]^T$, $\hat{y} = [\hat{y}_1, \cdots, \hat{y}_L]^T$ and $\hat{v} = [\hat{v}_1, \cdots, \hat{v}_K]^T$.

Proposition 1: Consider an average Reno/DRQ network given by Eqs. (7)-(15) and formulate the corresponding aggregate log utility maximization problem (Problem **P**) as in Eqs. (2)-(5) with $\alpha_s = \frac{\sqrt{3/2}M_s}{\tau_s}$. If the Lagrange multiplier vectors, θ^* and π^*, of this corresponding Problem **P** satisfy the following conditions:

$$\mathbf{C1}: \quad \theta_k^* < 1, \ \forall k \in K(s), \ \forall s \in S, \tag{16}$$

$$\mathbf{C2}: \quad \pi_l^* < 1, \ \forall l \in L(s), \ \forall s \in S, \tag{17}$$

then, the average Reno/DRQ network has a unique equilibrium point (x^*, σ^*, ρ^*, p^*, y^*, z^*, v^*, $\hat{y}^*$, $\hat{v}^*$) and (x^*, σ^*, ρ^*) is the primal-dual optimal solution of the corresponding Problem **P**. In addition, $v_k^* > \underline{b}_k$ if $\sigma_k^* > 0$ and $0 \le v_k^* \le \underline{b}_k$ otherwise, and $y_l^* > \underline{b}_l$ if $\rho_l^* > 0$ and $0 \le y_l^* \le \underline{b}_l$ otherwise, for all $k \in K$ and $l \in L$.

Proof: The proof is given in Appendix. ■

Proposition 1 implies that once the Reno/DRQ network reaches its steady state (i.e., equilibrium), the average data rates of Reno sources satisfy weighted proportional fairness with weights $\alpha_s = \frac{\sqrt{3/2}M_s}{\tau_s}$. In addition, if a CPU k is a bottleneck (i.e., $\sigma_k^* > 0$), its average equilibrium queue length v_k^* stays at a constant value greater than $\underline{b}_k$, and if not, it stays at a constant value between 0 and $\underline{b}_k$. The same is true for link congestion.

The existence and uniqueness of such an equilibrium point in the Reno/DRQ network is guaranteed if conditions **C1** and **C2** hold in the corresponding Problem **P**. Otherwise, the Reno/DRQ networks do not have an equilibrium point.

In the current Internet environment, however, these conditions will hardly be violated particularly as the bandwidth-delay products of flows increase. By applying **C1** and **C2** to the Lagrangian optimality condition of Problem **P** in Eq. (6) with $\alpha_s = \frac{\sqrt{3/2}M_s}{\tau_s}$, we have

$$\frac{r_s^*\tau_s}{M_s} = \frac{\sqrt{3/2}}{\sum_{k\in K(s)} w_s^k\theta_k^* + \sum_{l\in L(s)}\pi_l^*} \tag{18}$$

$$> \frac{\sqrt{3/2}}{\sum_{k\in K(s)} w_s^k + |L(s)|} \tag{19}$$

where $\frac{r_s^*\tau_s}{M_s}$ is the bandwidth-delay product (or window size) of flow s, measured in packets. The maximum packet size in the Internet is $M_s = 1,536$ bytes (i.e., maximum Ethernet packet size). Flows that have the minimum processing density are IP forwarding applications with maximum packet size [9]. For instance, a measurement study in [12] showed that per-packet processing time required for NetBSD radix-tree routing table lookup on a Pentium 167 MHz processor is 51 μs (for a faster CPU, the processing time reduces; so as what matters is the number of cycles per bit, this estimate applies to the other CPUs). Thus, the processing density for this application flow is about w_s^k=51(μsec)x167(MHz)/1,536(bytes)=0.69 (cycles/bit). Therefore, from Eq. (19), the worst-case lower bound on the window size becomes $\frac{r_s^*\tau_s}{M_s} > 1.77$ (packets), which occurs when the flow traverses a CPU only in the path (i.e., $|K(s)| = 1$ and $|L(s)| = 0$) . This concludes that the conditions **C1** and **C2** will never be violated as long as the steady-state average TCP window size is sustainable at a value greater than or equal to 2 packets, even in the worst case.

B. DRQ implementation

In this section, we present a simple scalable packet marking (or discarding) scheme that closely approximates the DRQ objective function we laid out in Eq. (15).

A3: We assume that for all times

$$\left(\sum_{k\in K(s)} w_s^k\sigma_k(t-\tau_{ks}) + \sum_{l\in L(s)}\rho_l(t-\tau_{ls})\right)^2 \ll 1, \ \forall s \in S. \tag{20}$$

This assumption implies that $(w_s^k\sigma_k(t))^2 \ll 1$, $\forall k \in K$, $\rho_l(t)^2 \ll 1$, $\forall l \in L$, and any product of $w_s^k\sigma_k(t)$ and $\rho_l(t)$ is also much smaller than 1. Note that our analysis is based on long-term average values of $\sigma_k(t)$ and $\rho_l(t)$. The typical operating points of TCP in the Internet during steady state where TCP shows a reasonable performance are under low end-to-end loss probabilities (less than 1%). Since the end-to-end average probabilities are low, the marking probabilities at individual links and CPUs can be much lower.

Let R be the set of all the resources (including CPUs and links) in the network. Also, for each flow s, let $R(s) = \{1, \cdots, |R(s)|\} \subset R$ be the set of all the resources that it traverses along its path and let $i \in R(s)$ denote the i-th resource along its path and indicate whether it is a CPU or a link. Then, some manipulation after applying Assumption **A3** to Eq. (15) gives

$$p_s(t) \approx \left(\sum_{k\in K(s)} w_s^k\sigma_k(t-\tau_{ks}) + \sum_{l\in L(s)}\rho_l(t-\tau_{ls})\right)^2 \tag{21}$$
$$= \sum_{i=1}^{|R(s)|}\delta_i(t-\tau_{is}) + \sum_{i=2}^{|R(s)|}\sum_{i'=1}^{i-1}\epsilon_i(t-\tau_{is})\epsilon_{i'}(t-\tau_{i's})$$

where

$$\delta_i(t) = \begin{cases} (w_s^i\sigma_i(t))^2 & \text{if } i \text{ indicates CPU} \\ \rho_i(t)^2 & \text{if } i \text{ indicates link} \end{cases} \tag{22}$$

and

$$\epsilon_i(t) = \begin{cases} \sqrt{2}w_s^i\sigma_i(t) & \text{if } i \text{ indicates CPU} \\ \sqrt{2}\rho_i(t) & \text{if } i \text{ indicates link.} \end{cases} \tag{23}$$

Eq. (21) tells that each resource $i \in R(s)$ (except the first resource in $R(s)$, i.e., i=1) contributes to $p_s(t)$ with

When a packet arrives at resource i at time t:
if (ECN $\neq$ 11)
set ECN to 11 with probability $\delta_i(t)$;
if (ECN == 00)
set ECN to 10 with probability $\epsilon_i(t)$;
else if (ECN == 10)
set ECN to 11 with probability $\epsilon_i(t)$;

Fig. 3. DRQ's ECN marking algorithm

two quantities, $\delta_i(t-\tau_{is})$ and $\sum_{i'=1}^{i-1}\epsilon_i(t-\tau_{is})\epsilon_{i'}(t-\tau_{i's})$. Moreover, resource i can compute the former using its own congestion information, i.e., $\sigma_i(t)$ if it is a CPU or $\rho_i(t)$ if it is a link, whereas it cannot compute the latter without knowing the congestion information of its upstream resources on its path ($\forall\ l' < l$). That is, the latter requires an inter-resource signaling to exchange the congestion information. For this reason, we refer to $\delta_i(t)$ as *intra-resource marking probability* of resource i at time t and $\epsilon_i(t)$ as *inter-resource marking probability* of resource i at time t. We solve this intra- and inter-resource marking problem using two-bit ECN flags without explicit communication between resources.

Consider the two-bit ECN field in the IP header [22]. Among the four possible values of ECN bits, we use three values to indicate three cases: *initial state* (ECN=00), *signaling-marked* (ECN=10) and *congestion-marked* (ECN=11). When a packet is congestion-marked (ECN=11), the packet is either marked (if TCP supports ECN) or discarded (if not). DRQ sets the ECN bits as shown in Figure 3.

Below, we verify that the ECN marking scheme in Figure 3 approximately implements the objective function in Eq. (21). Consider a flow s with path $R(s)$. For now, we drop the time index t to simplify the notation. Let P_{00}^i, P_{10}^i, P_{11}^i respectively denote the probabilities that packets of flow s will have ECN=00, ECN=10, ECN=11, upon departure from resource i in $R(s)$. Then, the proposed ECN marking scheme can be expressed by the following recursion. For $i = 1, 2, \cdots, |R(s)|$,

$$\begin{aligned} P_{11}^i &= P_{11}^{i-1} + (1-P_{11}^{i-1})\delta_i + P_{10}^{i-1}(1-\delta_i)\epsilon_i \quad (24)\\ &= 1-(1-\delta_i)(1-P_{11}^{i-1}-P_{10}^{i-1}\epsilon_i), \quad (25)\\ P_{10}^i &= P_{10}^{i-1}(1-\delta_i)(1-\epsilon_i) + P_{00}^{i-1}(1-\delta_i)\epsilon_i, \quad (26)\\ P_{00}^i &= p_{00}^{i-1}(1-\delta_i)(1-\epsilon_i) \quad (27)\end{aligned}$$

with the initial condition that $P_{00}^0 = 1$, $P_{10}^0 = 0$, $P_{11}^0 = 0$.

Evolving i from 0 to $|R(s)|$, we obtain

$$P_{11}^{|R(s)|} = 1 - \prod_{i=1}^{|R(s)|}(1-\delta_i)\left(1 - \sum_{i=2}^{|R(s)|}\sum_{i'=1}^{i-1}\epsilon_i\epsilon_{i'} + \Theta\right) \quad (28)$$

where Θ is the higher-order terms (order ≥ 3) of ϵ_i's. By Assumption **A3**, we have

$$\begin{aligned} P_{11}^{|R(s)|} &\approx 1 - \prod_{i=1}^{|R(s)|}(1-\delta_i)\left(1 - \sum_{i=2}^{|R(s)|}\sum_{i'=1}^{i-1}\epsilon_i\epsilon_{i'}\right) \quad (29)\\ &\approx \sum_{i=1}^{|R(s)|}\delta_l + \sum_{i=2}^{|R(s)|}\sum_{i'=1}^{i-1}\epsilon_i\epsilon_{i'} \quad (30)\end{aligned}$$

which concludes that the proposed ECN marking scheme approximately implements the DRQ objective function in Eq. (21) since $P_{11}^{|R(s)|} = p_s$.

C. DRQ stability

In this section, we explore the stability of Reno/DRQ networks. Unfortunately, analyzing its global stability is an extremely difficult task since the dynamics involved are nonlinear and retarded. Here, we present a partial result concerning local stability, i.e., stability around the equilibrium point.

Define $|R|\text{x}|S|$ matrix $\Gamma(z)$ whose (i,s) element is given by

$$\Gamma_{is}(z) = \begin{cases} w_s^i e^{-z\tau_{is}} & \text{if } s \in S(i) \text{ and } i \text{ indicates CPU} \\ e^{-z\tau_{is}} & \text{if } s \in S(i) \text{ and } i \text{ indicates link} \\ 0 & \text{otherwise.} \end{cases} \quad (31)$$

Proposition 2: An average Reno/DRQ network is locally stable if we choose the RED parameters in DRQ such that $\max\{\frac{m_k}{\bar{b}_k-\underline{b}_k}, \frac{1-m_k}{\bar{b}_k}\}C_k \in (0,\psi)$, $\forall k \in K$, and $\max\{\frac{m_l}{\bar{b}_l-\underline{b}_l}, \frac{1-m_l}{\bar{b}_l}\}B_l \in (0,\psi)$, $\forall l \in L$, and

$$\psi \leq \frac{\sqrt{3/2}\phi_{min}^2[\Gamma(0)]}{|R|\Lambda_{max}^2\tau_{max}^3 w_{max}} \quad (32)$$

where $\Lambda_{max} = \max\{\frac{\max\{C_k\}}{\min\{w_s^k\}\min\{M_s\}}, \frac{\max\{B_l\}}{\min\{M_s\}}\}$, $\tau_{max} = \max\{\tau_s\}$, $w_{max} = \max\{w_s^k, 1\}$ and $\phi_{min}[\Gamma(0)]$ denotes the smallest singular values of the matrix $\Gamma(z)$ evaluated at $z = 0$.

Proof: We omit the proof since it is a straightforward application of the TCP/RED stability result in [23]. ■

IV. Performance

A. Simulation setup

In this section, we use simulation to verify the performance of DRQ in the dual-resource environment with TCP Reno sources. We compare the performance of DRQ with that of the two other AQM schemes that we discussed in the introduction. One scheme is to use the simplest approach where both CPU and link queues use RED and the other is to use DRR (a variant of WFQ) to schedule CPU usage among competing flows according to the processing density of each flow. DRR maintains per flow queues, and equalizes the CPU usage in a round robin fashion when the processing demand is higher than the CPU capacity (i.e., CPU-limited). In some sense, these choices of AQM are two extreme; one is very cheap, but less fair in use of CPU as RED is oblivious to differing CPU demands of flows and the other is very expensive, but fair in use of CPU as DRR installs equal shares of CPU among these flows. Our goal is to demonstrate through simulation that

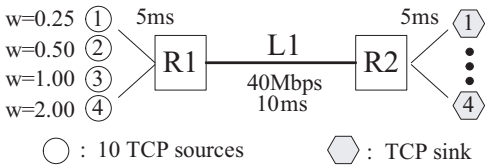


Fig. 4. Single link scenario in dumbell topology

with much less implementation cost, DRQ supports fairness and efficiency as DRR. Note that all three schemes use RED for link queues, but DRQ uses its own marking algorithm for link queues as shown in Figure 3 which uses the marking probability obtained from the underlying RED queue for link queues. We call the scheme with DRR for CPU queues and RED for link queues, *DRR-RED*, the scheme with RED for CPU queues and RED for link queues, *RED-RED*.

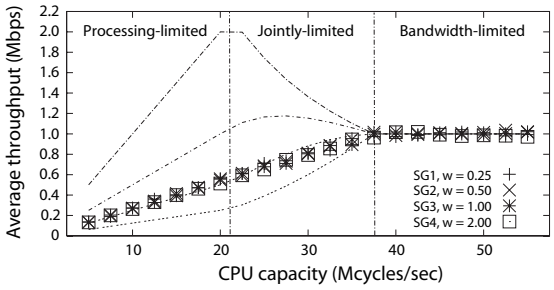
The simulation is performed in the NS-2 [24] environment. We modified NS-2 to emulate the CPU capacity by simply holding a packet for its processing time duration. In the simulation, RED queues are implemented using its default setting for the "gentle" RED mode [25] ($m_i = 0.1$, $\underline{b}_i = 50$ pkts, $\overline{b}_i = 550$ pkts and $\lambda_i = 10^{-4}$. The packet size is fixed at 500 Bytes). The same RED setting is used for the link queues of DRR-RED and RED-RED, and also for both CPU and link queues of DRQ (DRQ uses a function of the marking probabilities to mark or drop packets for both queues). In our analytical model, we separate CPU and link. To simplify the simulation setup and its description, when we refer to a "link" for the simulation setup, we assume that each link l consists of one CPU and one Tx link (i.e., bandwidth).

By adjusting CPU capacity C_l, link bandwidth B_l, and the amount of background traffic, we can control the bottleneck conditions. Our simulation topologies are chosen from a various set of Internet topologies from simple dumbell topologies to more complex WAN topologies. Below we discuss these setups and simulation scenarios in detail and their corresponding results for the three schemes we discussed above.

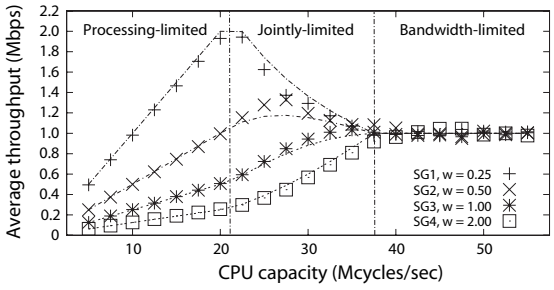
B. Dumbell with long-lived TCP flows

To confirm our analysis in Section II-B, we run a single link bottleneck case. Figure 4 shows an instance of the dumbell topology commonly used in congestion control research. We fix the bandwidth of the bottleneck link to 40 Mbps and vary its CPU capacity from 5 Mcycles/s to 55 Mcycles/s. This variation allows the bottleneck to move from the CPU-limited region to the BW-limited region. Four classes of long-lived TCP flows are added for simulation whose processing densities are 0.25, 0.5, 1.0 and 2.0 respectively. We simulate ten TCP Reno flows for each class. All the flows have the same RTT of 40 ms.

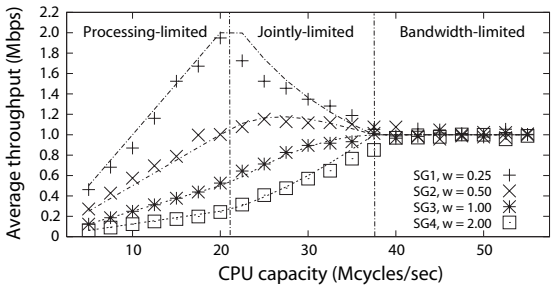
In presenting our results, we take the average throughput of TCP flows that belong to the same class. Figure 5 plots the average throughput of each class. To see whether DRQ achieves PF allocation, we also plot the ideal proportional fair rate for each class (which is shown in a dotted line). As shown in Figure 5(a), where we use typical RED schemes at both



(a) RED-RED



(b) DRR-RED



(c) DRQ

Fig. 5. Average throughput of four different classes of long-lived TCP flows in the dumbell topology. Each class has a different CPU demand per bit (w). No other background traffic is added. The Dotted lines indicate the ideal PF rate allocation for each class. In the figure, we find that DRQ and DRR-RED show good fairness under the CPU-limited region while RED-RED does not.

queues, all TCP flows achieve the same throughput regardless of the CPU capacity of the link and their processing densities. Figures 5(b) and (c) show that the average throughput curves of DRR-RED and DRQ follow the ideal PF rates reasonably well. When CPU is only a bottleneck resource, the PF rate of each flow must be inversely proportional to its processing density w_s, in order to share CPU equally. Under the BW-limited region, the proportionally-fair rate of each flow is identical to the equal share of the bandwidth. Under the jointly-limited region, flows maintain the PF rates while fully utilizing both resources. Although DRQ does not employ the per-flow queue structure as DRR, its performance is comparable to that of DRR-RED.

Figure 6 shows that the aggregate throughput achieved by each scheme. It shows that RED-RED has much lower bandwidth utilization than the two other schemes. This is because, as discussed in Section II-B, when CPU is a bottleneck

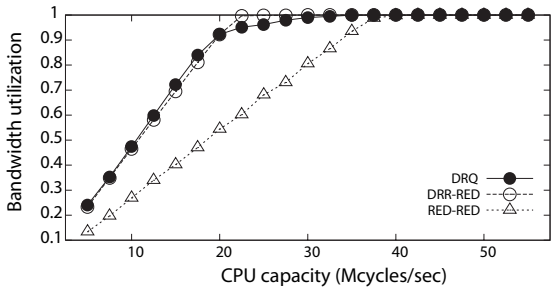


Fig. 6. Comparison of bandwidth utilization in the Dumbbell single bottleneck topology. RED-RED achieves far less bandwidth utilization than DRR-RED and DRQ when CPU becomes a bottleneck.

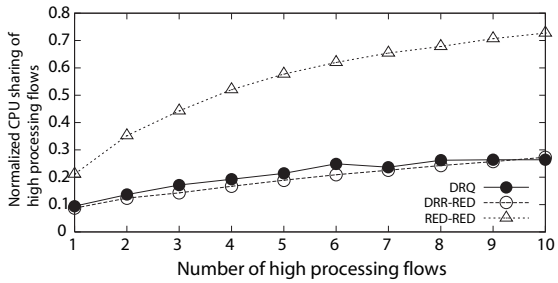
resource, the equilibrium operating points of TCP flows over the RED CPU queue that achieve the equal bandwidth usage while keeping the total CPU usage below the CPU capacity are much lower than those of the other schemes that need to ensure the equal sharing of CPU (not the bandwidth) under the CPU-limited region.

C. Impact of flows with high processing demands

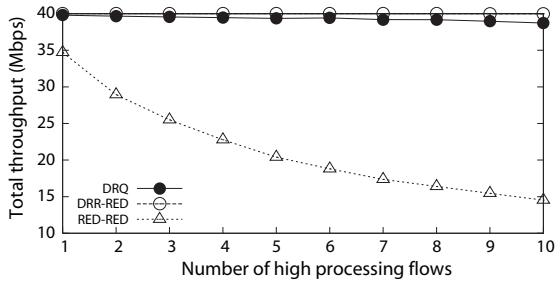
In the introduction, we indicated that RED-RED can cause extreme unfairness in use of resources. To show this by experiment, we construct a simulation run where we fix the CPU capacity to 40 Mcycles/s and add an increasing number of flows with a high CPU demand ($w_s = 10$) in the same setup as the dumbell sink bottleneck environment in Section IV-B. We call these flows *high processing flows*. From Figure 5, at 40 Mcycles/s, when no high processing flows are added, CPU is not a bottleneck. But as the number of high processing flows increases, the network moves into the CPU-limited region. Figure 7 shows the results of this simulation run. In Figure 7 (a), as we increase the number of high processing flows, the aggregate CPU share of high processing flows deviates significantly from the equal CPU share; under a larger number of high processing flows (e.g., 10 flows), these flows dominate the CPU usage over the other lower processing density flows, driving them to starvation. In contrast, DRQ and DRR approximately implement the equal CPU sharing policy. Even though the number of high processing flows increases, the bandwidth remains a bottleneck resource as before, so the link is in the jointly-limited region which is the reason why the CPU share of high-processing flows go beyond 20%.

D. Dumbell with background Internet traffic

No Internet links are without cross traffic. In order to emulate more realistic Internet environments, we add cross traffic modelled from various observations on RTT distribution [26], flow sizes [27] and flow arrival [28]. As modelling the Internet traffic in itself is a topic of research, we do not dwell on which model is more realistic. In this paper, we present one model that contains the statistical characteristics that are commonly assumed or confirmed by researchers. These characteristics include that the distribution of flow sizes has a long-range dependency [29], the RTTs of flows is rather exponentially



(a) CPU sharing

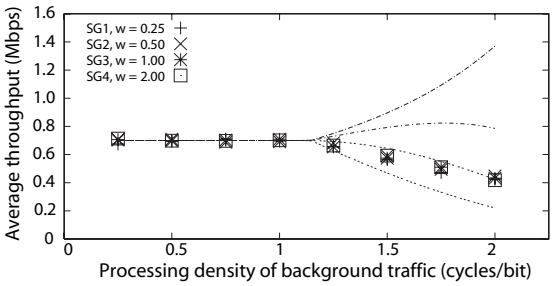


(b) Total throughput

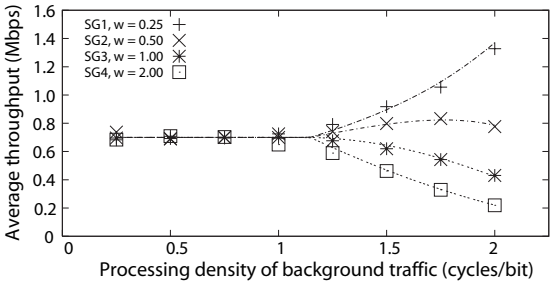
Fig. 7. Impact of high processing flows. As the number of high processing flows increase, the network becomes more CPU-bound. Under RED-RED, these flows can dominate the use of CPU, reaching about 80% CPU usage with only 10 flows, starving 40 competing, but low processing flows.

distributed [30] and the arrivals of flows are exponentially distributed [28], [31]. Following these characteristics, our cross traffic consists of a number of short-lived TCP flows that follow a Poisson arrival process and send a random number of packets derived from the Pareto distribution with an average of 30 packets, and the shape parameter 1.2. The RTT of each short-lived flow is randomly selected from a range of 20 to 60 ms. We fix the bottleneck bandwidth and CPU capacities to 40 Mbps and 40 Mcycles/s, respectively and generate the same number of long-lived TCP flows as in the experiment for Figure 5. With these parameters, the cross traffic consumes about 30% of the link bandwidth capacity. In this experiment, we vary the CPU demand of short-lived flows to create various bottleneck conditions. We make no claims about how realistically our model characterizes the Internet cross traffic, but we believe that the simulation in this model more realistically reflects real network situations than the one with no background traffic.

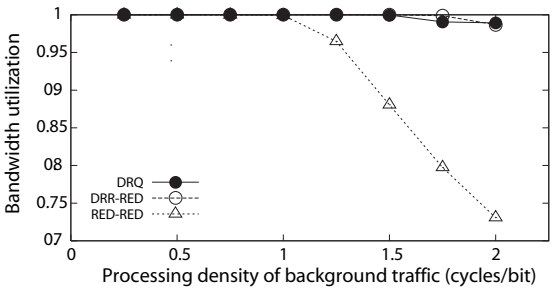
Figure 8 plots the average throughput of long-lived TCP flows, and the total bandwidth utilization of the bottleneck link as we increase the processing densities of short-lived TCP flows used to generate cross traffic. The results are very similar to those from the simulation without cross traffic; in all situations, RED-RED ensures the equal bandwidth sharing while DRQ and DRR-RED ensure equal sharing of CPU usages under the CPU-limited region and equal sharing of bandwidth under the BW-limited region. In this experiment, we can also find severe under-utilization of bandwidth by RED-RED under the CPU-limited region. As cross traffic



(a) RED-RED



(b) DRQ



(c) Bandwidth utilization

Fig. 8. Simulation result in dumbell topology with short-lived (web-like) background traffic. Dashed line implies proportional fair rate for each source group. Even with cross traffic, the result is similar to that from the simple model without background traffic.

helps increase the utilization of bandwidth, we find DRQ and DRR-RED to encourage higher utilization of the bottleneck bandwidth.

E. Parking lot simulation with multiple dual-resource bottleneck links

To increase realism in our simulation, we simulate the environment where multiple links can be dual-resource constrained. Figure 9 shows a parking lot topology with following link capacities $B_1 = B_2 = 50$ Mbps, $C_1 = C_2 = C_3 = 40$ Mcycles/s. We vary the bandwidth capacity of link $L3$ from 20 to 90 Mbps to create varying bottleneck conditions. Four class of long-lived TCP flows (denoted as $SG1$ to $SG4$) traverse links $L1$, $L2$ and $L3$ from sources 1-4 to sinks 1-4, respectively. One class of long-lived TCP flows from source 5 to sink 5 (denoted as $SG5$) traverses link L1 only and another class from source 6 to sink 6 (denoted as $SG6$) traverses link $L2$ only. Additional four sets of long-lived TCP flows traverse link $L3$ only (denoted as $SG7$ to $SG10$). We report

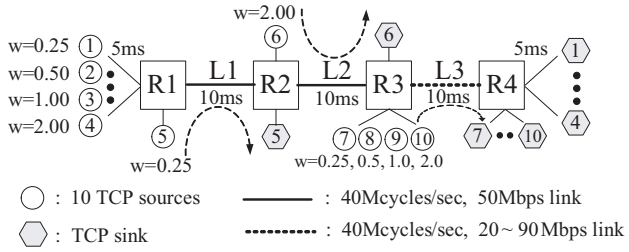
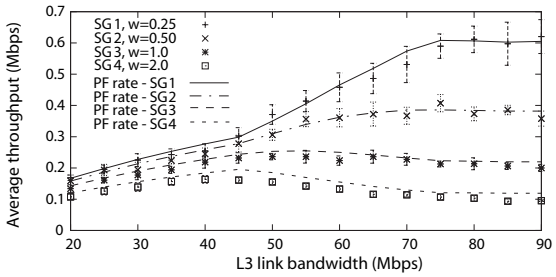


Fig. 9. Multiple link simulation scenario in parking lot topology.

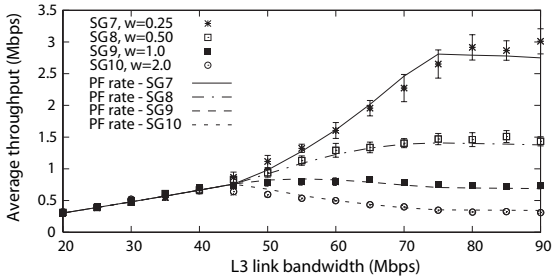
the average throughput of long-lived flows ($SG1$ to $SG4$, and $SG7$ to $SG10$) that pass through link $L3$ as we vary the link bandwidth of $L3$. On each path that long-lived TCP flows pass through, we add a small amount of short-lived background traffic to increase dynamics in the network traffic patterns. In this setup, each link runs one of the AQM schemes being evaluated.

Figures 10 (a) and (b) show the average throughput of the long-lived TCP flows that go through link $L3$. The average throughput values are shown with their corresponding 95% confidence intervals. We also plot with a dotted line the ideal PF rate for each class. The average throughput values closely follow the analytical PF rates.

Figure 11 shows the total throughput of all flows from $SG1$ to $SG10$ including those that traverse links $L1$ and $L2$. In this figure, we find that DRQ achieves a much higher total throughput than DRR-RED and RED-RED. Although we have consistently seen that DRR-RED and DRQ achieve higher utilization of bandwidth than RED-RED, this is the first time we find DRQ gets more throughput than DRR-RED. This happens because though DRR-RED performs similar to DRQ in the single bottleneck case, in principle, DRR and DRQ have different fairness notions especially for the flows with different numbers of hops. DRR has a goal to equally share the resource irrespective of path routes (which is the same goal as max-min fairness), but DRQ follows the proportional fairness criteria. In this topology, SG1-SG4 have a longer route and use more network resources than SG5, SG6, and SG7-SG10. So the total throughput difference comes from DRQ's use of proportional fairness that discriminates SG1-SG4 over the other flows because it uses more hops. The comparison of L3 link utilization shows that DRQ and DRR-RED use nearly the same amount, in Figure 12. This is because $L3$ lies in the CPU-limited region so the bandwidth usage of the flows is governed by the fair usage of CPU. This means that the bandwidth usage difference between SG1-SG4, and the other flows occur because of SG5 and SG6 flows that traverse only one link creating a bandwidth bottleneck on their corresponding link. However, it is incorrect to say that the fairness notion of DRQ always guarantees higher throughput than the fairness notion that DRR-RED follows as it is quite possible that there are other unique situations where DRR-RED gets more throughput (which is indeed shown in the next simulation). We leave as future study, studying the exact conditions where DRR-RED



(a)



(b)

Fig. 10. Throughput of TCP/DRQ in multiple link simulations: (a) SG1-SG4, (b) SG7-SG10. Even in cases where multiple links are dual-resource constrained, DRQ achieves proportional fairness.

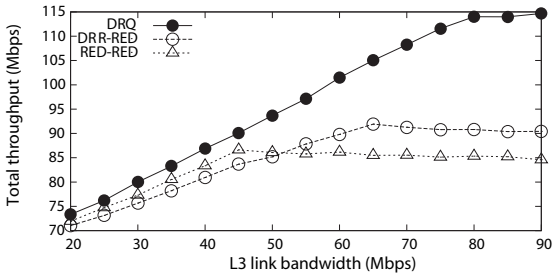


Fig. 11. Total throughput comparison between DRQ, DRR-RED, and RED-RED. It shows the sum of throughput for all flows from SG1 to SG10. In this setup, DRQ achieves the best throughput over the other schemes. This is because the PF rate allocation of DRQ installs fairness among flows that traverse different numbers of hops. RED-RED still consistently shows lower throughput.

can have better throughput than DRQ and vice versa.

F. Impact of incremental deployment

In this section, we examine the performance impact of incrementally deploying DRQ in the current Internet. The natural places where DRQ can be initially deployed are likely to be edges. This is because the current Internet trend is to keep the "middle" slim by pushing complicated tasks to the edges of the Internet. Thus, while core routers focus on moving packets as fast as possible, routers, proxies and gateways located at edges of the Internet perform various in-network processing on packets.

We consider an ISP environment where core routers and most edge routers are free from processing constraint but a small number of designated edge gateways handle TCP flows with in-network processing requirement. Figure 13 models one

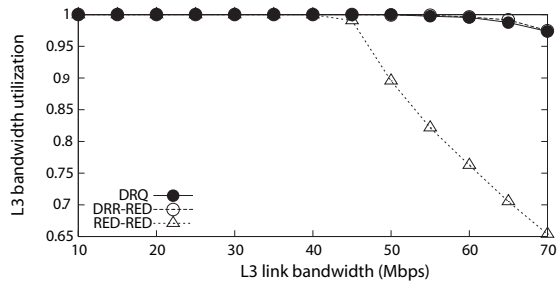


Fig. 12. L3 link bandwidth utilization comparison between DRQ, DRR-RED, and RED-RED.

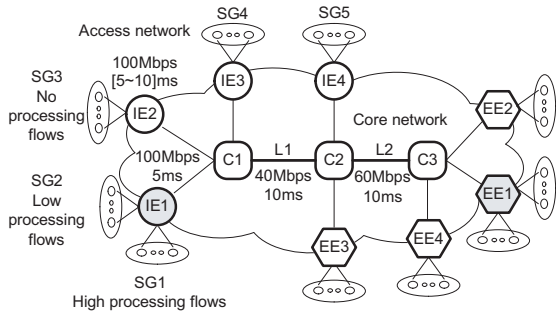


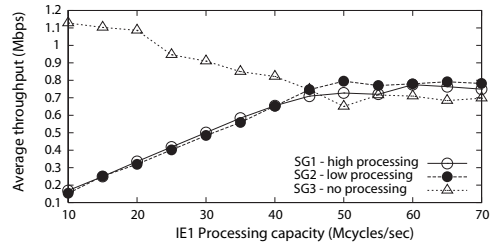
Fig. 13. Simulation topology for edge deployment

example of such environments. Our goal is to assess whether a small incremental deployment of DRQ gives any performance advantage while keeping the rest of the network intact. In the figure, SG's denote TCP source groups, each with ten TCP sources, and C1 to C3 denote core routers, IE1 to IE4 denote ingress edge routers and EE1 to EE4 denote egress edge routers, respectively. Flows from IE1 are routed through the shortest path to EE1 and etc. In this setup, all the routers except IE1 and EE1 are conventional packet routers with no CPU constraint and IE1 and EE1 are the media gateways/routers that may run in the CPU-limited region.

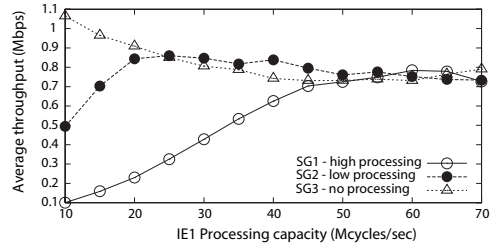
In this simulation, we consider two different source groups, $SG1$ and $SG2$, that traverse the same end-to-end path from IE1 to EE1. $SG1$ consists of flows with a high processing density (w_1 = 2.5) while $SG2$ consists of flows with a low processing density ($w_2 = 0.5$). Flows from $SG3$ to $SG5$ do not require any CPU processing. Note also that in this setup, $SG1$, $SG2$ and $SG3$ share the same bottleneck links ($L1$ and $L2$).

In Figure 14 we compare the average throughput of three source groups $SG1$, $SG2$ and $SG3$, and the aggregate throughput of SG1 and SG2 while varying the CPU capacity of IE1. As expected, with RED-RED, SG1 and SG2 achieve the equal bandwidth sharing irrespective of the CPU capacity, which give lower aggregate throughput. We can see that employing the DRQ (or DRR-RED) only at IE1 and EE1 serves more packets than RED-RED by preventing flows with high processing demands from starving those flows with lower processing demands. The aggregate throughput of DRR-RED is slightly higher than that of DRQ.

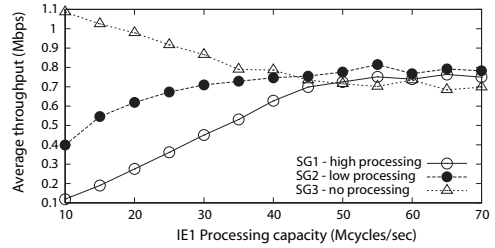
In the next simulation, we increase the processing density



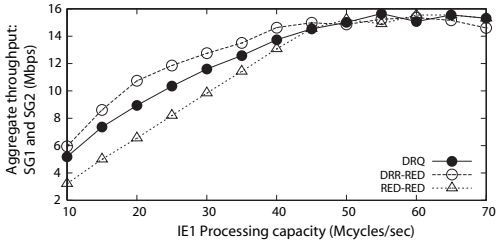
(a) RED-RED



(b) DRR-RED

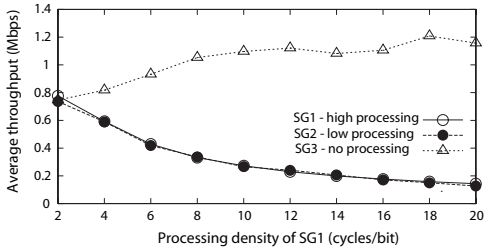


(c) DRQ

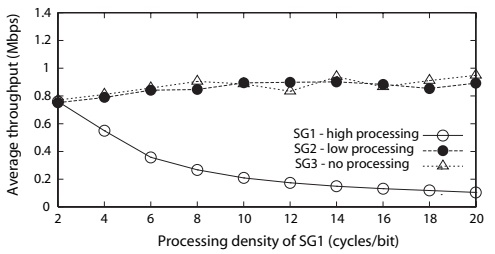


(d) Aggregate throughput of flows using IE1

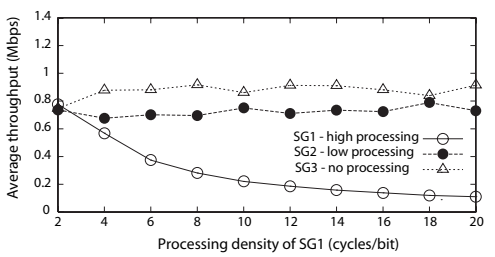
Fig. 14. Average and aggregate throughput of different schemes when some of the edge routers are under the CPU constraint. Even with a small number of edge routers employing DRQ, DRQ can provide higher throughput for the TCP flows with in-network processing requirements.



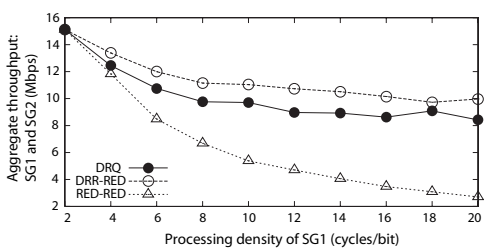
(a) RED-RED



(b) DRR-RED



(c) DRQ



(d) Aggregate throughput of flows using IE1

Fig. 15. Average and aggregate throughput when processing density values of neighboring flows (SG1) are increased. This shows that DRQ and DRR-RED, even in a partial deployment, achieve fairness in the CPU usage when some portion of flows has increasingly higher processing demands.

(w_1) of SG1 from 2 to 20 to see the impact of flows with high processing demands while fixing the processing density of SG2 to a relatively low value ($w_2 = 1.0$) in the same setup as the above and examine whether the fairness achieved by DRQ in a small number of routers contributes to increasing the total throughput of flows that pass through those routers. The CPU capacity of IE1 is fixed to 30 Mcycles/s.

In Figure 15 (a)-(c), we can verify that with DRQ or DRR-RED, the throughput of the high processing flows ($SG1$) is kept lower than that of the low processing flows ($SG2$) to balance its CPU usage with the low processing flows in IE1 and EE1 as we increase the processing density of the high processing flows. That results in much higher aggregate throughput for $SG1$ and $SG2$ than with RED-RED. In RED-RED, because the average throughput of both $SG1$ and $SG2$ flows is maintained to follow equal sharing, the CPU usage (not shown due to the space limitation) of $SG1$ is highly unfair to that of $SG2$ because $SG1$ will be using much CPU than $SG2$ when they have the same number of packets on the network. This phenomenon has a security implication where a malicious flow with a high processing density can easily starve out other competing flows from consuming CPU resources.

V. Conclusion

We have shown that under DRQ the average equilibrium throughput for TCP Reno sources becomes proportionally fair in the dual-resource environment. Moreover, such an equilibrium is unique and almost always exists, more specifically, as long as the per-flow steady-state average TCP window size (or per-flow bandwidth-delay product) is sustainable at a value greater than or equal to 2 packets.

DRQ significantly outperforms RED-RED scheme while maintaining a certain level of fairness. DRQ is scalable to a large number of flows and incrementally deployable since it does not require any change in end-host TCP stacks and builds on an ordinary RED queue.

The throughput gain achieved by DRQ over RED-RED comes mostly from two features of its marking scheme. The weighted marking at a CPU queue (in proportion to processing densities of flows) yields fair sharing of CPU cycles whenever this resource is scarce, which consequently, results in overall throughput increase. Second, the inter-resource marking, which is a unique feature of DRQ, gives extra penalty to flows that traverses more number of resources, which also results in overall throughput increase.

We show by simulation that even a partial deployment of DRQ is beneficial to increasing performance when it is implemented on a few selected locations where special in-network processing services (e.g., media translation, protocol conversion, security and PEP) are enabled.

References

[1] Y. Gottlieb and L. Peterson, "A comparative study of extensible routers," in *Proc. of IEEE OpenArch*, June 2002, pp. 51–62.

[2] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communications Review*, vol. 29, no. 2, pp. 7–23, April 1999.

[3] S. Chong, M. Shin, J. Mo, and H.-W. Lee, "Flow control with processing constraint," *IEEE Commun. Lett.*, vol. 9, no. 10, pp. 957–959, Oct. 2005.

[4] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM Computer Communications Review*, vol. 19, no. 4, pp. 1–12, 1989.

[5] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *Proc. of ACM SIGCOMM*, Sept. 1995, pp. 231–242.

[6] S. Floyd and V. Jacobson, "Random Early Detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.

[7] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, pp. 48–53, May 2001.

[8] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "On designing improved controllers for AQM routers supporting TCP flows," pp. 1726–1734, Apr. 2001.

[9] T. Wolf and M. A. Franklin, "CommBench - a telecommunications benchmark for network processors," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Austin, TX, Apr. 2000, pp. 154–162.

[10] G. Memik, M. Smith, and W. Hu, "NetBench: A benchmarking suite for network processors," in *Proc. of IEEE International Conference on Computer-Aided Design*, San Jose, CA, Nov. 2001.

[11] M. Tsai, C. Kulkarni, C. Sauer, N. Shah, and K. Keutzer, "A benchmarking methodology for network processors," in *Proc. of 1st Network Processor Workshop, 8th Int. Symp. on High Performance Computer Architectures (HPCA)*, Cambridge, MA, Feb. 2002.

[12] P. Pappu and T. Wolf, "Scheduling processing resources in programmable routers," in *Proc. of IEEE INFOCOM*, New York, NY, June 2002, pp. 104–112.

[13] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *J. of the Operational Research Society*, vol. 49, pp. 237–252, Apr. 1998.

[14] S. H. Low and D. E. Lapsley, "Optimization flow control I: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, pp. 861–875, Dec. 1999.

[15] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[16] F. P. Kelly, "Mathematical modelling of the internet," in *Mathematics Unlimited – 2001 and Beyond*, B. Engquist and W. Schmid, Eds. Berlin: Springer Verlag, 2001, pp. 685–702.

[17] V. Misra, W. B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. of ACM SIGCOMM*, Sept. 2000, pp. 151–160.

[18] Y. Liu, F. Presti, V. Misra, and D. Towsley, "Fluid models and solutions for large-scale IP networks," in *Proc. of ACM SIGMETRICS*, San Diego, CA, June 2003, pp. 151–160.

[19] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking*, vol. 11, pp. 525–536, Aug. 2003.

[20] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proc. of IEEE INFOCOM*, Anchorage, Alaska, Apr. 2001, pp. 1510–1519.

[21] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle, "Linear stability of TCP/RED and a scalable control," *Computer Networks*, vol. 43, pp. 633–647, Dec. 2003.

[22] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP," IETF RFC 2481, 1999.

[23] H. Han, C. V. Hollot, Y. Chait, and V. Misra, "TCP networks stabilized by buffer-based AQMs," in *Proc. of IEEE INFOCOM*, Hongkong, Mar. 2004, pp. 964–974.

[24] "ns-2 network simulator," 2000. [Online]. Available: http://www.isi.edu/nsnam/ns/

[25] S. Floyd, "Recommendations on using the gentle variant of RED," Mar. 2000. [Online]. Available: http://www.aciri.org/floyd/red/gentle.html

[26] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, "Variability in TCP round-trip times," in *Proc. of 3rd ACM SIGCOMM Conference on Internet Measurement Conference*, Oct. 2003, pp. 279–284.

[27] M. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: evidence and possible causes," *IEEE/ACM Trans. Networking*, vol. 5, pp. 835–846, Dec. 1997.

[28] V. Paxson and S. Floyd, "Wide-area traffic: The failure of poisson modeling," *IEEE/ACM Trans. Networking*, vol. 3, pp. 226–244, June 1995.

[29] K. Park, in *Self–Similar Network Traffic and Performance Evaluation*, W. Willinger, Ed. New York: John Wiley & Sons, 2000.

[30] H. Jiang and C. Dovrolis, "Passive estimation of TCP round-trip times," *ACM SIGCOMM Computer Communications Review*, vol. 32, pp. 75–88, July 2002.

[31] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proc. of ACM SIGMETRICS*, July 1998, pp. 151–160.

Appendix

Proof of Proposition 1: First, we show that the primal-optimal solution (r^*, θ^*, π^*) of Problem **P** with $\alpha_s = \frac{\sqrt{3/2}M_s}{\tau_s}$ forms an equilibrium point (x^*, σ^*, ρ^*, p^*, y^*, z^*, v^*, $\hat{y}^*$, $\hat{v}^*$) in the average Reno/DRQ network where $x^* = r^*, \sigma^* = \theta^*, \rho^* = \pi^*$ if θ^* and π^* satisfy **C1** and **C2**. Since $\sum_{s\in S(l)} r_s^* \le B_l$, $\forall l \in L$, and $\sum_{s\in S(k)} w_s^k r_s^* \le C_k$, $\forall k \in K$, Eqs. (7) and (8) imply that there exist $y_l^* \ge 0$, $\forall l \in L$, and $z_k^*, v_k^* \ge 0$, $\forall k \in K$. Thus, from Eqs. (11) and (14), there exist $\hat{y}_l^* = y_l^* \ge 0$, $\forall l \in L$, and $\hat{v}_k^* = v_k^* \ge 0$, $\forall k \in K$, whose specific values can be obtained from Eqs. (10) and (13) for $\rho_l^* = \pi_l^*$ and $\sigma_k^* = \theta_k^*$ since $0 \le \pi_l^* < 1$ and $0 \le \theta_k^* < 1$. Finally, for $x_s^* > 0$, Eqs. (9) and (15) yield $x_s^* = \frac{\sqrt{3/2}M_s}{\tau_s}(\sum_{k\in K(s)} w_s^k \sigma_k^* + \sum_{l\in L(s)} \rho_l^*)^{-1}$, which

also holds for (r^*, θ^*, π^*) by the Lagrangian optimality of (r^*, θ^*, π^*) in Eq. (6) with $\alpha_s = \frac{\sqrt{3/2}M_s}{\tau_s}$.

Next we prove the converse that if the average Reno/DRQ network has an equilibrium point (x^*, σ^*, ρ^*, p^*, y^*, z^*, v^*, $\hat{y}^*$, $\hat{v}^*$), then (x^*, σ^*, ρ^*) is the primal-dual optimal solution of the corresponding Problem $\mathbf{P}$ with $\alpha_s = \frac{\sqrt{3/2}M_s}{\tau_s}$ and, moreover, no other equilibrium points can exist in the network. By the duality theory [15], the necessary and sufficient condition for (x^*, σ^*, ρ^*) to be the primal-dual optimal solution of Problem $\mathbf{P}$ is that x^* is primal feasible, (σ^*, ρ^*) is dual feasible, and (x^*, σ^*, ρ^*) satisfies Lagrangian optimality and complementary slackness. First, Eqs. (7) and (8) imply that $\sum_{s \in S(l)} x_s^* \leq B_l$ for all $l \in L$ and $\sum_{s \in S(k)} w_s^k x_s^* \leq C_k$ for all $k \in K$, and Eq. (9) implies that $x^* \geq 0$. Thus, x^* is primal feasible. Second, the dual feasibility of (σ^*, ρ^*) is obvious since σ^* and ρ^* cannot be negative by the definition of REDs in Eqs. (10) and (13). Third, since Eq. (6) is the Lagrangian optimality condition of Problem $\mathbf{P}$, we need to show that

$$x_s^* = \frac{\alpha_s}{\sum_{k \in K(s)} w_s^k \sigma_k^* + \sum_{l \in L(s)} \rho_l^*}, \quad \forall\ s \in S \tag{33}$$

hold. Consider flow s. If we suppose that $\sigma_k^* = 0$, $\forall k \in K(s)$, and $\rho_l^* = 0$, $\forall l \in L(s)$, for this flow s, then $p_s^* = 0$ from Eq. (15), which contradicts Eq. (9) since it cannot have an equilibrium point satisfying $p_s^* = 0$. Thus, at least one of σ_k^*, $k \in K(s)$, and ρ_l^*, $l \in L(s)$, must be positive, which implies that $0 < p_s^* < 1$ from Eq. (15) and consequently $x_s^* > 0$ from Eq. (9). Therefore, from Eq. (9), we know that $\frac{M_s(1-p_s^*)}{\tau_s^2} = \frac{2}{3}\frac{{x_s^*}^2 p_s^*}{M_s}$ (N_s is cancelled out). Solving x_s^* from this equation by substituting Eq. (15) at equilibrium for p_s^*, we get Eq. (33) with $\alpha_s = \frac{\sqrt{3/2}M_s}{\tau_s}$, which concludes the Lagrangian optimality of (x^*, σ^*, ρ^*). Lastly, we need to show that $\sigma_k^*(\sum_{s \in S(k)} w_s^k x_s^* - C_k) = 0$ and $\rho_l^*(\sum_{s \in S(l)} x_s^* - B_l) = 0$ for all $k \in K$ and $l \in L$ to check complementary slackness of (x^*, σ^*, ρ^*). Consider arbitrary $k \in K$. If we suppose that $\sum_{s \in S(k)} w_s^k x_s^* - C_k < 0$, then $z_k^* = v_k^* = 0$ from Eq. (8). Thus, $\hat{v}_k^* = 0$ from Eq. (14) and consequently $\sigma_k^* = 0$ from Eq. (13), which concludes $\sigma_k^*(\sum_{s \in S(k)} w_s^k x_s^* - C_k) = 0$. Similarly, $\rho_l^*(\sum_{s \in S(l)} x_s^* - B_l) = 0$ for arbitrary $l \in L$. Therefore, we conclude that (x^*, σ^*, ρ^*) is the primal-dual optimal solution of Problem $\mathbf{P}$. Moreover, x^* is unique since the optimal solution of Problem $\mathbf{P}$ is unique. From Eq. (10), it is obvious that $y_l^* > \underline{b}_l$ if $\rho_l^* > 0$ and $0 \leq y_l^* \leq \underline{b}_l$ otherwise. The same argument can be applied to CPU queues from Eq. (13).