Multi-Resource Generalized Processor Sharing for Packet Processing

Wei Wang, Ben Liang, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Abstract*—Middleboxes have found widespread adoption in today's networks. They perform a variety of network functions such as WAN optimization, intrusion detection, and network-level firewalls. Processing packets to serve these functions often require multiple middlebox resources, *e.g.*, CPU and link bandwidth. Furthermore, different packet traffic flows may consume significantly different amounts of various resources, depending on the network functions that are applied. Multi-resource fair queueing is therefore needed to allow flows to share multiple middlebox resources in a fair manner. In this paper, we clarify the fairness requirements of a queueing scheme and present Dominant Resource Generalized Processor Sharing (DRGPS), a fluid flow-based fair queueing idealization that strictly realizes *Dominant Resource Fairness* (DRF) at all times. As a form of Generalized Processor Sharing (GPS) running on multiple resources, DRGPS serves as a benchmark that practical packet-by-packet fair queueing algorithm should follow. With DRGPS, techniques and insights that have been developed for traditional fair queueing can be leveraged to schedule multiple resources. As a case study, we extend Worst-case Fair Weighted Fair Queueing (WF^{2}Q) to the multi-resource setting and analyze its performance.

I. Introduction

Network appliances or "middleboxes" are ubiquitous today, especially in enterprise networks and datacenters [1], [2]. These middleboxes perform a wide range of critical network functions, including WAN optimization, intrusion detection, as well as network-level and application-level firewalls. Many middleboxes today already perform different network functions for different traffic flows, *e.g.,* WAN optimizations for the inter-site flows and intrusion detection for the external flows. Recent architecture innovations further advocate consolidating more functions onto the same device – in the form of software-defined middleboxes deployed in either commodity servers [1], [3], [4] or IaaS cloud instances [2] – for better infrastructure management and scalability.

Such a trend of function consolidation in middleboxes complicates resource scheduling. Unlike basic forwarding, most middlebox functions perform deep packet processing based on the packet content, and therefore require a variety of hardware resources, *e.g.,* CPU, memory bandwidth, link bandwidth, *etc*. Packet processing for these middlebox functions differs significantly in terms of the required hardware resources. For example, performing intrusion detection usually bottlenecks on the CPU resources, while forwarding a large amount of small packets via software routers congests the memory bandwidth [5]. Thus, depending on the network functions they go through, different traffic flows may consume vastly different amounts of middlebox resources [6]. A fair queueing algorithm is therefore highly desirable to schedule packets in a way such that each flow receives *predictable service isolation* independent of the other's demand.

Although single-resource scheduling has been extensively investigated in the past decades [7], multi-resource fair queueing has largely been an uncharted territory. In fact, it remains unclear what notion of fairness should a scheduling algorithm pursue, and how the fairness of a given scheduling scheme should be measured. To answer the questions above requires a fair queueing benchmark, which also plays a central role in resource scheduling. Indeed, when there is only a single resource to schedule (*i.e.,* link bandwidth), Generalized Processor Sharing (GPS) [8], [9] has been proposed as a benchmark. GPS implements strict max-min fairness in the idealized fluid model and guides the design of many packet-by-packet scheduling schemes, including WFQ, WF^{2}Q, SCFQ, SFQ, DRR, *etc.*, all of which are approximations to GPS in practice.

Despite the important role of a queueing benchmark, it remains unclear what the benchmark is when multiple resources are to be scheduled. It has been shown by Ghodsi *et al.* [6] that naively extending GPS to schedule multiple resources (*e.g.*, per-resource fairness and bottleneck fairness) will compromise service isolation, as the service received by a flow may be strategically manipulated by another. Ghodsi *et al.* further suggest a compelling packet-by-packet scheduling alternative based on Dominant Resource Fairness (DRF) [10], in which each flow receives roughly the same processing time on its most congested resource. However, without specifying a benchmarking scheme, it is unable to measure how good the design is or if there is any room for improvement.

In search for such a fair queueing benchmark for multiple resources, we apply the DRF notion to the idealized *fluid flow* traffic environment where flows receive service in arbitrarily small increments on all resources. The resulting idealization, referred to as *Dominant Resource GPS* (DRGPS), allows flows to receive equal service on their dominant resources at all times. DRGPS offers perfect service isolation that is immune to any strategic behaviours of flows, and is *work conserving* as well. It hence serves as an attractive fair queueing benchmark among various forms of multi-resource GPS (*e.g.*, GPS with per-resource fairness or bottleneck fairness). We also design a simple algorithm that can accurately emulate DRGPS by stamping service tags upon packet arrivals.

We believe in the importance of DRGPS in multi-resource scheduling for a number of reasons. First, as a benchmark, DRGPS can be used to measure the performance of a given queueing scheme. We consider two fairness metrics, the *Absolute Fairness Bound* (AFB) and the *Relative Fairness Bound* (RFB). Though both can be similarly defined as in the single-resource scenario [11], AFB might not be well justified under the multi-resource setting and is usually hard to obtain. As for RFB, we show that, counter-intuitively, a packet service discipline may achieve better fairness performance in the multi-resource setting as compared with the single-resource counterpart.

More importantly, we see that DRGPS guides the design of practical packet-by-packet service disciplines. With it, techniques and insights that have been developed for fair queueing (*e.g.*, [11] Ch. 9) could be borrowed into multi-resource scheduling design. We present our findings via both high-level discussions and concrete case studies. We show that, by emulating DRGPS, well-known fair queueing algorithms such as WFQ [8], [9], WF^{2}Q [12], and FQS [13] will have immediate multi-resource extensions. Focusing on multi-resource WF^{2}Q only, we analyze its performance and derive novel bounds on its fairness, measured by the multi-resource RFB. Many practical considerations are also discussed in this paper. Based on the insights derived from DRGPS, it is possible to leverage the substantial effort that has been put forth on fair queueing to the new, yet critical, multi-resource environment in today's networks.

II. Fair Queueing and Its Design Objectives

For a queueing discipline, one central issue to be addressed is the notion of fairness. In essence, what queueing algorithm is deemed to be fair? Despite the pioneering work of Ghodsi *et al.* [6], the answer to this question remains fuzzy in the middlebox environment, where traffic flows require multiple hardware resources. In this section, we briefly review those desired scheduling properties that are uniformly required in the fair queueing literature [7], [8]. We extend them to the multi-resource environment and define multi-resource fair scheduling.

An essential property of fair queueing is to offer *predictable service isolation*. In single-resource queueing, for example, when link bandwidth is the only resource to schedule, each of the n backlogged flows should receive $1/n$ bandwidth share. Weighted fairness generalizes this property, such that each flow i is assigned a weight w_i and will receive $w_i/\sum_j w_j$ bandwidth share.

Property 1 (Service isolation): Suppose there are n flows that are backlogged. A multi-resource queueing scheme offers predictable service isolation if for each flow i, the received service is at least at the level when *every resource* is equally allocated. Further, when flow i is assigned a weight w_i, then the received service is at least at the level when every resource is allocated in proportion to the weight, *i.e.*, flow i receives $w_i/\sum_j w_j$ allocated share on each resource.

Note that, under multi-resource fair queueing, having the same service share does not imply the same resource allocation, as resources that are allocated might not be fully utilized.

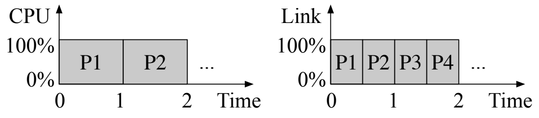


Fig. 1. Packets may consume different amounts of resources, and may have different processing rates on different resources.

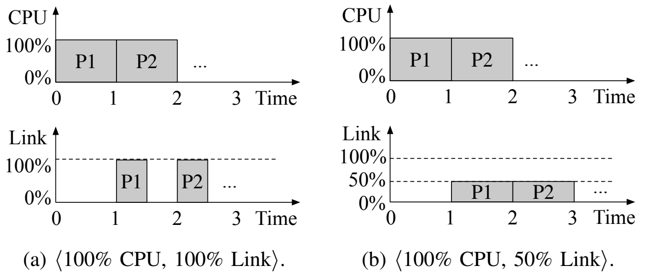


Fig. 2. Different resource allocations may lead to the same service for a flow. (a) Throughput mismatch between CPU and link bandwidth in Fig. 1 makes it impossible to fully utilize the allocated link bandwidth. The received service is 1 packet per time unit. (b) Reducing the bandwidth allocation to 50% will not reduce the received service.

To see this, consider a flow whose traffic needs encryption before transmission, and hence more time is needed for the CPU to process a packet than the link to transmit it. As shown in Fig. 1, when 100% of each resource is applied, the CPU processing time is twice the link transmission time. Consider two allocations, one allocating ⟨100% CPU, 100% Link⟩, the other allocating ⟨100% CPU, 50% Link⟩. As illustrated in Figs. 2a and 2b, under two different allocations, the flow receives the same service of 1 packet per unit time. For this reason, to offer service isolation, it is not always necessary to equally divide every resource among all traffic flows.

In fact, a naive scheduling scheme that equally divides all resources among traffic flows (referred to as per-resource fairness in [6]) is vulnerable to strategic behaviours. As noted by Ghodsi *et al.* [6], by artificially inflating their demand for resources they do not need, some flows may receive better service, at the cost of other flows. To discourage such strategic behaviours, we further require *truthfulness* in a scheduling scheme.

Property 2 (Truthfulness): A multi-resource queueing scheme is truthful if no flow can receive better service (*i.e.*, finish faster) by misreporting the amount of resources it requires.

Both service isolation and truthfulness have been noted by Ghodsi *et al.* [6] as the design objective of fair queueing[1]. While they ensure the basic requirements of fairness, we believe resource utilization is another important dimension to evaluate a fair queueing scheme. We therefore introduce *work conservation* to reflect such a concern of queueing efficiency.

Property 3 (Work conservation): A multi-resource queueing scheme is work conserving if no resource that could be used to serve a busy flow is wasted in idle. In other

[1]Service isolation is defined in another form in [6], called *share guarantee*. Our definition here is more intuitive and precise.

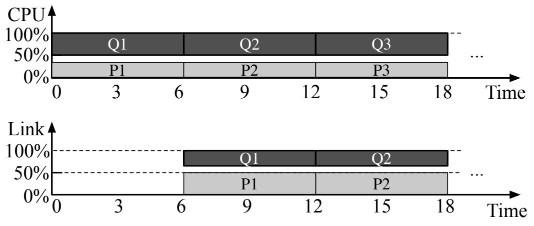


Fig. 3. This schedule offers service isolation but is non-work-conserving. Here, P1, P2,... are packets of flow 1, while Q1, Q2,... are packets of flow 2.

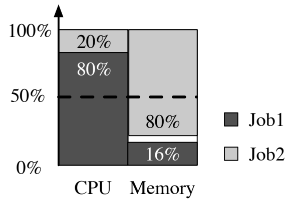


Fig. 4. An example of a DRF allocation, under which the dominant shares of two jobs are equalized.

words, whenever there is a flow that is backlogged, at least one resource is fully utilized.

One may conclude that work conservation is implied from service isolation. However, this is not true. Consider two flows: packets of flow 1 require 2 time units for CPU processing and 3 time units for link transmission, while packets of flow 2 requires 3 time units for CPU processing and 2 time units for transmission. As shown in Fig. 3, the schedule in which flow 1 and 2 receive $\langle$1/3 CPU, 1/2 Link$\rangle$ and $\langle$1/2 CPU, 1/3 Link$\rangle$ respectively offers service isolation, but it is not work conserving, since 1/6 CPU and 1/6 link bandwidth are wasted in idle, even though flow 1 and 2 are backlogged.

Having all three properties above defines a fair queueing scheme. When there is a single resource to schedule (*i.e.,* link bandwidth), all these requirements are perfectly met by GPS [8], [9] in the idealized fluid model, which can then be used as a fair queueing benchmark. Many well-known packet-by-packet scheduling disciplines, such as WFQ, WF^{2}Q, SCFQ, SFQ, DRR, *etc.*, are then designed to approximate GPS.

However, having multiple resources to schedule significantly complicates the queueing design. In fact, Ghodsi *et al.* [6] show that several natural scheduling algorithms for middleboxes, such as *bottleneck fair queueing* and *per-resource fair queueing*, all fail to provide service isolation. Ghodsi *et al.* then suggest a promising packet-based alternative that achieves Dominant Resource Fairness (DRF) over time. However, it remains largely unknown if there is a GPS-like idealized schedule that can be used as a benchmark in the multi-resource setting, without which one cannot properly evaluate what a packet-based schedule should approximate. We answer this question in the next section.

III. Dominant Resource GPS (DRGPS)

In this section, we present Dominant Resource GPS (DRGPS), a fluid flow-based service discipline that implements strict DRF allocation at all times. We show that DRGPS possesses all important fairness properties given in the previous section, and hence serves as an ideal fair queueing benchmark based on which practical packet-based scheduling could be designed and measured.

A. Dominant Resource Fairness (DRF)

Since DRGPS is designed to implement DRF allocation, we briefly review DRF and some relevant concepts. DRF was originally designed for job scheduling in datacenters [10]. In that setting, the equivalence of a flow is a computing job, and the equivalence of a packets is a job's task. Running a job's task may consume multiple hardware resources, taking up a fractional share of each of the resources. The *dominant resource* of a job is the resource whose share by the job is maximum among all resources consumed by the job. A job's *dominant share* is the share of its dominant resource. For example, consider a computing cluster with 100 CPUs and 200 GB memory in total, and a job whose tasks require 30 CPUs and 40 GB memory. The dominant resource of this job is CPU, and the associated dominant share is 3/10 (as compared to 1/5 for memory).

The DRF allocation implements the max-min fairness on every job's dominant resource. It tries to "equalize" the dominant share allocated to each job. As an example, we refer to Fig. 4, where two jobs run tasks with resource profiles $\langle 10 \text{ CPUs}, 4 \text{ GB}\rangle$ and $\langle 2 \text{ CPUs}, 16 \text{ GB}\rangle$ in a cluster with 100 CPUs and 200 GB memory. The dominant resource of job 1 is CPU while the dominant resource of job 2 is memory. The DRF allocation then allocates $\langle 80 \text{ CPUs}, 32 \text{ GB}\rangle$ to job 1 (*i.e.,* 8 tasks) and $\langle 20 \text{ CPUs}, 160 \text{ GB}\rangle$ to job 2 (*i.e.,* 10 tasks), through which the dominant share of the two jobs is equalized to 4/5. The attractiveness of DRF lies in a set of highly desired fairness properties of the resulting allocation, notably the share incentive, Pareto efficiency, truthfulness, and envy-freeness [10], [14].

Achieving the DRF allocation via a queueing scheme imposes nontrivial challenges. As pointed out in [6], in datacenter environments, it is typical to have many more resources than active jobs, and one can simply divide resources across jobs. However, this is not the case in a middlebox, where resources are much fewer than the number of active flows, and must be shared in *time* instead of *space*. We will discuss in the following subsection how such a gap can be effectively bridged via a multi-resource fluid flow model.

B. Multi-Resource Fluid Flow Model

In practice, resources are scheduled in sequence to process a packet. For example, CPU is often scheduled first for packet processing, followed by memory bandwidth, which is consumed to forward processed packets to the Network Interface Card (NIC), in which the link bandwidth is scheduled for packet transmission.

Depending on the functions it goes through, a packet may consume different amounts of middlebox resources, each requiring different processing times. As shown in the previous example of Fig. 1, for a packet that needs encryption before transmission, the packet processing rate on CPU is only half the rate on the link bandwidth. Such service mismatch makes it impossible to fully utilize the link resource. We see in Fig. 2a that even if 100% bandwidth is allocated for packet transmission, the link remains idle for 50% of the time. As a result, the service received is bottlenecked on CPU. This implies that allocating full link bandwidth for packet transmission is unnecessary. Instead, if we only allocate 50% bandwidth and assume that packets can be served in *arbitrarily small increments* on the link resource, we will have a scheduling outcome shown in Fig. 2b. We see that the received service remains 1 packet per time and is the same as that in Fig. 2a.

Generally speaking, the discrepancy among processing rates on different resources leads to allocation waste, as the received service is bottlenecked by the minimum one across all resources. Therefore, resources should be allocated in a way such that packets are processed at the same rate. This can be exactly realized in the *multi-resource fluid model*, where packets receive the service in *infinitesimally small increments on every resource*.

Formally, given some packet, let τ_r be the *processing time* on resource r when 100% of resource r is allocated to process it. The *full service rate* on resource r is then $1/\tau_r$. Now let f_r be the share (fraction) of resource r allocated. The corresponding service rate on resource r is f_r/τ_r. A *non-wasteful* allocation should have a uniform service rate across all resources, *i.e.,*

$$f_r/\tau_r = f_{r'}/\tau_{r'} \tag{1}$$

for all r and r'. In the previous example, $\langle \tau_1, \tau_2 \rangle = \langle 1 \text{ CPU time}, 0.5 \text{ Link time} \rangle$ (see Fig. 1). The allocation $\langle 100\% \text{ CPU}, 50\% \text{ Link} \rangle$ in Fig. 2b is non-wasteful with a uniform service received on both CPU and link bandwidth (*i.e.,* 1 packet per time).

Here we make a key observation on non-wasteful allocations. Since resources are processed at a uniform service rate, it is equivalent to considering all of them to be scheduled *in parallel*. Fig. 5 shows an equivalent representation of Fig. 2b, where both CPU and link bandwidth are scheduled simultaneously. Note that such a parallel resource consumption model is only possible in the idealized multi-resource fluid model. With it, resource scheduling in time has an equivalent representation of resource allocation in space, which we will use in the next subsection.

C. Dominant Resource Generalized Processor Sharing

DRGPS implements exact DRF allocation in the fluid model, at all times. In particular, for any packet, its *dominant resource* is simply the one that needs the most time to process when using 100% of the resource, *i.e.,* the one with the maximum processing time τ_r. In Fig. 1, both P1's and P2's dominant resource is CPU. The *dominant share* is then defined as the fraction of the dominant resource allocated, and is 100% for P1 and P2 in Fig. 5. At any given time, DRGPS seeks to "equalize" the dominant share of packets across all flows (with appropriate weights in the case of weighted fairness), leading to an exact DRF allocation in the fluid flow model.

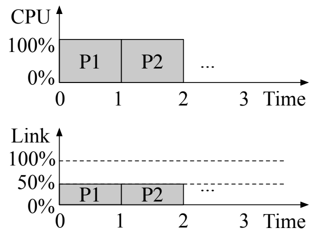


Fig. 5. An equivalent view of Fig. 2b, where resources are scheduled in parallel.

TABLE I
RESOURCE PROFILES OF PACKETS IN TWO FLOWS.

Packet	Flow	Arrival Time	$\langle$CPU, Link$\rangle$
P1	Flow 1	0	$\langle 4, 2 \rangle$
Q1	Flow 2	1	$\langle 1, 1 \rangle$
Q2	Flow 2	2	$\langle 1, 3 \rangle$

As an example, consider two equally weighted flows requiring both CPU and link bandwidth. Flow 1 has one packet P1 to serve at time 0, while flow 2 has two, Q1 and Q2, arriving at times 1 and 2, respectively. P1's resource profile is $\langle 4, 2 \rangle$. That is, it takes 4 units of time for CPU to process P1, and 2 for Link, both working with full utilization. The resource profiles of Q1 and Q2 are $\langle 1, 1 \rangle$ and $\langle 1, 3 \rangle$, respectively. Table I gives a brief summary.

The resulting DRGPS allocation over time is given in Table II and is also depicted in Fig. 6. At time 0, only P1 is ready for service. Based on its resource profile, DRGPS allocates 100% CPU and 50% link bandwidth (*i.e.,* $\langle 1 \text{ CPU}, 1/2 \text{ Link} \rangle$), leading to a maximum uniform service rate 1/4. This allocation remains until time 1, at which time Q1 is ready for flow 2, competing with P1 for both CPU and link bandwidth. Since CPU is the dominant resource of both packets, it is evenly allocated to each of them. As a result, P1 receives $\langle 1/2 \text{ CPU}, 1/4 \text{ Link} \rangle$ while Q1 receives $\langle 1/2 \text{ CPU}, 1/2 \text{ Link} \rangle$, where the link bandwidth is allocated in proportion to the resource profile of the two packets. With this allocation, it takes 2 time units to serve Q1. Hence, at time 3, Q2 replaces Q1 and competes with P1 for resources. Unlike P1, Q2's dominant resource is the link bandwidth. DRGPS then allocates $\langle 2/3 \text{ CPU}, 1/3 \text{ Link} \rangle$ to P1 and $\langle 2/9 \text{ CPU}, 2/3 \text{ Link} \rangle$ to Q2, under which their dominant shares are equalized and the throughput is maximized. Such an allocation maintains until P1 gets fully served at time 6. From then on, Q2 is the only packet to serve. It is then allocated $\langle 1/3 \text{ CPU}, 1 \text{ Link} \rangle$ and finishes at time 7.

We now formalize the description of DRGPS. Let us define $\mathcal{B}(t)$ as the set of flows that are backlogged at time t. These flows are competing for m middlebox resources. For flow $i \in \mathcal{B}(t)$, let w_i be its weight, and $\langle \tau_{i,1}, \ldots, \tau_{i,m} \rangle$ be the resource profile of its packet *currently being served*, where $\tau_{i,r}$ is the processing time on resource r (assuming full utilization). The

TABLE II
THE RESULTING DRGPS SCHEDULING FOR THE EXAMPLE OF TABLE I.

Time Interval	Flow 1		Flow 2	
	Packet	Allocation	Packet	Allocation
[0, 1)	P1	$\langle 1, 1/2\rangle$	N/A	N/A
[1, 3)	P1	$\langle 1/2, 1/4\rangle$	Q1	$\langle 1/2, 1/2\rangle$
[3, 6)	P1	$\langle 2/3, 1/3\rangle$	Q2	$\langle 2/9, 2/3\rangle$
[6, 7)	N/A	N/A	Q2	$\langle 1/3, 1\rangle$

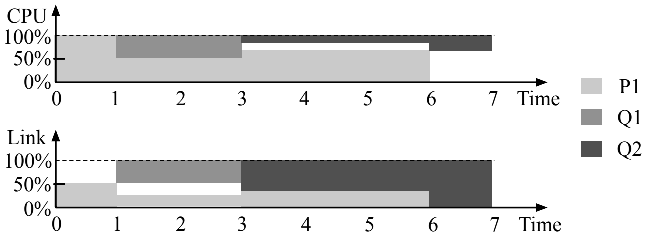


Fig. 6. An pictorial illustration of DRGPS for the example of Table I.

dominant resource of this packet is denoted as

$$r_i^* = \arg\max_r \tau_{i,r} \ . \tag{2}$$

With DRGPS, the current packet in flow i receives the DRF allocation $\langle f_{i,1}^t, \dots, f_{i,m}^t\rangle$, where $f_{i,r}^t$ is the share of resource r allocated at time t. The computation of $\langle f_{i,1}^t, \dots, f_{i,m}^t\rangle$ follows directly from [10], [14]. In particular, when $\tau_{i,r} > 0$ for all flow i and resource r, the resulting dominant share has a simple form:

$$f_{i,r_i^*}^t = \frac{w_i}{\max_r \sum_{j\in\mathcal{B}(t)} w_j \bar{\tau}_{j,r}} \ , \tag{3}$$

where $\bar{\tau}_{j,r} = \tau_{j,r}/\tau_{j,r_j^*}$ is the *normalized processing time* on resource r for flow j. We see that all flows' *normalized dominant shares* are equalized under this allocation, *i.e.,*

$$f_{i,r_i^*}^t / w_i = f_{j,r_j^*}^t / w_j, \quad \forall i,j \in \mathcal{B}(t) \ . \tag{4}$$

The share of a non-dominant resource r is determined based on the dominant share as follows. Given an allocation $\langle f_{i,1}^t, \dots, f_{i,m}^t\rangle$, the service received (*i.e.,* processing rate) s_i^t is bounded by the minimum one on all resources, *i.e.,*

$$s_i^t(\langle f_{i,1}^t, \dots, f_{i,m}^t\rangle) = \min_r f_{i,r}^t/\tau_{i,r} \ . \tag{5}$$

To avoid unnecessary allocation waste, one should ensure that

$$f_{i,r}^t/\tau_{i,r} = f_{i,r'}^t/\tau_{i,r'} \ , \tag{6}$$

for all resource r and r'. Now taking $r' = r_{i^*}$, we have

$$f_{i,r}^t = f_{i,r_i^*}^t \bar{\tau}_{i,r}, \text{ for all } r. \tag{7}$$

Next, we verify that DRGPS possesses all desired fairness properties mentioned in Sec. II. Therefore, DRGPS offers an ideal fair queueing benchmark in the multi-resource setting.

First, it offers service protection.

Proposition 1 (Service isolation): Under DRGPS, at any given time t, the service received by a backlogged flow is at least at the level that it would have received by allocating resources in proportion to its weight, *i.e.,* for all flow $i \in \mathcal{B}(t)$,

$$s_i^t(\langle f_{i,1}^t, \dots, f_{i,m}^t\rangle) \ge s_i^t\left(\left\langle \frac{w_i}{\sum_j w_j}, \dots, \frac{w_i}{\sum_j w_j}\right\rangle\right) . \tag{8}$$

Second, no scheduler manipulation is possible in DRGPS:

Proposition 2 (Truthfulness): Under DRGPS, no flow can receive better service by misreporting the amount of resources it requires. Formally, for any flow $i \in \mathcal{B}(t)$, let $\langle f_{i,1}^t, \dots, f_{i,m}^t\rangle$ be the resulting allocation when i truthfully reports its packet profile $\langle \tau_{i,1}, \dots, \tau_{i,m}\rangle$, and let $\langle \hat{f}_{i,1}^t, \dots, \hat{f}_{i,m}^t\rangle$ be the resulting allocation when i misreports. We then have

$$s_i^t(\langle f_{i,1}^t, \dots, f_{i,m}^t\rangle) \ge s_i^t(\langle \hat{f}_{i,1}^t, \dots, \hat{f}_{i,m}^t\rangle) \ . \tag{9}$$

The proofs of Proposition 1 and 2 follow directly from the properties of sharing incentive and strategy-proofness of the DRF allocation in [14].

Third, DRGPS is work conserving:

Proposition 3 (Work conservation): Under DRGPS, at least one resource is fully utilized when there is a packet that has not yet finished service. Formally, at any time t when $\mathcal{B}(t) \neq \emptyset$, there exists a resource r, such that

$$\sum_{i\in\mathcal{B}(t)} f_{i,r}^t = 1 \ . \tag{10}$$

Proof: The proof can be obtained from combining the Pareto optimality of DRF [14], and the work of Gutman et al. [15] (Definitions 7, 8, 9 and Proposition 3). The details are omitted due to space limitation. ■

Other beneficial properties of DRGPS includes *envy-freeness*, that no flow prefers the other's service allocation to its own (assuming equal weight), *single-resource fairness*, that DRGPS reduces to GPS when scheduling a single resource, and *population monotonicity* [10], [14], that a flow will not see the decrease of its service when other flows are finished and relinquish their required resources.

D. Emulating DRGPS in Real-Time

So far, we have presented the theoretical model of DRGPS. As we will show in Sec. V, DRGPS can also be used to design practical packet-by-packet queueing schemes. A simple algorithm is therefore needed to accurately emulate DRGPS. Similar to GPS, such an emulation can be achieved based on the evaluation of a time function that represents the progress of work in the system. This function $v(t)$, called *virtual time*, has a rate of increase in time equal to that of the normalized service received by any backlogged flow on the dominant resource, *i.e.,*

$$v'(t) = f_{i,r_i^*}^t / w_i, \quad \forall i \in \mathcal{B}(t) \ . \tag{11}$$

Below we provide the formal definition.

Definition 1: The virtual time of DRGPS is defined as the function $v(t)$ which satisfies the following:

$$\begin{aligned} v(t) &= 0, \quad \text{if} \quad \mathcal{B}(t) = \emptyset \text{ or } t = 0 \ , \\ \frac{\mathrm{d}}{\mathrm{d}t} v(t) &= \frac{1}{\max_r \sum_{j\in\mathcal{B}(t)} w_j \bar{\tau}_{j,r}}, \quad \text{o.w.} \end{aligned} \tag{12}$$

The definition above indicates that, when there is no backlogged flow in the system, the virtual time is reset to 0, which is equivalent to setting $t = 0$. It is hence without loss of generality to focus on the *busy period*, in which there are always packets to process, *i.e.,* $\mathcal{B}(t) \neq \emptyset$ for all t in the period.

We now consider a flow i and its sequence of packets. Denote the kth packet of the sequence by p_i^k, its arrival time by a_i^k, and the time it finishes service under DRGPS by d_i^k. Let $\langle \tau_{i,1}^k, \dots, \tau_{i,m}^k \rangle$ be the packet's profile, where $\tau_{i,r}^k$ is the required processing time on resource r. We further define S_i^k as the virtual time when packet p_i^k starts to receive service, and F_i^k as the virtual time when p_i^k finishes service, *i.e.,*

$$F_i^k = v(d_i^k), \quad k = 1, 2, \dots \tag{13}$$

We refer to S_i^k and F_i^k as the *virtual starting and finishing times* of packet p_i^k, respectively. The following proposition reveals their relationship.

Proposition 4: Under DRGPS, for every flow i, its virtual starting and finishing times satisfy the following relationship:

$$\begin{aligned} S_i^k &= \max\{F_i^{k-1}, v(a_i^k)\} \ , \\ F_i^k &= \tau_{i,r_i^{k*}}^k / w_i + S_i^k \ , \end{aligned} \tag{14}$$

where $F_i^0 = 0$ for all flow i.

Proof: Let $T_i(t_1, t_2)$ be the total processing time flow i receives in the time interval (t_1, t_2). Let b_i^k be the time that p_i^k starts to receive service, *i.e.,*

$$b_i^k = \max\{a_i^k, d_i^{k-1}\} \ , \tag{15}$$

where we define $d_i^0 = 0$. Note that all of the previous packets of flow i are completely served by b_i^k. We have

$$\begin{aligned} \tau_{i,r_i^{k*}}^k / w_i &= T_i(b_i^k, d_i^k)/w_i \\ &= v(d_i^k) - v(b_i^k) \ , \end{aligned} \tag{16}$$

where the last equality holds because flow i is backlogged during (b_i^k, d_i^k). By the definition of $v(t)$, we see that it is increasing during the busy period. Therefore, from (15), we have

$$v(b_i^k) = \max\{v(a_i^k), v(d_i^k)\} \ . \tag{17}$$

Substituting (17) to (16), we see the statement holds. ■

Proposition 4 provides a simple iterative algorithm to accurately emulate DRGPS in real-time. Upon the arrival of each packet, two service tags, the virtual starting time and the virtual finishing time, are stamped, with their values iteratively computed from (14). These service tags contain all the scheduling information of a packet in the DRGPS system (*i.e.,* when the packet gets served and when it finishes) with which the scheduling details are easily reconstructed.

Note that though DRGPS can be accurately emulated, it cannot be implemented unless flows are served in arbitrarily small increments. In contrast, practical service disciplines must schedule packets as discrete entities. Under this constraint, how to closely approximate DRGPS is a major challenge. This challenge echos the significant efforts that have been put forth to approximate GPS in the single-resource setting. We see in the next section that, with DRGPS, these efforts can be leveraged to schedule multiple resources.

IV. Packet-Based Multi-Resource Fair Queueing

To closely approximate DRGPS, practical packet-by-packet queueing scheme should schedule packets in a way such that the DRF allocation is achieved over time. Two fundamental questions therefore arise: (1) How do we measure the performance gap between DRGPS and a packet-based scheme? (2) How can a packet-by-packet alternative be designed to closely track DRGPS? We take some initial steps towards answering these questions, where we start off by elaborating the performance measures.

A. Fairness Measures

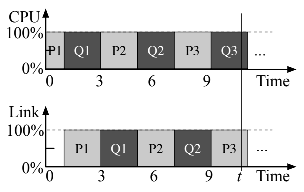
Fairness is our primary concern. When there is a single resource to schedule, two fairness metrics, Absolute Fairness Bound (AFB) and Relative Fairness Found (RFB), are widely adopted in the fair queueing literature [11]. Both can be extended to the multi-resource setting.

Absolute Fairness Bound (AFB): AFB compares the work progress of a packet-by-packet queueing scheme (real system) with that in a referencing GPS system that receives the same packet arrival process as in the real system. For any given flow, AFB compares the service this flow receives in both the real system and the referencing GPS system. The maximum service gap is then used as a metric to measure the fairness of a real system [11].

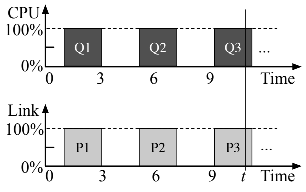
This idea may be directly extended to the multi-resource setting, where a referencing DRGPS is maintained to track the service received, which is then used to compare with the service received in the real system. However, such a comparison may be unfair. Since resources are processed *in parallel* under DRGPS, its work progress may be far ahead of that in real systems, in which resources are scheduled *in sequence*. As a result, with AFB, some times it is hard to tell if the discrepancy of work progress on the dominant resource is due to unfairness of the scheduling algorithm itself or the intrinsic advantage of the parallel resource processing model adopted in DRGPS. Moreover, AFB is usually hard to obtain, as it requires more involved analysis, even in the single-resource setting [11]. It is hence less popular as compared with RFB in the fair queueing literature [16], [7], [17].

Relative Fairness Bound (RFB): RFB is a another widely adopted fairness metric in the fair queueing literature [16], [11]. Without maintaining a referencing GPS system, RFB measures the fairness of a real system by bounding the gap between service received by a pair of backlogged flows. This idea can also be naturally extended to the multi-resource setting, which we shown below.

From the perspective of DRGPS, a service discipline is fair if it equalizes all flows' service received on their dominant resources (see (4)) in all time intervals. This is equivalent to allocating equal processing time on the dominant resource across backlogged flows. Based on this intuition, we define



(a) Real scheduling.



(b) Service on the dominant resource.

Fig. 7. Even packets are scheduled as entities, the gap of service that two flows received on their dominant resources could be 0.

RFB as the maximum gap of the normalized service received on the dominant resource between two backlogged flows.

Definition 2: For any packet arrival process, let $T_i(t_1, t_2)$ be the aggregate service (processing time) flow i receives on its *dominant resource* in the time interval (t_1, t_2). Let $\mathcal{B}(t_1, t_2)$ be the set of flows that are backlogged in (t_1, t_2). We define the Relative Fairness Bound (RFB) as

$$R = \sup_{t_1, t_2; i, j \in \mathcal{B}(t_1, t_2)} \left| \frac{T_i(t_1, t_2)}{w_i} - \frac{T_j(t_1, t_2)}{w_j} \right| . \quad (18)$$

RFB is well justified in the multi-resource setting. The service gap accurately reflects the fairness of the scheduler. Intuitively, RFB measures the degree to which the DRF allocation is violated. The smaller the measure is, the more closely the scheme approximates DRGPS, and the fairer the scheduler is. As an extreme example, we see that the RFB of DRGPS is 0.

Proposition 5: The RFB of DRGPS is 0. In particular, we have

$$\frac{T_i(t_1, t_2)}{w_i} = \frac{T_j(t_1, t_2)}{w_j} , \quad (19)$$

for any two flows $i, j \in \mathcal{B}(t_1, t_2)$,

Proof: It is easy to verify that under DRGPS, for any flow i, we have

$$T_i(t_1, t_2) = \int_{t_1}^{t_2} f_{i, r_i^*}^t \mathrm{d}t . \quad (20)$$

By (4), we see that for any two flows $i, j \in \mathcal{B}(t_1, t_2)$,

$$\frac{T_i(t_1, t_2)}{w_i} - \frac{T_j(t_1, t_2)}{w_j} = \int_{t_1}^{t_2} \left(\frac{f_{i, r_i^*}^t}{w_i} - \frac{f_{j, r_j^*}^t}{w_j} \right) \mathrm{d}t = 0, \quad (21)$$

for all t. This implies that the RFB of DRGPS is 0. ■

There is an important result regarding RFB in the traditional fair queueing literature. The well-known work of Golestani [16] shows that no packet-by-packet queueing scheme can achieve zero RFB, as packets are scheduled as discrete entities. Golestani further gives a lower bound on the maximum service gap between a pair of busy flows. This result, however, no longer holds in the multi-resource setting, which we show via a counter-example.

Consider two equally weighted traffic flows that keep sending packets, where flow 1 sends P1, P2, ... while flow 2 sends Q1, Q2, ... Except packet P1, which requires 1 time unit for CPU processing and 2 for link transmission, each of the other packets requires 2 time units on both CPU and link transmission. In this case, we can view link bandwidth as the dominant resource of flow 1, while CPU is the dominant resource of flow 2. As illustrated in Fig. 7a, if flow 1 and flow 2's packets are scheduled alternately, then both flows will receive exactly the same amount of service on their dominant resources at all times (see Fig. 7b), *i.e.,*

$$T_1(t_1, t_2) = T_2(t_1, t_2) . \quad (22)$$

It is easy to see that RFB of the given schedule is 0.

This is a pleasant surprise. The example above indicates that under some circumstance, scheduling multiple resources may be "fairer" than scheduling a single one. The key reason here is that, even when packets are scheduled as discrete entities, two flows can receive service on their dominant resources in parallel, which is impossible under the single-resource setting. This demonstrates the significant difference between single-resource and multi-resource scheduling.

Despite such difference, with DRGPS, the insights and techniques derived for single-resource queueing could still be leveraged in the multi-resource scenario. We briefly discuss this in the next subsection.

B. Packet-By-Packet Scheduling Based on DRGPS

A significant benefit of the idealized DRGPS model is that it enables us to leverage the extensive fair queueing literature to design packet-by-packet scheduling algorithms. Below we give high-level discussions on several design approaches. A detailed case study is deferred to Sec. V.

As an analogy to the single-resource fair queueing, there are three potential approaches for packet-based scheduling to approximate DRGPS. First, similarly to [8], [13], [9], [12], we can emulate DRGPS in the background, using the algorithm in Sec. III-D, and serve packets by referencing the algorithm's scheduling results. Just as in the single-resource case, multiple scheduling choices are available. For example, packets can be scheduled based on either the order of starting time (*e.g.,* FQS [13]) or finishing time (*e.g.,* WFQ [8], [9]) in the referencing DRGPS system. A more complicated scheduling algorithm is also possible. For example, similarly to WF^{2}Q [12], an admission control scheme might be applied when multiple packets are available to schedule – those that are not yet served in the referencing DRGPS system are ineligible for scheduling.

Second, algorithms that emulate DRGPS (or GPS) are usually competitive in terms of both fairness and delay, but they might suffer from high computational complexity in the emulation process. A well-known approach to alleviate this difficulty in the fair queueing literature is to estimate the work progress of GPS based on packets that are currently served (*e.g.,* SCFQ, SFQ, *etc.*). Similar approaches can also be adopted in the multi-resource setting. Since the main complexity of emulating DRGPS is contributed by evaluating the virtual time defined in (12), the key challenge is to efficiently estimate it. The insights derived for the single-resource case can be applied. In fact, the scheduling discipline proposed in [6] may be considered to belong to this category of design, although it directly extends SFQ without referencing DRGPS.

Finally, another line of popular scheduling schemes serve flows in a round-robin fashion (*e.g.,* DRR [18] and SRR [19]), such that their received services are roughly equalized. These algorithms could also be extended to the multi-resource setting. Flows are still served round-robin, but the objective is to roughly equalize the service received on their dominant resources.

Though all three approaches above could be potentially applied to designing packet-by-packet fair queueing schemes, due to the space constraint, we only focus on multi-resource WF^{2}Q as a case study in the next section.

V. A Case Study: Dominant Resource WF^{2}Q

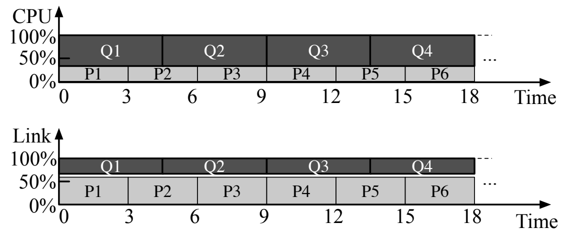
As a case study, we extend WF^{2}Q to the multi-resource setting and analyze its performance, through which we demonstrate the significance of DRGPS by showing that (1) how a packet-based queueing scheme could be designed based on DRGPS and (2) how its performance is measured using the metrics proposed in the previous section. We start by elaborating on Multi-Resource WF^{2}Q.

A. Dominant Resource WF^{2}Q

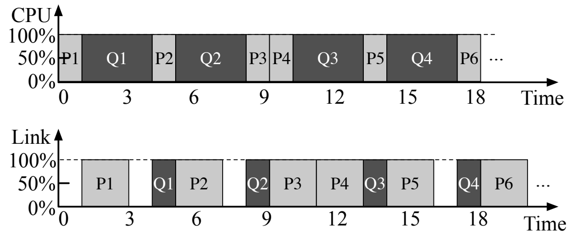
Similar to conventional WF^{2}Q [12], Multi-Resource WF^{2}Q (DRWF^{2}Q) emulates DRGPS in the background and schedules packets based on the order of their finishing times in the referencing DRGPS system. In particular, upon the arrival of packet p_i^k (*i.e.*, the kth packet in flow i), two service tags, the virtual starting time S_i^k and finishing time F_i^k, are stamped, with their values iteratively computed from (14). Whenever there is a scheduling opportunity at time t, packets that *already start service* under DRGPS are *eligible to schedule*, *i.e.,* those with $S_i^k \le v(t)$, where $v(t)$ is the virtual time defined in (12). Among these packets, the one that finishes the earliest (*i.e.,* having the smallest F_i^k) is scheduled.

For example, consider two equally weighted flows that keep sending packets. Flow 1 sends P1, P2, $\dots$, each with the same resource profile $\langle 1 \text{ CPU time}, 2 \text{ Link time}\rangle$, while flow 2 sends Q1, Q2, $\dots$, each with the same profile $\langle 3 \text{ CPU time}, 1 \text{ Link time}\rangle$. Note that the two flows have different dominant resources. Fig. 8a shows the scheduling outcome under DRGPS, based on which, the scheduling results of DRWF^{2}Q are computed and illustrated in Fig. 8b. Consider time 5, at which P2 finishes its service on CPU under DRWF^{2}Q. Both Q2 and P3 are available for service with the same finishing time under DRGPS. However, only Q2 is eligible to be scheduled because at this time, the service for P3 has not yet started under DRGPS.

In this example, DRWF^{2}Q is shown to closely track the progress of DRGPS, just like WF^{2}Q does GPS. Over time, the service received by flow 1 and flow 2 are $\langle 1/3 \text{ CPU time}, 2/3 \text{ Link time}\rangle$ and $\langle 2/3 \text{ CPU time}, 2/9 \text{ Link time}\rangle$, respectively, which are exactly the same as that under DRGPS. A more general analysis is presented in the next subsection.



(a) Scheduling outcome under DRGPS.



(b) Scheduling outcome under DRWF^{2}Q.

Fig. 8. An example of DRWF^{2}Q, where packets are scheduled based on the order of their finishing time in the referencing DRGPS system.

B. Performance Analysis

We now analyze the fairness performance of DRWF^{2}Q, using the extended RFB metric defined in Sec. IV.

Proposition 6: Under DRWF^{2}Q, for any two flows i and j that are backlogged in (t_1, t_2), we have

$$\left| \frac{T_i(t_1,t_2)}{w_i} - \frac{T_j(t_1,t_2)}{w_j} \right| \le 4 \max\left\{ \frac{\tau_i^{\max}}{w_i}, \frac{\tau_j^{\max}}{w_j} \right\}. \quad (23)$$

Proof: Let $(\underline{t}, \bar{t}) \supset (t_1, t_2)$ be the largest time interval during which both flows are backlogged. That is, before time $\underline{t}$ (after time $\bar{t}$), either flow i or j is inactive. For any time $t \in (\underline{t}, \bar{t})$, we claim

$$\left| \frac{T_i(\underline{t},t)}{w_i} - \frac{T_j(\underline{t},t)}{w_j} \right| \le 2 \max\left\{ \frac{\tau_i^{\max}}{w_i}, \frac{\tau_j^{\max}}{w_j} \right\}. \quad (24)$$

Once (24) is proved, we see the statement holds by noting $T_l(t_1,t_2) = T_l(\underline{t},t_2) - T_l(\underline{t},t_1), l = 1,2$.

Suppose at time t, the k_ith packet of flow i and the k_jth packet of flow j are being served, which are denoted as $p_i^{k_i}$ and $p_j^{k_j}$, respectively. Let $B(p)$ and $E(p)$ be the starting and finishing times of packet p in the referencing DRGPS system, respectively. Also, let $T_i^*(a,b)$ be the aggregate service flow i receives on its dominant resource in (a,b) under DRGPS. Finally, let $\bar{t}_i$ and $\bar{t}_j$ satisfy the following relationships:

$$\begin{aligned} T_i(\underline{t},t) &= T_i^*(\underline{t},\bar{t}_i) \ , \\ T_j(\underline{t},t) &= T_j^*(\underline{t},\bar{t}_j) \ . \end{aligned} \quad (25)$$

To show (24), it is equivalent to showing

$$\left| \frac{T_i^*(\underline{t},\bar{t}_i)}{w_i} - \frac{T_j^*(\underline{t},\bar{t}_j)}{w_j} \right| \le 2 \max\left\{ \frac{\tau_i^{\max}}{w_i}, \frac{\tau_j^{\max}}{w_j} \right\}. \quad (26)$$

Without loss of generality, we assume packet $p_i^{k_i}$ finishes earlier than packet $p_j^{k_j}$ under DRGPS, *i.e.,*

$$E(p_i^{k_i}) \le E(p_j^{k_j}) \ . \tag{27}$$

It suffices to consider the following two cases.

Case 1: $B(p_i^{k_i}) \ge B(p_j^{k_j})$. In this case, we have

$$B(p_j^{k_j}) \le \bar{t}_l \le E(p_j^{k_j}), \ l = i, j. \tag{28}$$

As a result,

$$\begin{aligned} \left|\frac{T_i^*(\underline{t}, \bar{t}_i)}{w_i} - \frac{T_j^*(\underline{t}, \bar{t}_j)}{w_j}\right| &= \left|\frac{T_j^*(\underline{t}, \bar{t}_i)}{w_j} - \frac{T_j^*(\underline{t}, \bar{t}_j)}{w_j}\right| \\ &\le \frac{T_j^*(\underline{t}, E(p_j^{k_j})) - T_j^*(\underline{t}, B(p_j^{k_j}))}{w_j} \\ &\le \tau_j^{\max}/w_j, \end{aligned}$$

where the equality holds because of Proposition 5, and the first inequality is derived from (28).

Case 2: $B(p_i^{k_i}) < B(p_j^{k_j})$. We consider two sub-cases.

Sub-Case 1: $E(p_j^{k_j}) \le E(p_i^{k_i+1})$. In this case, we have

$$B(p_i^{k_i}) \le \bar{t}_l \le E(p_i^{k_i+1}), \ l = i, j \ .$$

Hence,

$$\begin{aligned} \left|\frac{T_i^*(\underline{t}, \bar{t}_i)}{w_i} - \frac{T_j^*(\underline{t}, \bar{t}_j)}{w_j}\right| &\le \frac{T_j^*(\underline{t}, E(p_i^{k_i+1})) - T_j^*(\underline{t}, B(p_i^{k_i}))}{w_j} \\ &\le 2\tau_j^{\max}/w_j. \end{aligned}$$

Sub-Case 2: $E(p_j^{k_j}) > E(p_i^{k_i+1})$. In this case, we must have

$$B(p_j^{k_j}) \le E(p_i^{k_i}) \ . \tag{29}$$

Otherwise, we will have

$$B(p_j^{k_j}) > E(p_i^{k_i}) = B(p_i^{k_i+1}), \tag{30}$$

where the last equality holds since flow i is busy. In other words, packet $p_i^{k_i+1}$ starts earlier than packet $p_j^{k_j}$ under DRGPS, which implies that when packet $p_j^{k_j}$ is scheduled under DRWF^{2}Q, packet $p_i^{k_i+1}$ is also eligible for service. This contradicts the principle of DRWF^{2}Q, as packet $p_i^{k_i+1}$ finishes earlier than $p_j^{k_j}$ and should be served before $p_j^{k_j}$.

With (29), we derive as follows:

$$\begin{aligned} \left|\frac{T_i^*(\underline{t}, \bar{t}_i)}{w_i} - \frac{T_j^*(\underline{t}, \bar{t}_j)}{w_j}\right| &\le \left|\frac{T_i^*(\underline{t}, \bar{t}_i)}{w_i} - \frac{T_i^*(\underline{t}, E(p_i^{k_i}))}{w_i}\right| \\ &\quad + \left|\frac{T_i^*(\underline{t}, E(p_i^{k_i}))}{w_i} - \frac{T_j^*(\underline{t}, \bar{t}_j)}{w_j}\right| \\ &\le \frac{T_i^*(\underline{t}, E(p_i^{k_i}))}{w_i} - \frac{T_i^*(\underline{t}, B(p_i^{k_i}))}{w_i} \\ &\quad + \frac{T_j^*(\underline{t}, E(p_j^{k_j}))}{w_j} - \frac{T_j^*(\underline{t}, B(p_j^{k_j}))}{w_j} \\ &\le \tau_i^{\max}/w_j + \tau_j^{\max}/w_j, \end{aligned}$$

where the second inequality follows from (29). ■

Proposition 6 directly leads to the following corollary.

Corollary 1 (RFB): The RFB of DRWF^{2}Q is

$$R = 4 \max_i \left\{ \frac{\tau_i^{\max}}{w_i} \right\} \ . \tag{31}$$

It is worth mentioning that the analysis above does not make any assumptions on the resource requirement patterns of a flow. In particular, *a flow may change its dominant resource at any time and on any packet*. Note that this is not the case in the analysis of [6] for its SFQ scheme, which holds only when each flow has a fixed dominant resource throughout the backlog period.

Though the case study above only focuses on WF^{2}Q, by emulating DRGPS, queueing schemes such as WFQ [8], [9] and FQS [13] will have immediate multi-resource extensions. Similar analysis can also be applied to these algorithms.

VI. Discussion and Future Work

In this section, we discuss some practical concerns that are important for real-world multi-resource fair queueing. We also share our views on some possible future directions.

First, to accurately approximate DRGPS, the system designer needs to know the processing time required by a packet on each resource. This information can be obtained either by packet profiling, before the packet is processed, or by monitoring resource usage during packet processing. However, both are expensive to implement for high-speed networks. The former requires deep packet inspection, while the latter needs to maintain a resource monitor. Note that none of these approaches is needed for conventional fair queueing, for which the only information required is the packet size and is available in the packet header.

A possible solution is to adopt some simple estimation of the processing time based on the packet size only. Since the transmission time can be accurately inferred from the packet size, such estimation is needed only for the other resources. For example, CPU and memory bandwidth are the two most commonly considered resources in middleboxes. The experiment in [6] reveals that the processing times associated with these two resources may be approximated by linear functions of the packet size, suggesting that a simple estimation model could be sufficient in practice.

Second, with increasingly common deployment of software-defined middleboxes on commodity servers, the work complexity of a queueing scheme becomes a more important concern. Algorithms that require emulating DRGPS might not be appropriate choices due to their high complexity. In such cases, simpler scheduling algorithms with constant work complexity, such as DRR [18], may find new application scenarios in the multi-resource setting. The design and evaluation of these algorithms against DRGPS could be fertile ground for future research.

Third, besides fairness, packet delay is also an important concern for a queueing scheme. Traditional fair queueing literature has suggested a variety of techniques to bound the end-to-end delay, in both theory and practice. However, it is unclear if these approaches can be leveraged in the multi-resource middlebox scenario.

Finally, in some scenarios, fairness may not be the primary concern. Instead, a queueing scheme with high resource utilization might be preferred. More generally, there might be some fairness-efficiency trade-off desired by a network operator. Despite recent theoretical works [20], it remains unknown how such trade-off can be implemented via a fair queueing scheme. New theoretical model as well as system designs are therefore required.

VII. Related Work

Service isolation and work conservation are two essential properties desired by fair queueing schemes [11]. GPS was proposed based on the fluid model and serves as an idealized benchmark for which all practical queueing schemes should approximate. GPS plays a central role in the fair queueing literature. Many well-known packet-by-packet scheduling schemes, such as WFQ [8], [9], WF^{2}Q [12], and FQS [13], schedule packets based on emulating GPS. Other popular queueing alternatives, such as SCFQ [16], SFQ [17], and DRR [18], are also designed to approximate the work progress of GPS.

Despite the extensive studies on scheduling link bandwidth, multi-resource fair queueing remains at a nascent stage. Ghodsi *et al.* [6] are the first to investigate this problem. Their design, referred to as DRFQ, borrows the intuition of DRF allocation, and schedules packets in a way such that flows receive roughly the same processing time on their most congested resources. Despite this seminal work, it remains unclear what queueing scheme is fair and how multi-resource fair queueing should be generally designed. The main obstacle is the lack of a GPS-like queueing benchmark, which motivates our work.

As for the notion of fairness in multi-resource allocation, Ghodsi *et al.* [10] proposed the DRF notion to equalize the dominant share of all users and showed a number of desired fairness properties possessed by the resultant allocation. As a significant step forward in multi-resource allocation, DRF has quickly attracted substantial attention and has been generalized to several new dimensions. Some notable works include Joe-Wong *et al.* [20], where the DRF measure is incorporated to a unifying framework to capture fairness-efficiency trade-offs, and Parkes *et al.* [14], where DRF is extended to include the case of having zero demand on certain resources, weighted user endowments, and indivisible tasks. These works are orthogonal and complementary to our investigation into multi-resource fair queueing.

VIII. Concluding Remarks

Middleboxes apply complex network functions in packet processing. Depending on the functional modules the packets must go through, different traffic flows may require vastly different amounts of various resources, including CPU cycles, memory bandwidth, and link bandwidth. In this paper, we generalize the conventional GPS fair scheduling algorithm to the multi-resource setting. The resultant DRGPS offers perfect service isolation that is immune to any strategic behaviours and is work conserving as well. It hence serves as an idealized fair queueing benchmark for which packet-by-packet scheduling schemes should approximate. More significantly, with DRGPS, many techniques and insights that have been developed for conventional fair queueing can also be adapted to design multi-resource packet-based scheduling disciplines. We have demonstrated how this can be achieved under the proposed DRGPS framework. As a case study, we have designed the DRWF^{2}Q scheme based on DRGPS and analyzed its fairness performance.

References

[1] V. Sekar, N. Egi, S. Ratnasamy, M. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. NSDI*. USENIX, 2012.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proc. SIGCOMM*. ACM, 2012.

[3] J. Anderson, R. Braud, R. Kapoor, G. Porter, and A. Vahdat, "xOMB: Extensible open middleboxes with commodity servers," in *Proc. ANCS*. ACM/IEEE, 2012.

[4] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, "Flow processing and the rise of commodity network hardware," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 20–26, 2009.

[5] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, "Towards high performance virtual routers on commodity hardware," in *Proc. CoNEXT*. ACM, 2008.

[6] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica, "Multi-resource fair queueing for packet processing," in *Proc. SIGCOMM*. ACM, 2012.

[7] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE*, vol. 83, no. 10, pp. 1374–1396, 1995.

[8] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. SIGCOMM*. ACM, 1989.

[9] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.

[10] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. NSDI*. USENIX, 2011.

[11] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, The Internet and Telephone Network*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[12] J. Bennett and H. Zhang, "WF^{2}Q: Worst-case fair weighted fair queueing," in *Proc. INFOCOM*. IEEE, 1996.

[13] A. Greenberg and N. Madras, "How fair is fair queuing," *J. ACM*, vol. 39, no. 3, pp. 568–598, 1992.

[14] D. Parkes, A. Procaccia, and N. Shah, "Beyond dominant resource fairness: Extensions, limitations, and indivisibilities," in *Proc. EC*. ACM, 2012.

[15] A. Gutman and N. Nisan, "Fair allocation without trade," in *Proc. AAMAS*. IFAAMAS, 2012.

[16] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. INFOCOM*. IEEE, 1994.

[17] P. Goyal, H. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 690–704, 1997.

[18] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375–385, 1996.

[19] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 4, pp. 365–386, 1995.

[20] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework," in *Proc. INFOCOM*. IEEE, 2012.