# Multi-Server Generalized Processor Sharing

Kok-Kiong Yap   Nick McKeown   Sachin Katti
Stanford University
Stanford, California 94305
Email: {yapkke,nickm,skatti}@stanford.edu

***Abstract*—End-hosts are increasingly equipped with multiple network interfaces, ranging from smartphones with multiple radios to servers with multi-homing. These interfaces are diverse; some are expensive to use (e.g. 4G), some are free (e.g WiFi) and they have different rates and reliability. On the other hand, end-hosts now run diverse applications with different priorities, from relatively less important web browsing to higher priority VoIP and video calls. Finally, users may have policies that constrain interface use (e.g. use 4G only for high priority flows). This paper tackles the question of how different applications can use different subsets of the available network interfaces, while ensuring a fair resource allocation among flows, while satisfying policy constraints. We generalize prior classical work on processor sharing (GPS) to the case of flows sharing different subsets of the available interfaces. We show a simple scheduling scheme for packet-by-packet GPS over multiple interfaces, and prove that it can provide bounded delay and rate guarantees.**

## I. Introduction

Computers frequently have multiple network interfaces. For example, smartphones commonly have WiFi, 3G, and 4G interfaces, and the number of interfaces is growing over time. If the phone uses several interfaces at the same time it can increase throughput for applications like video streaming and VoIP, it can increase connectivity, and it can reduce delay. Similarly, CDN servers often have multiple network interfaces, or several independent paths to a client, and can benefit from using several at the same time.

The research community has recently proposed mechanisms for a computer to stitch multiple interfaces together into a single high throughput logical connection [1], [2]. While this approach is a good start, we believe applications need more flexibility than simply using the logical combination all of the currently available interfaces. For example, consider a smartphone user using a video chat application while browsing the web. The voice channel might prefer to use network interfaces giving low-latency, continuous connectivity, while the video might prefer interfaces giving low-cost sustained high throughput, and the web client might prefer interfaces giving short bursts of high throughput. This suggests a model where different applications use different subsets of the available interfaces. In addition, the user might express policy preferences on network usage, for example to only use an expensive 4G connection for VoIP, while using free WiFi connectivity for large but less important data transfers. Similar preferences exist at the server, since different pieces of content might have different priorities, and the server might try to avoid using a high cost connection except for important traffic. So a natural question is: *How can applications share the set of available network interfaces, meet the individual needs of each application, and meet the policy constraints on resource usage?*

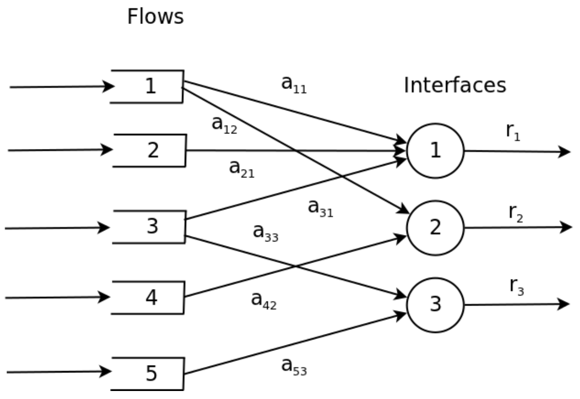

Fig. 1. Illustration of multiple flows served by multiple interfaces, where each interface operates at a different rate.

Current techniques such as multipath TCP give *equal* preference to every flow and sub-flow, and do not accommodate different application preferences or user policies. The focus in these works is on bandwidth aggregation and congestion control in the network, and not on how the applications within the host should share different sets of network interfaces. In this paper we tackle the question of how multiple application flows can fairly and efficiently share multiple network interfaces, provide rate and delay guarantees to each flow, and impose user policy constraints. Here we define fairness to be the well-known max-min fair allocation: *No flow can get better service without hurting the service of another flow with a lower level of service than itself.*

A natural starting point is Generalized Processor Sharing (GPS) [3], which shows how multiple competing flows can receive their max-min fair share of a bottleneck link, and how individual flows can have rate/delay guarantees. But all prior GPS work has been for a *single* shared interface, or settings where flows share all of the interfaces without constraint. The key contribution of this paper is to extend results for packetized GPS (*i.e.* max-min fairness and rate/delay guarantees) to a system with *multiple interfaces*, where each flow is served by a subset of the interfaces.

Figure 1 shows the model of our system: Each application flow has its own transmit queue (*e.g.* socket buffer) where packets wait until an interface is available. Each flow is assumed to be constrained to use *only a subset of the interfaces*, as shown by the edges in Figure 1 connecting the transmit queues to the interfaces. We call this the *routing constraint* because it limits how packets can be routed from the flow queues to the interfaces, and we represent it by the routing matrix $\Pi = [\pi_{ij}]$ connecting $m$ flows to $n$ interfaces, where $\pi_{ij} = 1$ if and only if flow $i$ is willing to be served by interface $j$. GPS for multiple interfaces is characterized by the weights of each flow $\phi_1, \phi_2, \cdots, \phi_m$, the service rate of each interface $r_1, r_2, \cdots, r_n$, and the routing matrix $\Pi$.

We define GPS to be a service discipline for which *currently backlogged flows are served bit-by-bit at their weighted max-min fair rate*, and *packetized* GPS (PGPS) extends GPS to the case where whole packets are served at a time. Our definition, by extending the original result to multiple interfaces, can be considered a generalized definition of GPS and includes the prior definition as a special case [3].

With our eventual goal of proving results for *packetized* GPS in mind, we follow the usual approach of first proving results for the idealized and simpler *bit-by-bit* case. In section II-A we use lexicographical maximization [4] to find how multiple interfaces can give weighted max-min service to backlogged flows when served bit-by-bit. Then in section II-B we show that this leads to a guaranteed minimum bit-by-bit service rate for each flow, and in section II-C we extend the results in [3] to show how the system can give delay guarantees. In section III we extend our bit-by-bit results to the *packetized* case providing both rate and delay bounds.

## II. Bit-by-bit GPS for Multiple Servers

We can determine if the system in Figure 1 can emulate generalized processor sharing (GPS) by asking the equivalent question: If we allow flows to be divided among servers[1] arbitrarily, is it possible to allocate a rate on each interface to each flow, such that each flow receives its weighted max-min fair allocation? If it is possible, then the flows could be served bit-by-bit at this rate, and would emulate GPS.

### A. Weighted max-min fair allocation

Our goal is to find rate $a_{ij}$ at which interface $j$ should served flow $i$, such that the aggregate service rate for each flow $a_i = \sum_j a_{ij}$ is the weighted max-min fair allocation. Using a general approach first described by Megiddo [4], we find the max-min fair allocation over several iterations. As we describe in more detail below, in each iteration we solve several maximum flow problems for the system shown in Figure 2 where we have added a source and sink node to our model.[2] The capacity of the edges from a flow to a server are infinite, and the capacity of the edge between server $j$ and sink is the service rate of server $j$, $r_j$.

In the first iteration we maximize the minimum flow rate. In other words, we find the minimum rate that any flow will

[1]We will use the terms *server* and *interface* interchangeably to tie our model (servers and queues) to the system that motivated our work (multiple network interfaces).

[2]This is similar to solving a maximum network flow to find a maximum match in a bipartite graph [5].

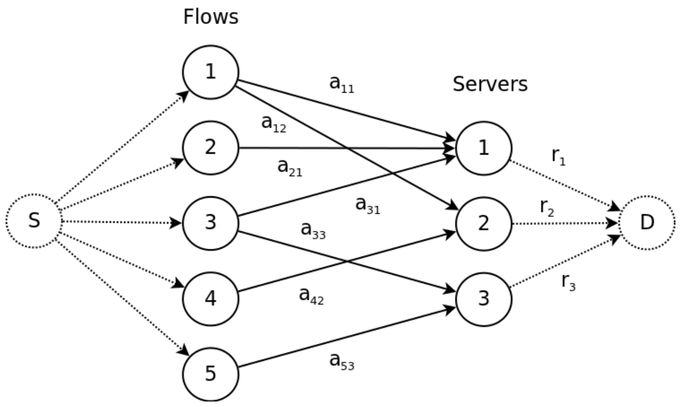

Fig. 2. System in Fig. 1 modeled as a bipartite graph. A source and sink is added as part of the formulation for the maximum flow problem.

receive, $a_{min} = \min_i a_i$. To do this, we set the capacity of the edge from the source to each flow $i$ to be $\phi_i x$, then find the largest total flow (i.e. the largest value of $x$) that does not exceed the rate of the servers. There are a number of ways to maximize $x$, and we choose to find it by a binary search; in each step of the search we find the maximum network flow, and then increase or decrease $x$ accordingly.[3]

At the end of the first iteration we have maximized the minimum rate $a_{min} = \min_i \phi_i x$. We now fix the rate of this flow for the remainder of the algorithm. In the second iteration, the whole process repeats for the remaining flows (keeping the first flow fixed), to find the second lowest flow rate. In each of the subsequent iterations, we find the rate of one flow per iteration, leading to a weighted max-min fair allocation.

### B. Rate Guarantee

For GPS with a single server of rate $r$, we know that the rate flow $i$ receives, $a_i(t) \geq \phi_i / \sum_i \phi_i * r$. By picking $\phi_i$ appropriately, we can guarantee a minimum rate at which each flow will be served. For example, if we want flow 1 to receive at least 10% of the link rate, then we simply set $\phi_1 = 0.1$ and make sure $\sum_i \phi_i \leq 1$.

Now that we have the weighted max-min fair allocation for the *multiple server case*, it is worth asking if we can still give a rate guarantee for each flow in the system. Theorem 1 tells us that a flow will indeed receive an equivalent rate in a multi-server GPS system.

**Theorem 1.** *Under the weighted max-min fair allocation, the rate flow $i$ receives is at least its weighted fair share among all the flows willing to share one or more interfaces with $i$,*

$$a_i(t) \geq \frac{\phi_i}{\sum_{j,\exists k,\pi_{ik}=1,\pi_{jk}=1} \phi_j} \sum_{j,\pi_{ij}=1} r_j(t). \qquad (1)$$

*Proof:* Imagine all the flows willing to share one or more interfaces with $i$ use exactly the same set of servers. Then the

[3]In theory, the binary search might not terminate, but in practice, we can include a small tolerance to allow it to terminate, or use a more complicated algorithm such as [6].

equation above is an equality, because $a_i$ is the weighted max-min fair allocation. If any flow uses less than this weighted fair share (because the flow has no more packets to send, or because the flow uses a server $i$ is unwilling to use, or because the flow is unwilling to use a server $i$ uses), then it would increase service rate allocation to the remaining flows, including flow $i$. ■

It follows that under the weighted max-min allocation flow $i$ will receive is also at least its weighted fair share of the interfaces it is willing to use,

$$a_i(t) \geq \frac{\phi_i}{\sum_j \phi_j} \sum_{j,\pi_{ij}=1} r_j(t).$$

because this is smaller than the right hand side of Equation 1. For simplicity of notation, we denote the guaranteed rate for flow $i$ as $g_i$ where $a_i(t) \geq g_i,\ \forall t$.[4]

For example, if in Figure 1 we want flow 1 to receive at least 20% of $r_1 + r_2$, then it is sufficient to set $\phi_1 = 0.2$, and $\phi_1 + \phi_2 + \phi_3 + \phi_4 \leq 1$ because only flows 1 to 4 share interfaces with flow 1. When we run the algorithm to set the weighted max-min fair allocation, flow 1 will receive at least the requested service rate.

### C. Leaky Bucket and Delay Guarantee

Another well-known property of single-server GPS is that it allows us to bound the delay of a packet through the system, if the arrival process is constrained. The usual approach is to assume that arrivals are *leaky bucket constrained.* If $A_i(t_1, t_2)$ is the number of arriving packets for flow $i$ in time interval $(t_1, t_2]$, then we say $A_i$ conforms to $(\sigma_i, \rho_i)$ (denoted $A_i \sim (\sigma_i, \rho_i)$) if

$$A(t_1, t_2) \leq \sigma_i + \rho_i(t_2 - t_1) \quad ,\forall t_2 \geq t_1 \geq 0. \qquad (2)$$

The burstiness of the arrival process is bounded by $\sigma_i$, while its sustainable average rate is bounded by $\rho_i$.

In the classic single-server GPS proof, it can be shown that the delay of a packet (the interval between when its last bit arrive to when its last bit is serviced) in flow $i$ is no more than $\sigma_i/\rho_i$. Admission control is very simple: If $\sum_i \rho_i < r$, and $\sum_i \sigma_i \leq B$, where $B$ is the size of the packet buffer, then flow $i$ can be admitted into the system and the delay guarantee can be met.

We will now prove that a multi-server GPS system has the same property, and the delay of a packet in flow $i$ is no more than $\sigma_i/\rho_i$ (Theorem 2). However, the process of deciding whether a new flow can be admitted is more complicated than for the single server case. We have to know which interfaces the flow is willing to use, and whether the requested service rate $\rho_i$ can be met. This means the system has to pick values for $\phi_j, \forall j$ such that the rate guaranteed by Equation 1, $a_i(t) > \rho_i$. If this condition can be met, then the delay guarantee is

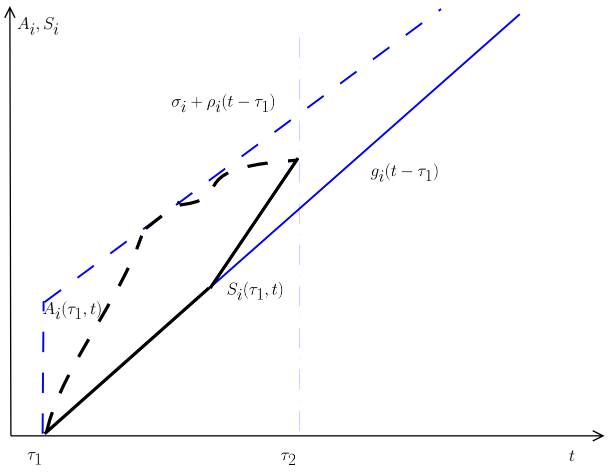

Fig. 3. Illustration of $A_i(\tau_1, t)$ and $S_i(\tau_1, t)$ and their respective upper and lower bounds. Observe that the horizontal distance between $A_i$ and $S_i$ characterizes the delay, while the vertical distance characterizes the backlog at time $t$.

accomplished, and the departure process will also be $(\sigma_i, \rho_i)$-constrained.

**Theorem 2.** *Imagine we wish to admit $(\sigma_i, \rho_i)$-constrained flow $i$ into a multi-server GPS system, and the flow is willing to use interfaces $\pi_{ij}, \forall j$. If $\sum_j \sigma_j \leq B$, and if we can find values of $\phi_j, \forall j$ such that $a_i(t) > \rho_i$, then the delay of any packet in flow $i$ is upper bounded by $\sigma_i/\rho_i$.*

*Proof:* Let $S_i(t_1, t_2)$ be the service received by flow $i$ in time interval $(t_1, t_2]$. Consider flow $i$ that arrives at $\tau_1$ (i.e., becomes backlogged) and finishes at $\tau_2$ (i.e., becomes non-backlogged). We observe that $A_i(\tau_1, t)$ is upper bounded by (2) and $S_i(\tau_1, t)$ is lower bounded by

$$S_i(\tau_1, t) \geq g_i(t - \tau_1) \quad ,\forall \tau_1 \leq t \leq \tau_2$$

where $a_i(t) \geq g_i$ as illustrated in Fig. 3. Since the delay is the length of horizontal line between $A_i$ and $S_i$, we can find its maxima through simple calculus.

We begin by deriving the inverse functions of the bounds,

$$y = \sigma_i + \rho_i(t_a - \tau_1) \quad \rightarrow \quad t_a = \frac{y - \sigma_i}{\rho_i} + \tau_1$$
$$y = g_i(t_s - \tau_1) \quad \rightarrow \quad t_s = \frac{y}{g_i} + \tau_1.$$

The delay of packets must then be upper bounded by

$$D = t_s - t_a = \frac{y}{g_i} - \frac{y - \sigma_i}{\rho_i}.$$

Observe that $D$ is an affine function with gradient

$$\frac{dD}{dy} = \frac{1}{g_i} - \frac{1}{\rho_i},$$

which is strictly negative constant because $\rho_i < g_i$. This means $D$ is monotonically decreasing. In our analysis, we are interested in the domain of $t \geq \tau_1$. Hence $D$ is maximized at $y = \sigma_i$. This in turn implies that packet delay

$$D \leq \frac{\sigma_i}{g_i} < \frac{\sigma_i}{\rho_i}$$

[4]We can bound the service rate $a_i(t)$ more tightly by calculating the weighted max-min fair rate for each flow assuming they are all backlogged. Let the result be $\mathcal{A}^* = [a^*_{ij}]$, and $g_i = \sum_j a^*_{ij}$. We can prove that $a_i(t) \geq g_i \quad ,\forall t$. However, this does not yield a closed-form solution. The proof is fairly simple and is omitted here.

because $\rho_i < g_i$. This maximum delay occurs for the last bit that arrives at time $\tau_1$ for the last bit that arrives in the initial burst.[5] ■

## III. Packet-by-Packet GPS for Multiple Servers

GPS for multiple servers is an idealized service discipline that assumes flows are infinitely divisible and can be serviced bit by bit. In practice however, flows transmit in units of packet which are of different sizes. In this section, we extend GPS to the packetized case, and prove that we can provide similar rate and delay guarantees with multiple servers as in the single server case.

Packet-by-packet GPS (PGPS) approximates GPS by serving packets with the smallest finishing time first. We show how the rate and delay guarantees from GPS can be extended to PGPS. Note that both GPS and PGPS are work-conserving since a server will be idle if and only if there are no other flows in the system that it can serve. Further, PGPS is busy (i.e., there is at least a server servicing a flow) if GPS is busy. The converse is not true since PGPS is constrained by the need to schedule packets. Hence, the accumulated service under GPS $S$ is greater or equal to that under PGPS $\hat{S}$ at all times, i.e., $\sum_i S_i(0,t) \geq \sum_i \hat{S}_i(0,t)$ for all $t$. We first show the delay guarantees that PGPS can provide in the multiple server case.

### A. Delay Guarantee

Single-server PGPS is shown to have additional delay of no more than $L_{max}/r$ as compared to single-server GPS, where $L_{max}$ is the maximum length of a packet and $r$ is the service rate of the server. Our goal is to provide a similar bound on additional delay for multi-server PGPS allowing the delay guarantee for multi-server GPS (in Theorem 2) to be naturally extended to multi-server PGPS. Our approach is to consider the cases where the packetized nature of PGPS imposes more delays than GPS, and to bound these delays. Consider a sequence of packets serviced by server $j$ under PGPS. Let packet $p_k$ be the $k^{\text{th}}$ packet in this sequence. If $p_k$ is serviced by server $j$ under GPS, it can suffer additional delay in PGPS for the following reasons:

1) $p_k$ did not arrive in time to be scheduled for service hence other packets are scheduled for service by server $j$, while $p_k$ has to wait due to the packet constraint compared to the GPS case. We denote this delay as $d_l$.
2) Server $j$ can also fall behind because it has to service a full packet, while under GPS it did not service all the bits in the packet since the packet can be split and serviced over multiple servers. Further, the policy of serving the packet with earliest finishing time first can induce server $j$ to serve a packet that it did not under GPS. This can also result in server $j$ falling behind. We denote this delay as $d_w$.
3) Finally, $p_k$ might be serviced at a faster rate across multiple interfaces under GPS than what an individual server can offer in PGPS. Hence an extra delay can be incurred if $p_k$ is serviced at a lower rate by server $j$ than what it receives under GPS. We denote this delay as $d_r$.

We will now analyze each of these delays :

*1) Delay due to mis-ordering, $d_l$:*

**Lemma 1.**

$$d_l \leq \frac{L_{max}}{r_{min}},$$

*where $L_{max}$ is the maximum size of a packet, and $r_{min} = \min_i r_i$ is the minimum service rate among all servers.*

*Proof: Assume server $j$ only services packets it also serviced under GPS. Let the length of $p_k$ be $L_k$, its arrival time be $a_k$, and its departure time under GPS and PGPS be $u_k$, $t_k$ respectively. Consider packet $p_m$ where $m$ is the largest index for which $0 \leq m < k$ and $u_m > u_k$.*

*For $m > 0$, $p_m$ begins transmission at $t_m - (L_m/r_j)$ and packets indexed $m+1$ to $k$ must arrive after this time, i.e,*

$$a_i > t_m - \frac{L_m}{r_j}, \quad \forall m < i \leq k.$$

*Since the packets indexed from $m+1$ to $k-1$ arrive after $t_m - (L_m/r_j)$ and departs before $p_k$ under GPS,*

$$\begin{aligned} u_k &\geq \frac{\sum_{i=m+1}^{k} L_i}{r_j} + t_m - \frac{L_m}{r_j} \\ \therefore u_k &\geq t_k - \frac{L_m}{r_j} \end{aligned}$$

*Therefore,*

$$d_l = t_k - u_k \leq \frac{L_{max}}{r_j} \leq \frac{L_{max}}{r_{min}}.$$

■

*2) Delay due to mis-serviced packets, $d_w$:* Consider the following example with servers $k, l$ having rate of $r_k, r_l$ respectively where $r_k \ll r_l$. If flows $i, j \in \mathcal{F}_k$ and $j \in \mathcal{F}_l$, we can show that under GPS service server $k$ would only serve flow $i$ and server $l$ would only serve flow $j$.

However under PGPS, server $k$ would service flow $i$ because of the following: The next packet in the queue for server $l$ has finishing time $t_{l+1} = t_l + (L_{l+1}/r_l)$ where $t_l$ is the finishing time of the current packet being serviced and $L_x$ is the length of the $x^{\text{th}}$ packet. Similarly, the next packet in the queue for server $k$ has finishing time $t_{k+1} = t_k + (L_{k+1}/r_k)$. Since $r_k \ll r_l$, $t_{l+1}$ can be smaller than $t_{k+1}$ and $i \in \mathcal{F}_k$, resulting in server $k$ choosing to service flow $i$. This results in an additional delay $d_w$ being added to the packets of flow $j$ being serviced by server $k$.

**Lemma 2.**

$$d_w < \lceil \log_2 n \rceil \frac{L_{max} r_{max}}{r_{min}^2},$$

*where $n$ is the number of servers, $L_{max}$ is the maximum size of a packet, and $r_{min} = \min_i r_i, r_{max} = \max_i r_i$ are*

[5] We can further show that $\sigma_i / \sum_i a^*_{ij}$ is a tight upper bound of packet delay. This delay occurs in the worst case scenario where $\sigma_j \gg \sigma_i$ and $\rho_j = g_j$ for all $j \neq i$ and flow $i$ experiences the worst possible arrival process $A_i(\tau_1, t) = \sigma_i + \rho_i(\tau_1, t)$.

*the minimum and maximum service rate among all servers respectively.*

*Proof: Consider the above example. Server $k$ will continue to choose to service flow $i$ until the next packet queued for service in server $l$ has finishing time greater than $t_{k+1}$. While the finishing time of packets for server $l$ increases by a maximum of $t_{k+1} < L_{max}/r_k$, they are sent at a lower rate on server $k$ delaying server $k$ up to $L_{max}r_l/r_k^2$.*

*So server $l$ can delay server $k$ by at most $L_{max}r_l/r_k^2$. However, we have $n$ servers in system. Can the other systems affect server $k$ similarly?*

*Once server $k$ has been delayed, it can only be further delayed by another server which is similarly delayed. Therefore, this can only happen to server $k$ for $\lceil \log_2 n \rceil$ times where $n$ is the number of servers in the system.*

$$d_w < \lceil \log_2 n \rceil \frac{L_{max} r_{max}}{r_{min}^2}.$$

■

Under multi-server GPS, a packet in flow $j$ can also be split and serviced over multiple servers. Under PGPS, it is necessary that server $k$ service the entire packet meaning it has to service the bits in that packet that it does not serve under GPS. We can think of the above as PGPS servicing a partial packet it did not serve under GPS, which can be considered a special of the above lemma.

*3) Delay due to different service rate, $d_r$:* Under GPS, $p_k$ of flow $i$ would be serviced at its weighted max-min fair rate $a_i(t)$. However under PGPS, a packet must be serviced as an entity, i.e., by a single server at the service rate of that server, which is unlikely to be exactly $a_i(t)$. This can incur an extra delay $d_r$ which is upper bounded in the following:

**Lemma 3.**

$$d_r < \frac{L_{max}}{r_{min}},$$

*where $L_{max}$ is the maximum length of a packet, and $r_{min}$ is the minimum service rate for a server.*

*Proof: Consider $p_k$ in flow $i$ of length $L_k$ serviced at $a_i(t)$ under GPS and $r_j$ under PGPS. The difference in transmission time is*

$$\begin{aligned} d &= \frac{L_k}{r_j} - \frac{L_k}{a_i(t)} \\ &\leq L_k \left( \frac{1}{r_{min}} - \frac{1}{\max_t a_i(t)} \right) \\ \therefore d_r &< \frac{L_{max}}{r_{min}}, \end{aligned}$$

*as required by our lemma.* ■

This result is intuitive: A packet cannot be delayed more than the time it takes for it to be serviced under PGPS.

Equipped with the above lemmas, we can now upper bound the delay of a packet under PGPS for multiple interfaces.

**Theorem 3.**

$$F_P - F_G < L_{max} \left( \frac{\lceil \log_2 n \rceil r_{max}}{r_{min}^2} + \frac{2}{r_{min}} \right)$$

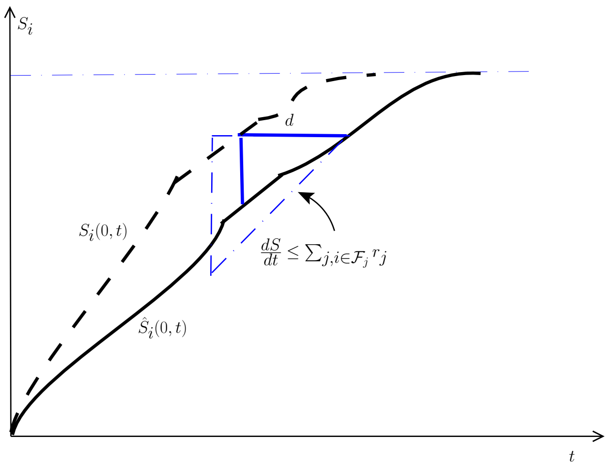

Fig. 4. Illustration of cumulative service under GPS and PGPS, with the relation of the service bound with respect to the delay bound.

*where $F_P, F_G$ are the finishing times of packet under PGPS and GPS respectively.*

*Proof: Consider that*

$$\begin{aligned} & F_P - F_G \\ & \quad \leq d_l + d_w + d_r \\ & \quad < \frac{L_{max}}{r_{min}} + \lceil \log_2 n \rceil \frac{L_{max} r_{max}}{r_{min}^2} + \frac{L_{max}}{r_{min}} \\ & \quad < L_{max} \left( \frac{\lceil \log_2 n \rceil r_{max}}{r_{min}^2} + \frac{2}{r_{min}} \right), \end{aligned}$$

*as we can see from Lemmas 1, 2 and 3.* ■

Similar to single-server PGPS, the additional delay incurred by multi-server PGPS is a function of $L_{max}$, the service rate of the servers $r$ and the number of servers $n$. These quantities are readily available in the process of flow admission, allowing the delay bound (in Theorem 2) to be easily extended for multi-server PGPS.

## *B. Service Bound*

The difference in cumulative service under single-server GPS and single-server PGPS is bounded by the length of the largest packet $L_{max}$. We can provide a similar bound based on the delay bounds we have just derived in Theorem 3. This enables us to upper bound the cumulative service a flow $i$ receives under multi-server PGPS compared to multi-server GPS.

**Theorem 4.** *The difference in cumulative service between PGPS and GPS is*

$$\begin{aligned} & S_i(0,t) - \hat{S}_i(0,t) \\ & \quad < L_{max} \left( \frac{\lceil \log_2 n \rceil r_{max}}{r_{min}^2} + \frac{2}{r_{min}} \right) \sum_{j,\pi_{ij}=1} r_j. \end{aligned}$$

*Proof: Consider the cumulative service of flow $i$ under GPS and PGPS, denoted as $S_i(0,t)$ and $\hat{S}_i(0,t)$ respectively. At any point in time, the delay is bounded by Theorem 3 (as* *illustrated by Fig. 4). Since the service rate of flow $i$ is upper bounded by $\sum_{j,\pi_{ij}=1} r_j$, we can deduce that*

$$\begin{aligned} & S_i(0,t) - \hat{S}_i(0,t) \\ &< \frac{dS}{dt}(F_P - F_G) \\ &< L_{max}\left(\frac{\lceil \log_2 n \rceil r_{max}}{r_{min}^2} + \frac{2}{r_{min}}\right) \sum_{j,\pi_{ij}=1} r_j. \end{aligned}$$

■

Theorem 4 in turns bound the maximum backlog possible under multi-server PGPS. This allows us to check if we have sufficient packet buffers to accommodate the additional backlog. Again, the service difference is a function of $L_{max}$, the service rate of the servers $r$ and the number of servers $n$ which is readily available during flow admission.

Let us consider a system of 12 servers with $L_{max} = 1500$ bytes, $r_{max} = 1$ Gbps and $r_{min} = 100$ Mbps. The extra delay incurred by multi-server PGPS is 5.04 $ms$ as compared to 0.012–0.12 $ms$ in the single server case. This also implies a maximum service difference of 3.125 MB for a flow with aggregate bandwidth of 5 Gbps which is significantly more than the 1.5 KB incurred by single-server PGPS.

## IV. Related Work

In [3], Parekh and Gallager defines GPS for the case of a single node, and proposed packet-by-packet version of the idealized algorithm. GPS provides a rigorous theoretical description of the bit-by-bit round robin server proposed independently by Gallager and Katevenis [7]. PGPS for a single server—also known as weighted fair queuing—was also independently proposed by Demers et. al. in [8]. GPS (and weighted fair queuing) has since been widely used as the canonical service discipline in many other work published under the banner of fair queuing.

There are several extensions of GPS in literature. Parekh and Gallager extends the analysis of GPS to the case of multiple nodes, i.e., GPS servers working in series [9]. In [10], Blanquer and Ozden extends GPS to the case where multiple servers of equal service capacity works in parallel to provide service to flows without any routing constraints between the flows and servers.

There are also other work that look at generalizing the notion max-min fairness to different contexts. In [11], Bejerano et. al. investigate the problem of fair allocation where the load is non-conserving, i.e., the same task can impose different load on different servers. In [12], Ghodsi et. al. explore what it means to be fairly allocated heterogeneous resources, i.e., resources of different types.

Our work differs from the above work in that we generalizes GPS for the case where flows *with arbitrary routing constraints* are serviced by servers of *varying service capacities*.

## V. Discussion

This paper shows how the simple idea of servicing flows in order of their earliest finishing times can provide strong service and delay guarantees with PGPS even when multiple servers are being used with routing constraints. While the current simple scheme bounds the relative difference in delay and service guarantees between GPS and PGPS, we believe there is some room for improving PGPS to more closely match GPS. Specifically as appendix B shows, in GPS flows and servers are clustered into constant “rate” clusters (CRCs) where flows in the same CRC are served at the same weighted rate (as in Theorem 5). Lemma 5 further shows that constraining a server to only serve flows in the same CRC does not hurt the rate and delay guarantee, and in fact might help improve them since there is less contention on that service. Hence, a better approximation of GPS for multiple servers might be to constrain each server to serve only flows in their own CRC.

The above heuristic might reduce the average amount of mis-service in PGPS, but it does not eliminate it. Specifically, as flows and servers change membership within a CRC when a flow becomes backlogged or non-backlogged, packets currently being serviced are “stuck” with the current server. Since PGPS cannot simultaneously service a flow using multiple servers unlike GPS, whichever server is chosen to service such a “stuck packet” under PGPS will be servicing this packet (or some part of this packet) in place of the other servers that GPS would have chosen. Hence, while the bounds improve, we believe there will be some gap between GPS and PGPS because of the atomicity of a packet under PGPS.

Nonetheless, the current simple PGPS scheduling technique provides a practical basis for implementing such resource sharing mechanisms in multiple server systems such as our multi-radio smartphones, or multi-homed servers with flow priorities and routing constraints. Our current work includes designing a low complexity scheduling algorithm that realizes PGPS, as well as extending to the case where the interfaces have time-varying rates.

## Acknowledgment

The authors would like to thank Tim Roughgarden, Guru Parulkar, Te-Yuan Huang, Masayoshi Kobayashi and Yiannis Yiakoumis for very helpful discussions that shaped the paper.

## References

[1] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, “Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet,” *IEEE/ACM Trans. Netw.*, vol. 14, pp. 1260–1271, December 2006. [Online]. Available: http://dx.doi.org/10.1109/TNET.2006.886738

[2] E. Nordstrom, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, “Serval: An end-host stack for service-centric networking,” in *Proc. Networked Systems Design and Implementation*, April 2012.

[3] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: the single-node case,” *IEEE/ACM Trans. Netw.*, vol. 1, pp. 344–357, June 1993. [Online]. Available: http://dx.doi.org/10.1109/90.234856

[4] N. Megiddo, “Optimal flows in networks with multiple sources and sinks,” *Mathematical Programming*, 1974.

[5] R. E. Tarjan, *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, 1991, ch. 9.

[6] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, "A fast parametric maximum flow algorithm and applications," *SIAM J. Comput.*, vol. 18, pp. 30–55, February 1989. [Online]. Available: http://dx.doi.org/10.1137/0218003

[7] E. L. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 9, pp. 1024–1039, 1991.

[8] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 19, pp. 1–12, August 1989. [Online]. Available: http://doi.acm.org/10.1145/75247.75248

[9] A. K. Parekh and R. G. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Trans. Netw.*, vol. 2, pp. 137–150, April 1994. [Online]. Available: http://dx.doi.org/10.1109/90.298432

[10] J. M. Blanquer and B. Özden, "Fair queuing for aggregated multiple links," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 189–197, August 2001. [Online]. Available: http://doi.acm.org/10.1145/964723.383074

[11] Y. Bejerano, S.-J. Han, and L. E. Li, "Fairness and load balancing in wireless lans using association control," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, ser. MobiCom '04. New York, NY, USA: ACM, 2004, pp. 315–329. [Online]. Available: http://doi.acm.org/10.1145/1023720.1023751

[12] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 24–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972490

## APPENDIX

### A. *Consistency with GPS for Single Server*

Generalized Processor Sharing (GPS) is defined for a single server in [3]. For a single server, it is characterized by weights of each flow $\phi_1, \phi_2, \cdots, \phi_m$ and rate of the server $r$, where $m$ is the number of flows. When $\phi_i = \phi$ for all $i$, GPS reduces to uniform processor sharing which provide max-min fair service to the flows. Let $S_i(t_1, t_2)$ be amount of service (in bits) flow $i$ receives in time interval $(t_1, t_2]$. We say the flow is backlogged at time $t$ if it has packets queued for service at time $t$. A GPS server is defined as one for which

$$\frac{S_i(t_1,t_2)}{S_j(t_1,t_2)} \geq \frac{\phi_i}{\phi_j} \quad ,\forall j$$

for flow $i$ that is continuously backlogged in $(t_1, t_2]$.

Our definition of multi-server GPS can be considered a generalization of the single-server GPS. In multi-server GPS, we service the set of currently backlogged flows $\mathcal{B}$ at their weighted max-min fair rate $a(t)$. The service received by flow $i$ in time interval $(t_1, t_2]$ is then given by

$$S_i(t_1,t_2) = \int_{t_1}^{t_2} a_i(t)dt.$$

For the case of a single server, the only reasonable routing matrix is for all flows to be using the only server available. For any subset of the flows backlogged, the weighted max-min fair allocation is to service flow $i \in \mathcal{B}(t)$ at rate $a_i(t) = \phi_i / \sum_{j \in \mathcal{B}(t)} \phi_j * r$. Consider flow $i$ that is continuously backlogged in $(t_1, t_2]$. The service received by flow $i$

$$S_i(t_1,t_2) = \int_{t_1}^{t_2} \frac{\phi_i}{\sum_{j\in\mathcal{B}(t)} \phi_j} r dt. \tag{3}$$

Now consider flow $j$ with weight $\phi_j$. If flow $j$ is backlogged at $t$, $a_j(t) = \phi_j / \sum_{i \in \mathcal{B}(t)} \phi_i * r$. Else $a_j(t) = 0$. Hence,

$$S_j(t_1,t_2) = \int_{t_1}^{t_2} a_j(t)dt \leq \int_{t_1}^{t_2} \frac{\phi_j}{\sum_{i\in\mathcal{B}(t)} \phi_i} r dt. \tag{4}$$

Combining (3) and (4),

$$\frac{S_i(t_1,t_2)}{S_j(t_1,t_2)} \geq \frac{\phi_i}{\phi_j},$$

as per the definition of GPS for a single server.

### B. *Disjoint Constant "Rate" Clusters*

In the following, we will demonstrate two properties of multi-server GPS:

1) Flows and servers are clustered into constant rate clusters (CRCs) where flows in the same CRC are served at the same weighted rate.
2) Constraining a server to only serve flows in the same CRC does not hurt the rate and delay guarantee.

Recap that given the set of backlogged flows $\mathcal{B}$, their weights $\phi$ and the service rate of the servers $r$, we can derive the weighted max-min fair rate using lexicographical maximization (described in section II-A). The result is $\mathcal{A} = [a_{ij}]$. From $\mathcal{A}$, we can derive the weighted max-min fair rate allocation for flow $i$ which is $a_i = \sum_j a_{ij}$. We say flow $i$ is in set $\mathcal{U}_j$ if and only if $a_{ij} > 0$.

We can now establish an interesting property of multi-server GPS: Flows that are serviced by a common interface would be serviced at the same weighted rate (as shown in Lemma 4). This result is intuitively pleasing since our goal here is to serve flows fairly.

**Lemma 4.**

$$i,j \in \mathcal{U}_k \quad \implies \quad \frac{a_i(t)}{\phi_i} = \frac{a_j(t)}{\phi_j}.$$

*Proof: If $a_i(t)/\phi_i > a_j(t)/\phi_j$, $a_i$ can be decreased to increase $a_j$—by reducing $a_{ik}$ to increase $a_{jk}$ since $i \in \mathcal{U}_k$ i.e., $a_{ik} > 0$. This contradicts the fact that $a_i, a_j$ are from the weighted max-min fair rate allocation. Therefore, $a_i(t)/\phi_i$ cannot be greater than $a_j(t)/\phi_j$. For the same reason, the converse is true, i.e., $a_j(t)/\phi_j$ cannot be greater than $a_i(t)/\phi_i$. Hence, $a_i(t)/\phi_i = a_j(t)/\phi_j$.* ■

Given the weighted max-min fair allocation, we can construct a bipartite graph between the flows and the servers. In this bipartite graph, an edge exists between flow $i$ and server $j$ if and only if $a_{ij} > 0$. We call each connected subgraph in the resulting graph a *constant "rate" cluster* (CRC). The reason for the name would be clear soon. We can now extend Lemma 4 to all flows within a CRC, showing that flows in the same CRC are served at the same weighted rate.

**Theorem 5.** *Flows in each CRC are serviced at the same weighted rate.*

*Proof: From definition, if flow $i$ is serviced by server $k$, $i, k$ must be in the same CRC.*

*Consider flow $j$ in the same CRC, which means flow $j$ is connected to flow $i$ in some way. There are two possibilities:*

1) *Flow $j$ can be connected to flow $i$ via a common interface $k$. From Lemma 4, this implies flows $i, j$ would be serviced at the same weighted rate, i.e., $a_i/\phi_i = a_j/\phi_j$.*
2) *Alternatively, flow $j$ can be connected to flow $i$ via a series of other flows and interfaces. Let the series of flows be $[x_1, x_2, \cdots, x_l]$. As with above,*

$$\frac{a_j}{\phi_j} = \frac{a_{x_1}}{\phi_{x_1}} = \cdots = \frac{a_{x_l}}{\phi_{x_l}} = \frac{a_i}{\phi_i}.$$

*Therefore, all flows in the same CRC must be serviced at the same weighted rate.* ■

Having established that flows in the same CRC are serviced at the same weighted rate, we can now demonstrate that it cannot hurt to constrain a server to service a flow in the same CRC. In other words, if flow $i$ and server $j$ are in the same CRC, there exists a valid weighted max-min fair allocation in which flow $i$ is indeed serviced by server $j$.

Consider a CRC (denoted as $\mathcal{C}$) with $m'$ flows and $n'$ servers. The rate is achieved for arbitrary flow $i \in \mathcal{C}$ via a subset of the servers in $\mathcal{C}$. The allocation of service rate between the flows and servers in $\mathcal{C}$ is captured by a $m'$ by $n'$ matrix $\mathcal{A}_{\mathcal{C}} = [a_{ij}]$ for all $i, j \in \mathcal{C}$. We know from Theorem 5 that all $m'$ flows are serviced at the same rate $a' = \left(\sum_{j\in\mathcal{C}} r_j\right)/m'$, i.e., $a_{\mathcal{C}} = [a_1, a_2, \cdots, a_{m'}] = [a', a', \cdots, a']$ where $a_i = \sum_j a_{ij}$.

**Lemma 5.** *Consider flow $i$ and server $j$ in the same CRC $\mathcal{C}$. If flow $i$ is willing to be serviced by server $j$, there exists feasible allocation where $i$ is indeed serviced by $j$, i.e.,*

$$i, j \in \mathcal{C}, \quad \pi_{ij} = 1 \implies \exists \mathcal{A}_{\mathcal{C}}, \quad a_{ij} > 0.$$

*Proof: Let's consider an allocation $\mathcal{A}'_{\mathcal{C}}$ where flow $i \in \mathcal{F}_k$ but $i \notin \mathcal{U}_k$. Since $i, j \in \mathcal{C}$, there must be a path from $i$ to $j$ in the subgraph. This means flow $i$ must use a server $k$ in $\mathcal{A}'_{\mathcal{C}}$ which in turns service a flow that is serviced by server $j$ or yet another server that is indirectly connected to $j$.*

*Hence, if we increase $a_{ij}$ by $\epsilon > 0$, we can reduce that amount of service on server $k$ which cascade this transfer of allocation until it reaches $i$. This would be a feasible allocation for which flow $i$ is serviced by server $j$, for which service rate of each flow remains the same.* ■