Multi-Resource Generalized Processor Sharing for Packet Processing

Wei Wang, Ben Liang, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
June 4, 2013
Middleboxes (MBs) are ubiquitous in today’s networks

The sheer number is on par with the L2/L3 infrastructures

Perform a wide range of critical network functionalities

WAN optimization, intrusion detection and prevention, etc.
Performing different network functionalities requires different amounts of MB resources

- Basic Forwarding: Bandwidth intensive
- IP Security Encryption: CPU intensive

![Graph showing resource usage for different middlebox modules]

Ghodsi et al SIGCOMM12
How to let flows \textit{fairly} share \textit{multiple resources} for packet processing?
What do we mean by fairness?

Fair queueing can be defined via a set of highly desired scheduling properties

Predictable service isolation

For each backlogged flow, the received service is \textit{at least} at the level when \textit{every resource is equally} allocated (or in proportion to the flow's weight)
Service isolation cannot be compromised by some strategic behaviours

A flow may cheat by asking for the amount of resources that are not needed

E.g., asking for more bandwidth by adding dummy payload to inflate the packet size

Truthfulness (Strategy-proofness)

No flow can receive better service (i.e., finish faster) by misreporting the amount of resources it requires
What do we mean by fairness? (Cont’d)

Work conservation

No resource that could be used to serve a busy flow is wasted in idle
Multi-Resource Fair Queueing

Simple fairness notion leads to unfairness in the multi-resource setting [Ghodsi12]

Per-resource fairness

Bottleneck fairness

A promising insight is suggested in [Ghodsi12]

Dominant Resource Fairness (DRF)

Flows should receive roughly the same service on their most congested resources (DRFQ)
Open Questions

Is there a general guideline to design multi-resource fair queueing?

What’s the benchmark for multi-resource fair queueing?

Any GPS-like fair queueing benchmark?

Can the techniques developed for the single-resource fair queueing be leveraged in the multi-resource setting?
Our Contribution

Dominant Resource GPS (DRGPS)

An *idealized fluid fair queueing benchmark* that achieves all desired scheduling properties

Clarify the design objective for practical queueing algorithms

Techniques developed for single-resource fair queueing algorithms can be leveraged in the multi-resource setting
DRGPS
Resources are scheduled in serial for packet processing

E.g., CPU first, followed by the link bandwidth
Multi-Resource Fluid Flow Model

Assume packets can be served in arbitrarily small increments on every resource.

(a) \(100\% \text{ CPU, } 100\% \text{ Link}\).

50\% link allocation is wasted.

(b) \(100\% \text{ CPU, } 50\% \text{ Link}\).

No resource allocation is wasted.
Non-wasteful allocation

Under non-wasteful allocation, we can view that all resources are consumed simultaneously, at the same rate.

<100% CPU, 50% Link>
For a packet, its **dominant resource** is the one that requires the most packet processing time

E.g., Packet P1 has $\langle\text{CPU time, Transmission Time}\rangle = \langle 1, 0.5 \rangle$

CPU is the dominant resource of P1

The **dominant share** is the fraction of dominant resource allocated to process the packet

E.g., $\langle 70\% \text{ CPU, 60}\% \text{ Link}\rangle$ is allocated to process P1

The dominant share of P1 is 70\%
Dominant Resource Fairness (DRF)

At any given time, every backlogged flow is allocated the same dominant share
Max-min fair on the dominant resource

DRGPS achieves the DRF allocation at all times!
**TABLE I**


<table>
<thead>
<tr>
<th>Packet</th>
<th>Flow</th>
<th>Arrival Time</th>
<th>(\langle\text{CPU, Link}\rangle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Flow 1</td>
<td>0</td>
<td>(\langle 4, 2\rangle)</td>
</tr>
<tr>
<td>Q1</td>
<td>Flow 2</td>
<td>1</td>
<td>(\langle 1, 1\rangle)</td>
</tr>
<tr>
<td>Q2</td>
<td>Flow 2</td>
<td>2</td>
<td>(\langle 1, 3\rangle)</td>
</tr>
</tbody>
</table>

**DRGPS: An Example**
Properties of DRGPS

DRGPS achieves all desired scheduling properties

- Predictable service isolation
- Truthfulness
- Work conservation

DRGPS therefore serves as an *idealized* fluid fair queueing benchmark in the multi-resource setting

*Cannot* be implemented because packets are assumed to be infinitely divisible
Packet-Based Multi-Resource Fair Queueing
DRGPS offers a design guideline

Leverage the design techniques developed for the traditional single-resource fair queueing

Schedule packets by emulating DRGPS

WFQ, WF²Q, FQS can have direct extensions to multiple resources

Approximate DRGPS without strict emulation

Estimate the work progress (virtual time) of DRGPS, e.g., SCFQ, SFQ, etc.

DRFQ [Ghodsi12] is a multi-resource SFQ extension

Serve flows in a simple round-robin fashion

Deficit Round Robin (DRR), Smoothed Round Robin (SRR), Stratified Round Robin (StRR)
Schedule packets by emulating DRGPS
Emulating DRGPS in Real-Time

DRGPS can be accurately emulated by stamping two service tags upon packet arrival

Virtual time $v(t)$

Tracks the work progress of DRGPS

Virtual starting time

The virtual time when the packet arrives the system

Virtual finishing time

The virtual time when packet finishes service under the DRGPS system
Proposition 4: Under DRGPS, for every flow $i$, its virtual starting and finishing times satisfy the following relationship:

$$S_i^k = \max\{F_i^{k-1}, v(a_i^k)\},$$

$$F_i^k = \frac{\tau_i^{k*}}{w_i} + S_i^k,$$

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

**Emulating DRGPS in Real-Time (Cont’d)**
Upon a packet arrival, both the starting time and the finishing time are stamped to the packet.

With the service tags, the scheduling results of DRGPS can be fully recovered.

Just like how GPS is emulated in the single-resource setting.
Schedule Packets by Emulating DRGSPS

A referencing DRGSPS system is maintained in background

Many scheduling choices are available

Packet that *finishes service the earliest* in the reference DRGSPS system is scheduled first, e.g., WFQ, PGPS

Packets that *starts service the earliest* in the reference DRGSPS system is scheduled first, e.g., FQS

Imposing some admission control policy, e.g., WF$^2$Q
A Case Study: 
Dominant Resource WF²Q
A referencing DRGPS system is maintained in background whenever there is a scheduling opportunity.

Packets that already started their service under the referencing DRGPS system are *eligible* for scheduling. Among them, the one that *finishes the earliest* will be scheduled.
A Running Example

Flow 1 sends P1, P2, ...
   Each packet requires <1 CPU time, 2 Transmission Time>

Flow 2 sends Q1, Q2, ...
   Each packet requires <3 CPU time, 1 Transmission Time>
A Running Example

(a) Scheduling outcome under DRGSP.

(b) Scheduling outcome under DRWF$^2$Q.
Relative fairness bound (RFB)

\[
R = \sup_{t_1,t_2;i,j \in B(t_1,t_2)} \left| \frac{T_i(t_1,t_2)}{w_i} - \frac{T_j(t_1,t_2)}{w_j} \right|
\]

DRGPS has RFB = 0
Proposition 6: Under DRWF$^2$Q, for any two flows $i$ and $j$ that are backlogged in $(t_1, t_2)$, we have
\[
\left|\frac{T_i(t_1, t_2)}{w_i} - \frac{T_j(t_1, t_2)}{w_j}\right| \leq 4 \max \left\{ \frac{\tau_i^{\max}}{w_i}, \frac{\tau_j^{\max}}{w_j} \right\}. \quad (23)
\]

Corollary 1 (RFB): The RFB of DRWF$^2$Q is
\[
R = 4 \max_i \left\{ \frac{\tau_i^{\max}}{w_i} \right\}.
\]
Conclusion

**DRGPs generalizes GPS to the multi-resource setting in MBs**

Offers perfect service isolation that is immune to any strategic behaviours and is work conserving as well.

Serves as a perfect multi-resource fair queueing benchmark to which all practical alternatives should approximate.

**With DRGPs, techniques developed for traditional fair queueing can be leveraged to the multi-resource setting**

**We design DRWF$^2$Q as a case study and analyze its fairness performance.**
Thanks!

http://iqua.ece.toronto.edu/~weiwang/