Networks of Multi-Server Queues with Parallel Processing

Céline Comte
Joint work with Thomas Bonald

Télécom ParisTech, France

YEQT - November 9, 2016

Introduction

Datacenters

- Specialized servers
- Massively parallel processing
- Highly variable job requirements

Examples

- Computer clusters
- Content Delivery Networks

Objective: Resource allocation with predictable performance

Background on Order Independent queues

Multi-server queues with parallel processing

Scheduling algorithm

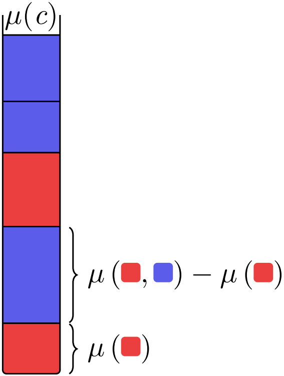
Queue state

- Finite set of job classes $I = \{$🟦, 🟥$\}$
  - Poisson external arrivals per class
  - Exponential job sizes with unit mean
- Queue state = sequence of job classes, ordered by arrival

$c = ($🟥, 🟦, 🟥, 🟦, 🟦$)$

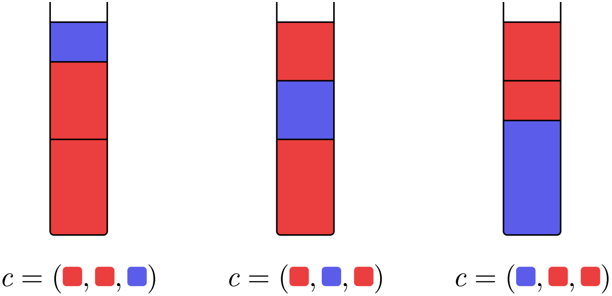
↑ head ↑ tail

- Total service rate in state $c = \mu(c)$



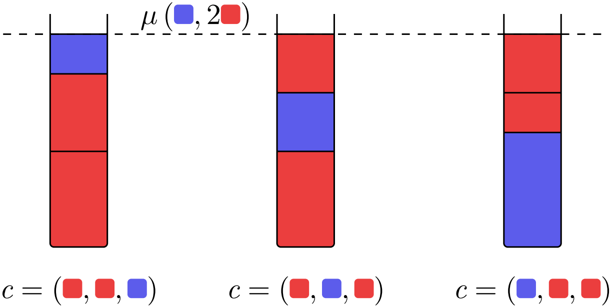
Service rate

- **Normalized:** $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing:** $\mu(c) \leq \mu(c, i)$ for any state c and class i
- **Order-independent:**



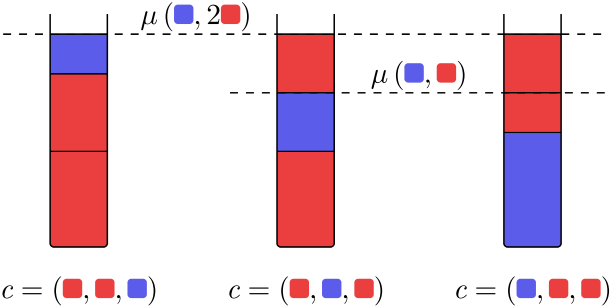
Service rate

- **Normalized:** $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing:** $\mu(c) \leq \mu(c, i)$ for any state c and class i
- **Order-independent:**



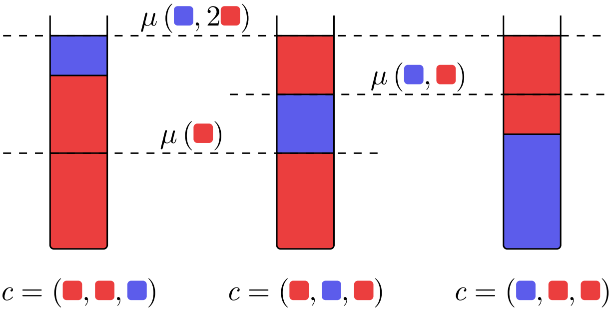
Service rate

- **Normalized:** $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing:** $\mu(c) \leq \mu(c, i)$ for any state c and class i
- **Order-independent:**



Service rate

- **Normalized:** $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing:** $\mu(c) \leq \mu(c, i)$ for any state c and class i
- **Order-independent:**



Order Independent queues (Berezner and Krzesinski, 1996)

- A stationary measure of the state $c = (c_1, \dots, c_n)$ is

$$\pi(c) = \pi(\emptyset) \prod_{k=1}^{n} \frac{\lambda_{c_k}}{\mu(c_1, \dots, c_k)}, \quad \forall c \in I^*$$

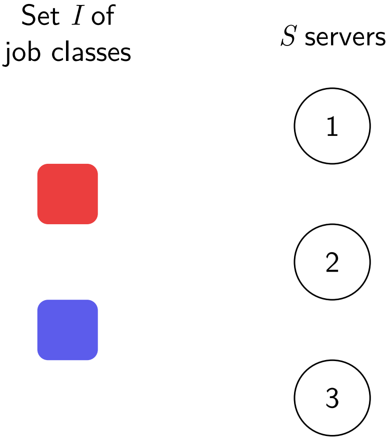
- The queue is **quasi-reversible**
  - The current state of the queue is independent of previous departures and future arrivals
  - Arrivals and departures form independent Poisson processes

Background on Order Independent queues

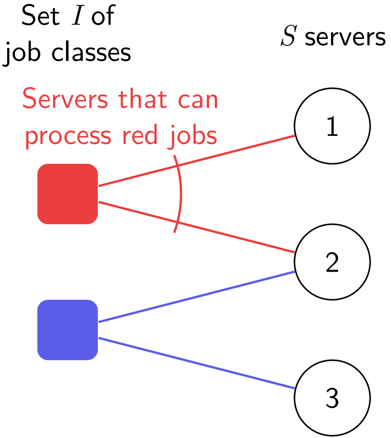
Multi-server queues with parallel processing

Scheduling algorithm

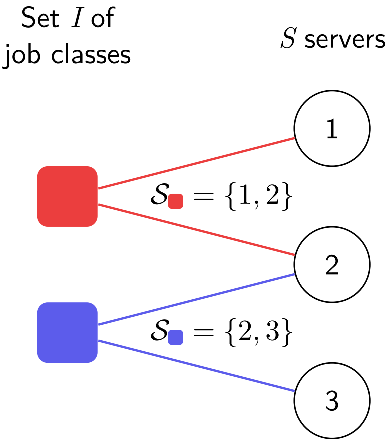
Server assignment



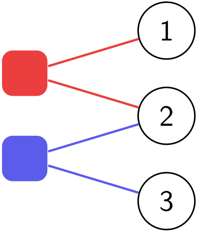
Server assignment



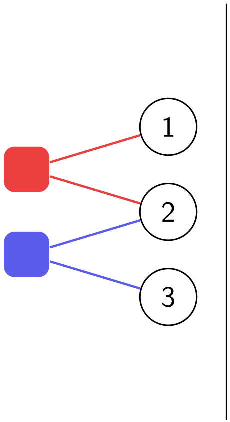
Server assignment



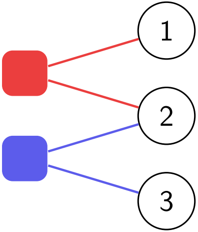
Service discipline



Service discipline

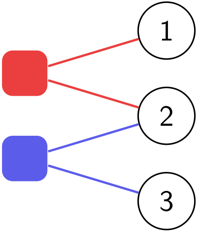


Service discipline



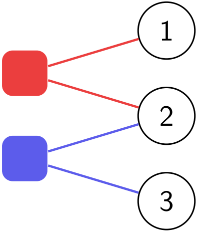
- Parallel processing

Service discipline



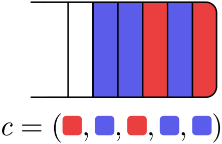
- Parallel processing
- First-come first-served per server

Service discipline

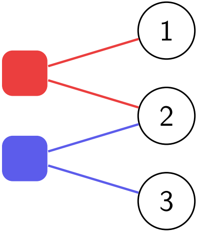


- Parallel processing
- First-come first-served per server

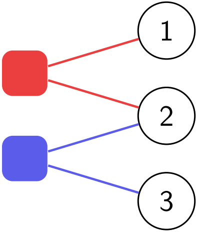
State of the queue

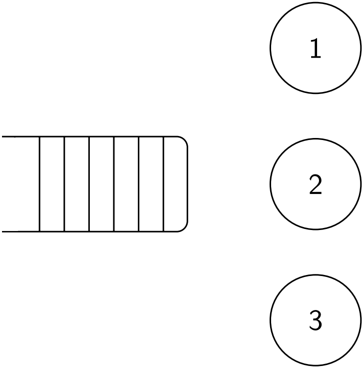


Service discipline

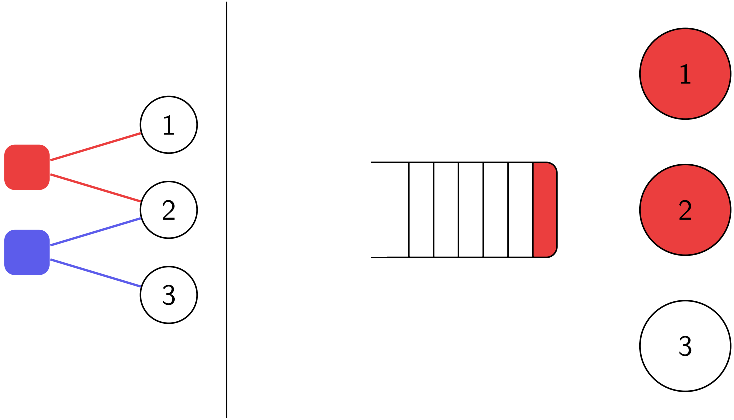


Service discipline

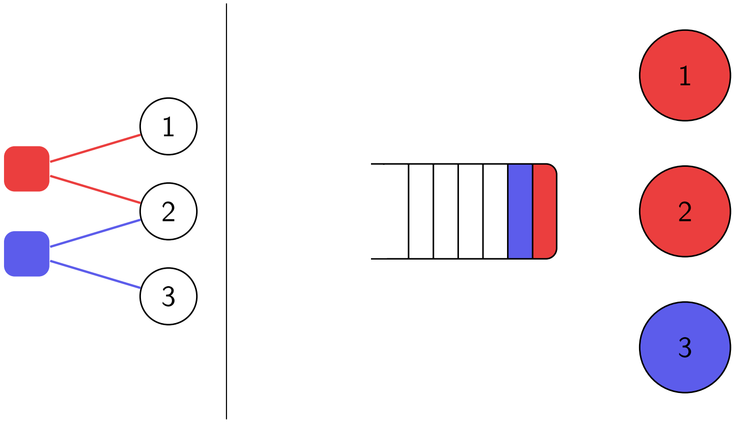




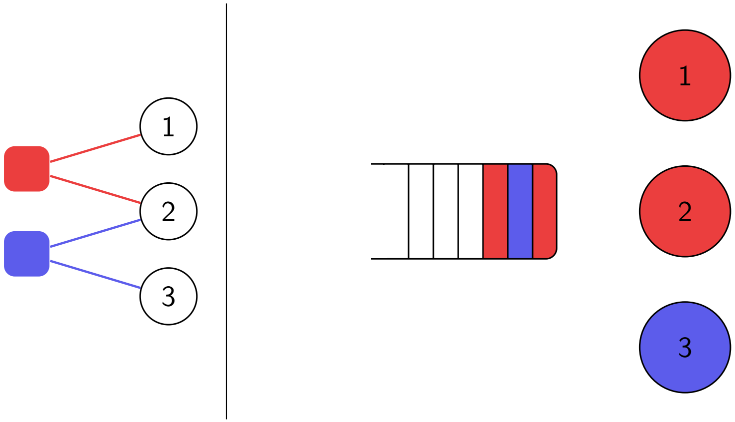
Service discipline



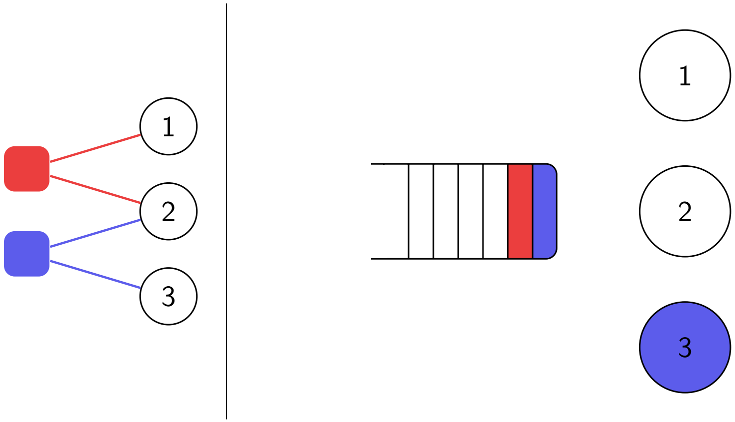
Service discipline



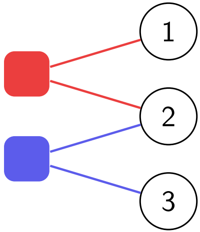
Service discipline

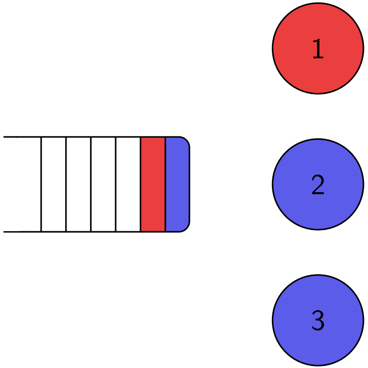


Service discipline

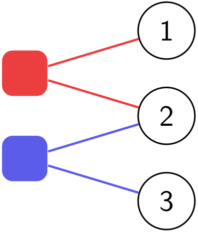


Service discipline





Service rate



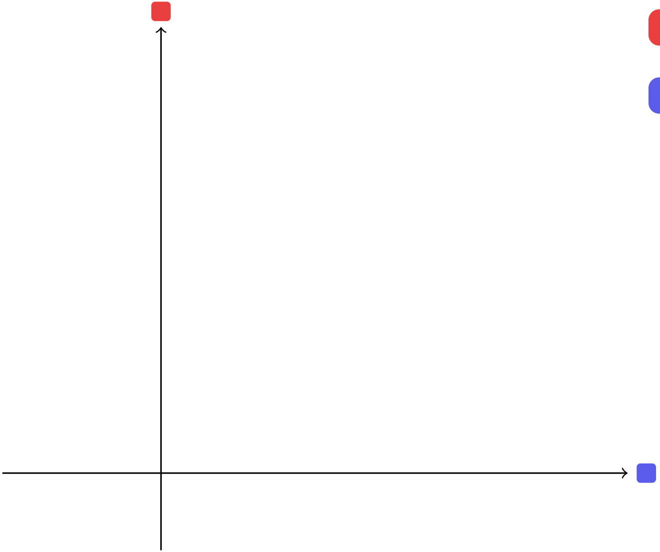
- Reinterpretation of (Gardner et al., 2015)
- μ only depends on the set of active classes

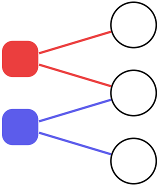
$$\mu(A) = \sum_{s \in \bigcup_{i \in A} \mathcal{S}_i} \text{capacity of server } s$$

- μ is submodular: if $A \subset B$ and $i \notin B$,

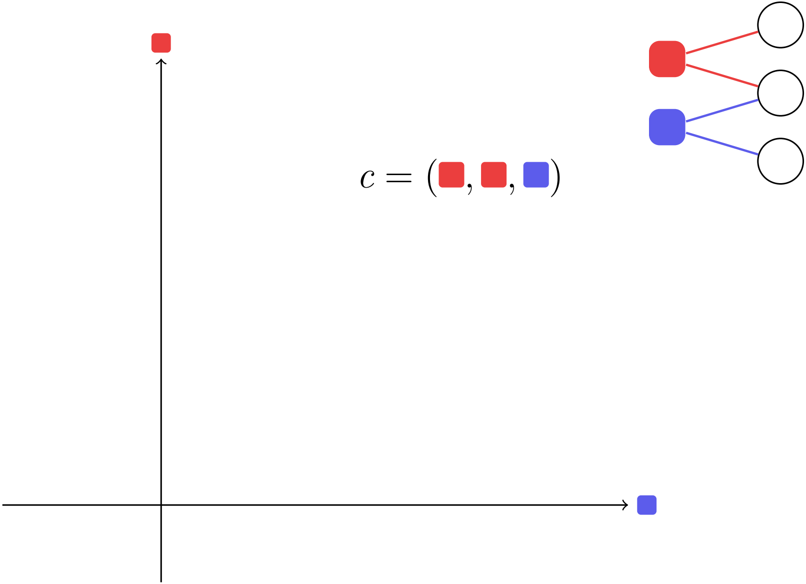
$$\mu(A \cup \{i\}) - \mu(A) \geq \mu(B \cup \{i\}) - \mu(B)$$

Service rate

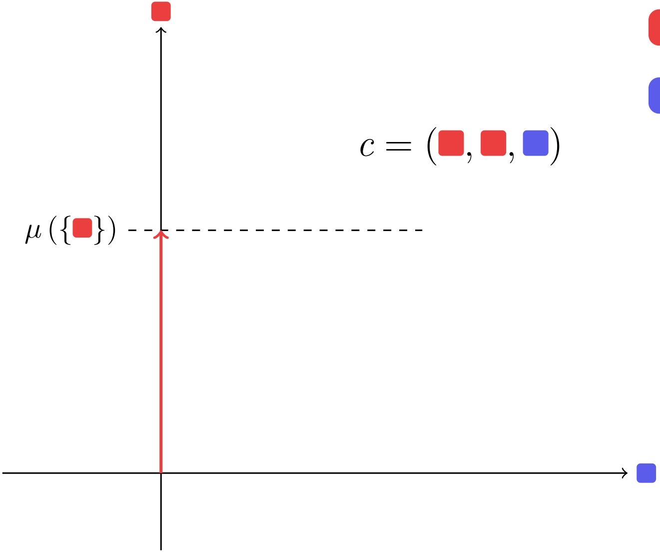




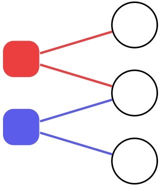
Service rate



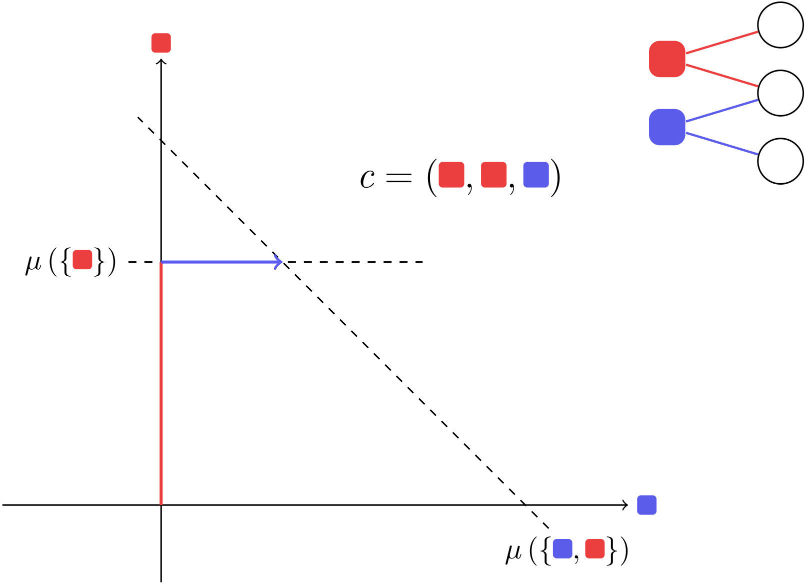
Service rate



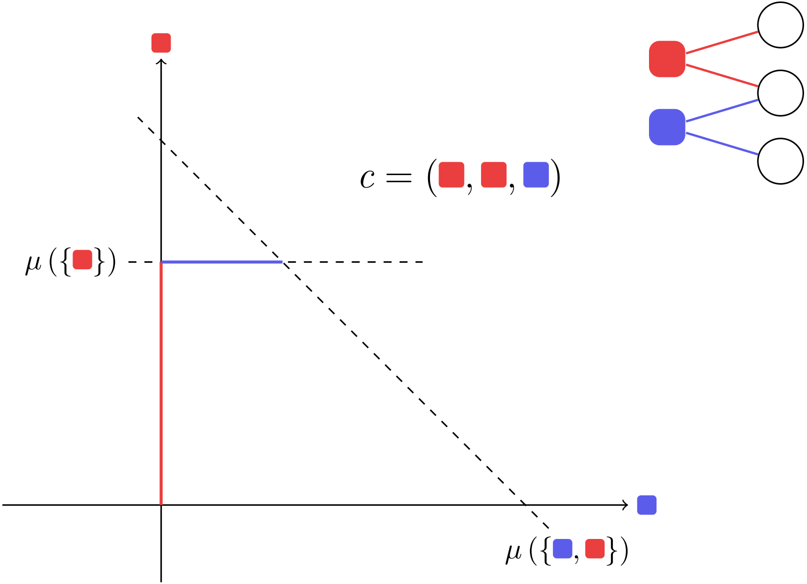
$$c = (■, ■, ■)$$



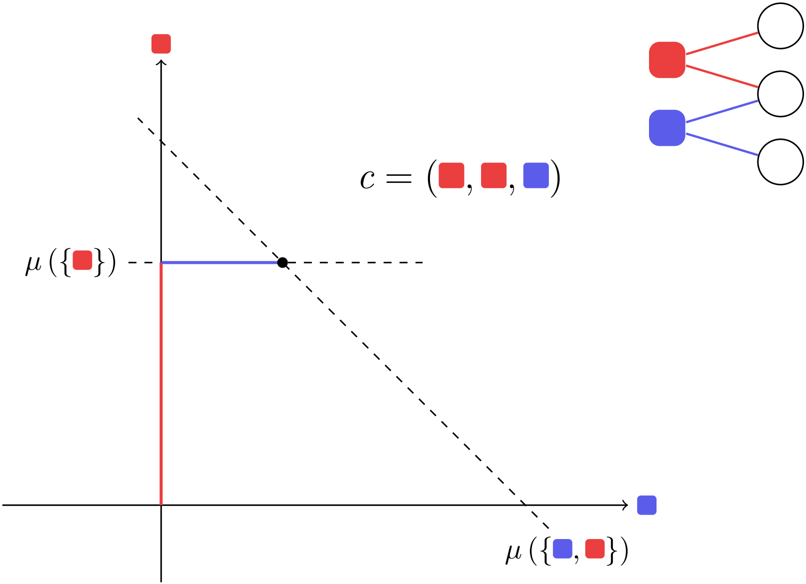
Service rate



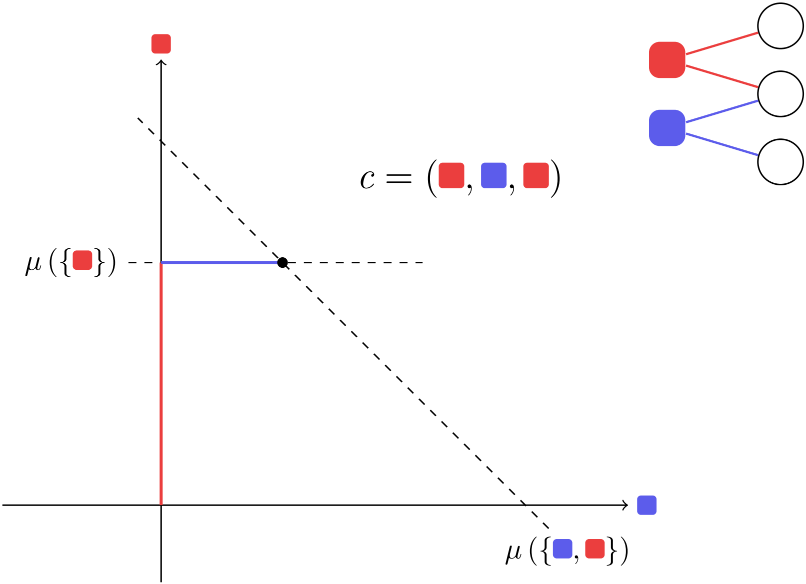
Service rate



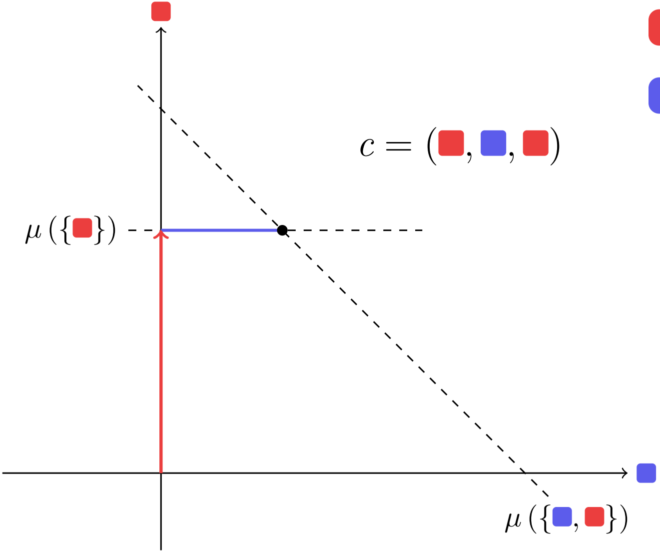
Service rate

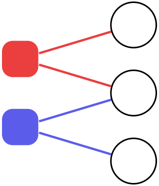


Service rate

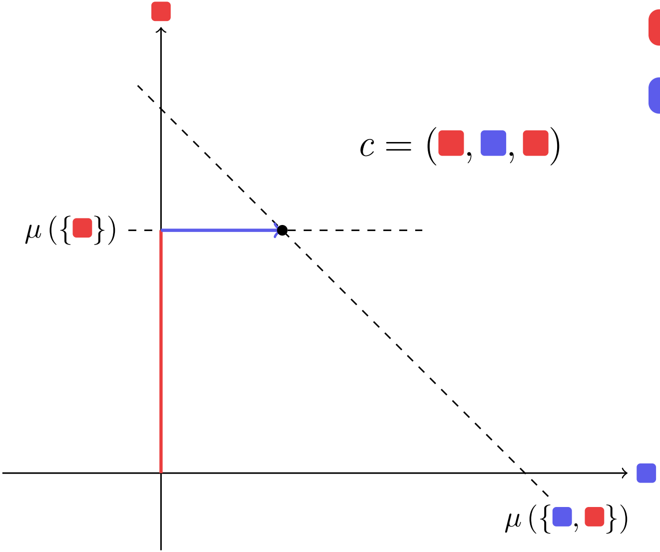


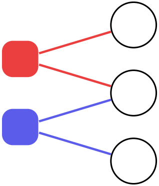
Service rate



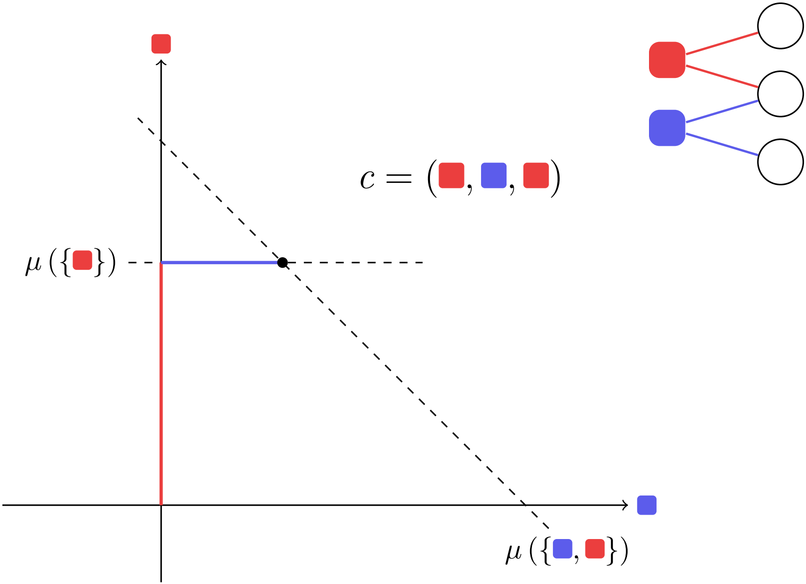


Service rate

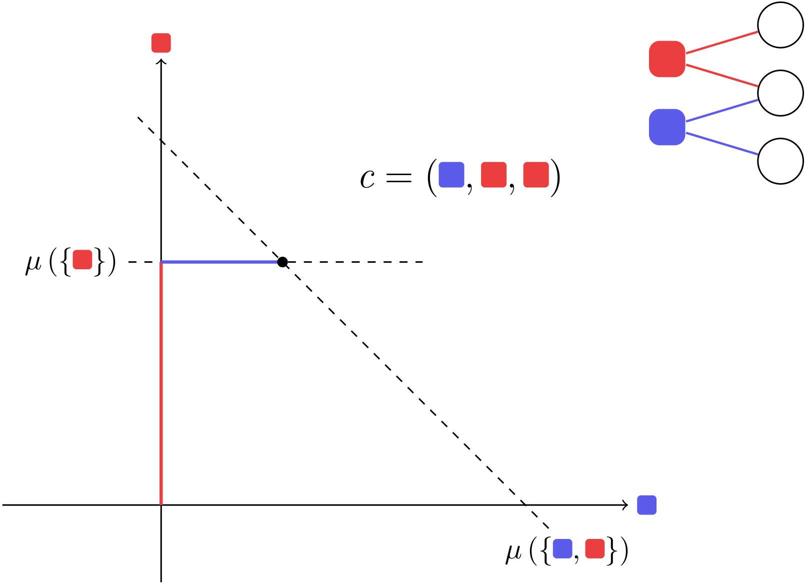




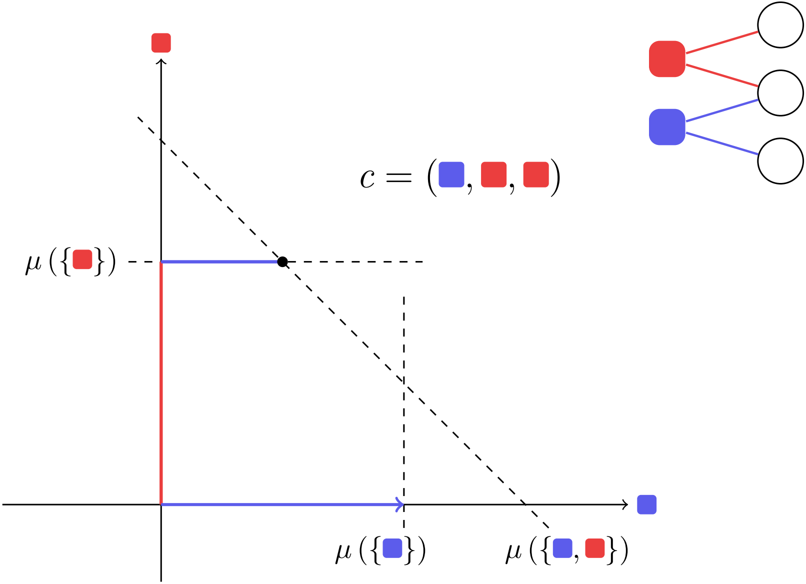
Service rate



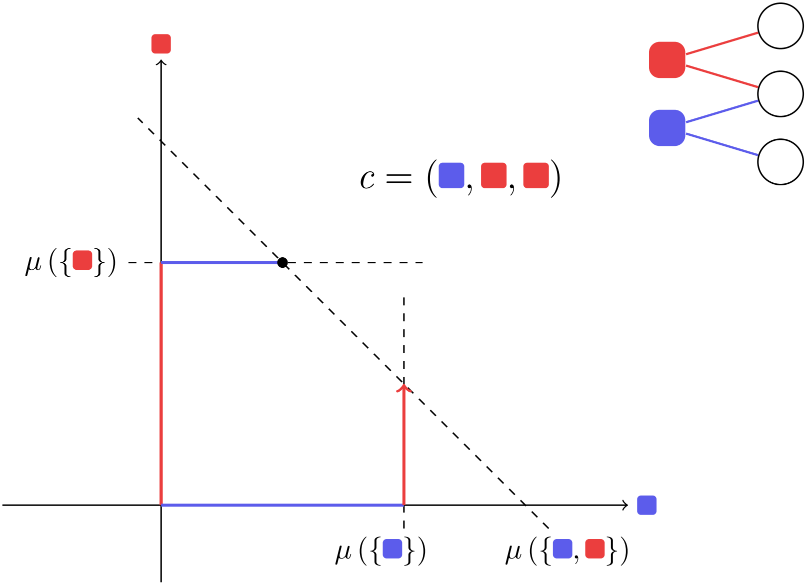
Service rate



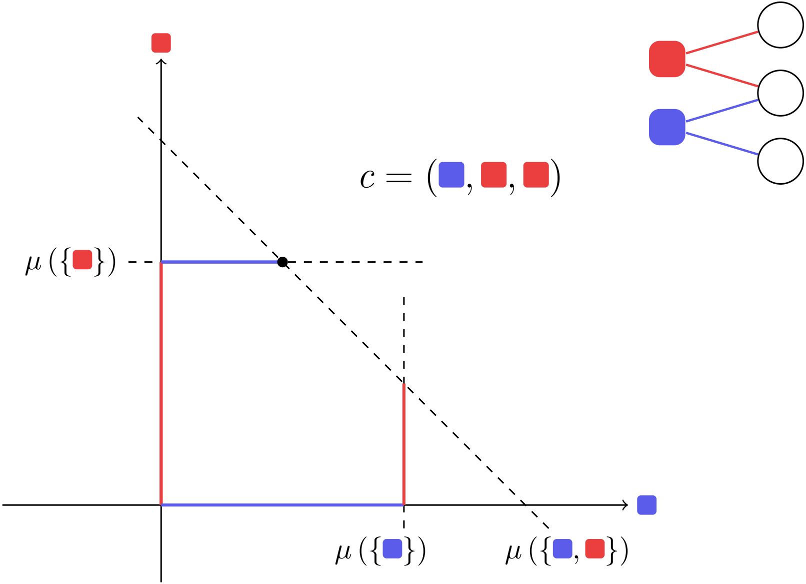
Service rate



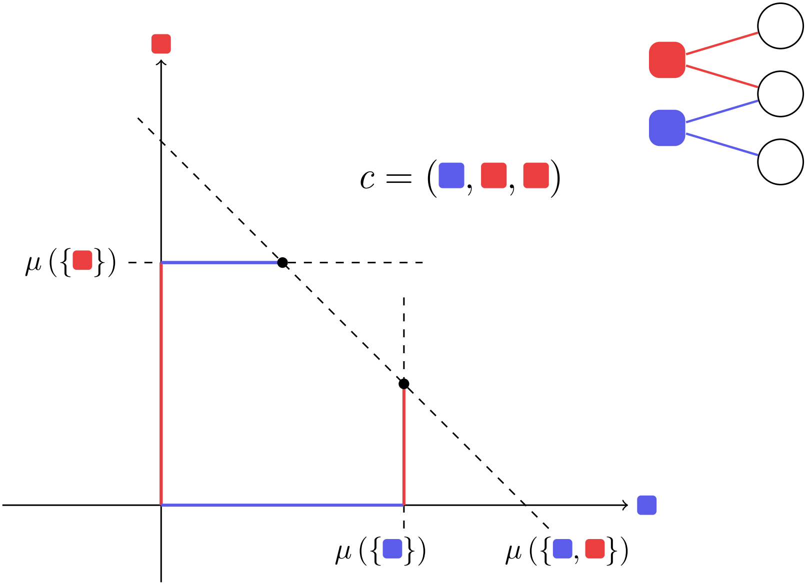
Service rate



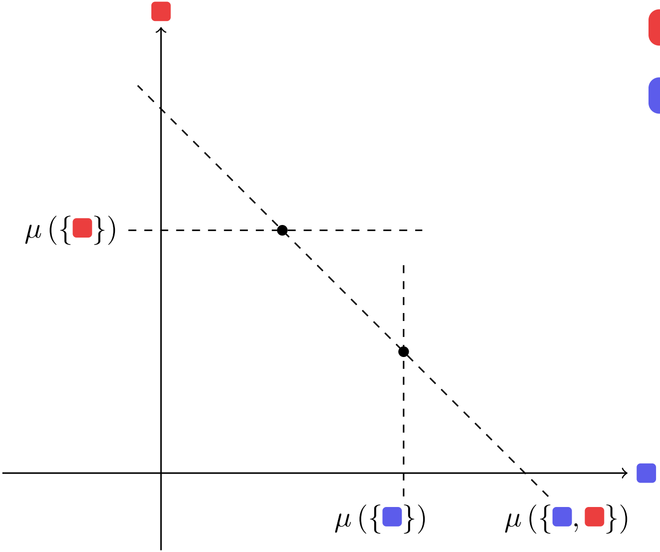
Service rate

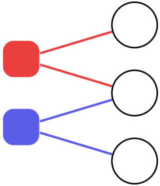


Service rate

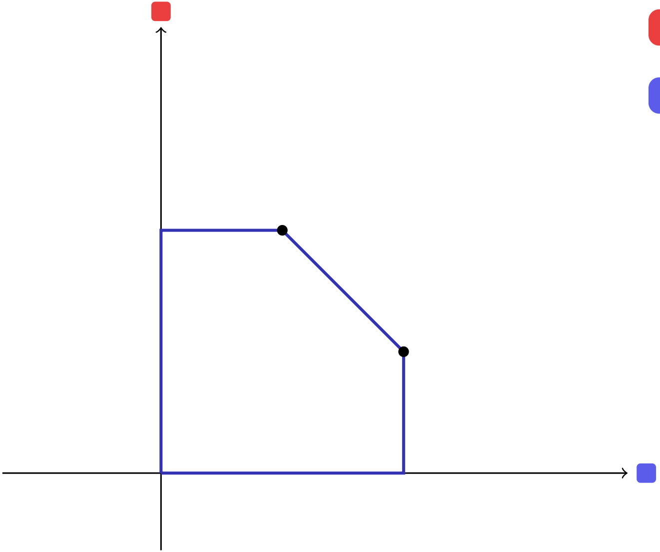


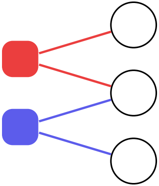
Service rate



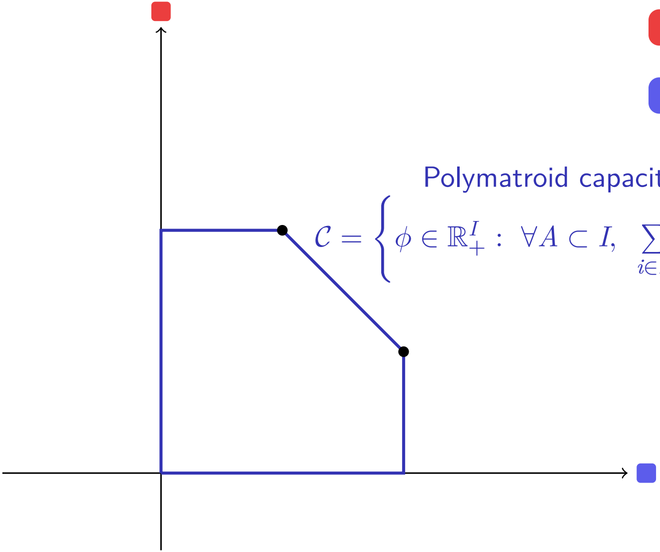


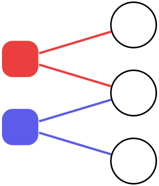
Service rate





Service rate





Polymatroid capacity set

$$\mathcal{C} = \left\{ \phi \in \mathbb{R}_+^I : \forall A \subset I, \sum_{i \in A} \phi_i \leq \mu(A) \right\}$$

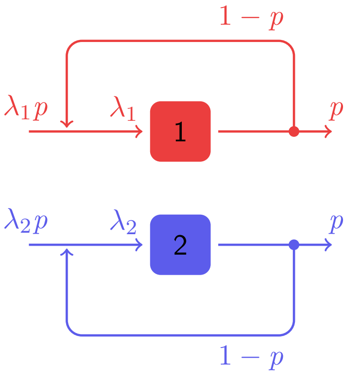
Stationary measure (Berezner and Krzesinski, 1996)

- A stationary measure of the queue state is

$$\pi(c) = \pi(\emptyset) \prod_{k=1}^{n} \frac{\lambda_{c_k}}{\mu(A(c_1, \ldots, c_k))}, \quad \forall c \in I^*$$

- The queue is **quasi-reversible**
  - The current state of the queue is independent of previous departures and future arrivals
  - Arrivals and departures form independent Poisson processes

Internal routing



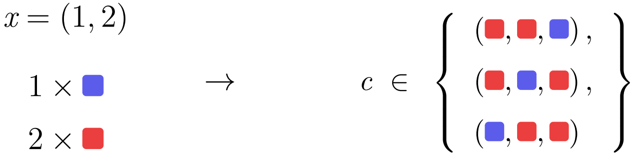
By quasi-reversiblity, the stationary measure of the queue state is

$$\pi(c) = \pi(\emptyset) \prod_{k=1}^{n} \frac{\lambda_{c_k}}{\mu(A(c_1, \ldots, c_k))},$$

independently of $p \in [0, 1]$.

State aggregation

Aggregate state $x = (x_i : i \in I) \in \mathbb{N}^I$



Stationary measure $\pi(x) = \sum_{c : |c| = x} \pi(c)$

Stationary distribution (Berezner and Krzesinski, 1996)

The stationary measure of the aggregate state x satisfies

$$\pi(x) = \pi(0)\Phi(x) \prod_{i \in I} {\lambda_i}^{x_i}, \quad \forall x \in \mathbb{N}^I,$$

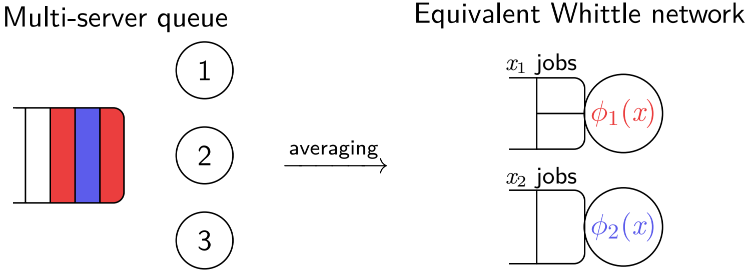
where Φ is defined by the recursion $\Phi(0) = 1$ and, for each $x \neq 0$,

$$\Phi(x) = \frac{1}{\mu(A(x))} \sum_{i \in A(x)} \Phi(x - e_i).$$

Equivalent Whittle network

π is the stationary measure of the state of a **Whittle network** of $|I|$ queues with arrival rates $\lambda_i, i \in I$, and service rates

$$\phi_i(x) = \frac{\Phi(x - e_i)}{\Phi(x)}, \quad \forall x \in \mathbb{N}^I, \quad \forall i \in A(x).$$



Equivalent Whittle network

Per-class service rates:

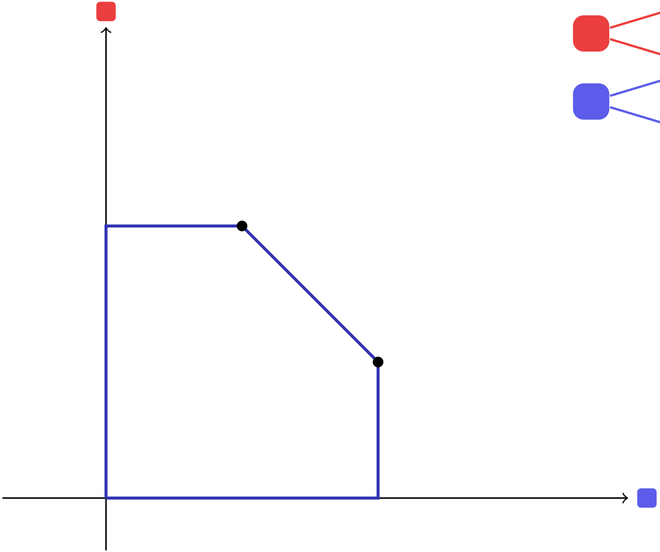
- $\mu_i(c)$ in state c of the multi-server queue
- $\phi_i(x)$ in state x of the Whittle network

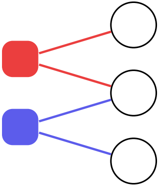
Theorem 1

The service rates in the equivalent Whittle network are the average per-class service rates in the multi-server queue:

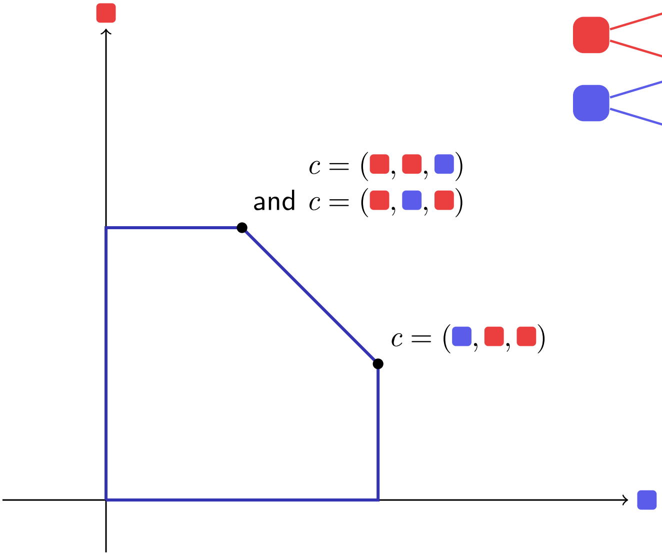
$$\phi_i(x) = \sum_{c:|c|=x} \frac{\pi(c)}{\pi(x)} \mu_i(c), \quad \forall x \in \mathbb{N}^I, \quad \forall i \in A(x).$$

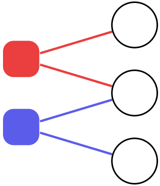
Average service rates



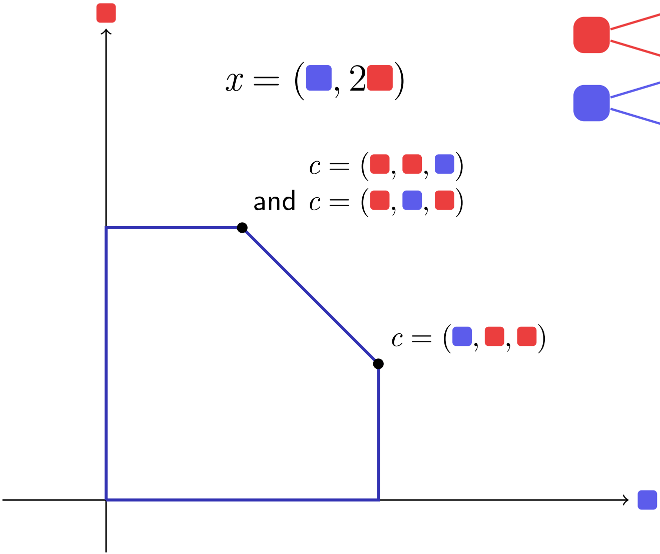


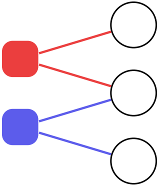
Average service rates



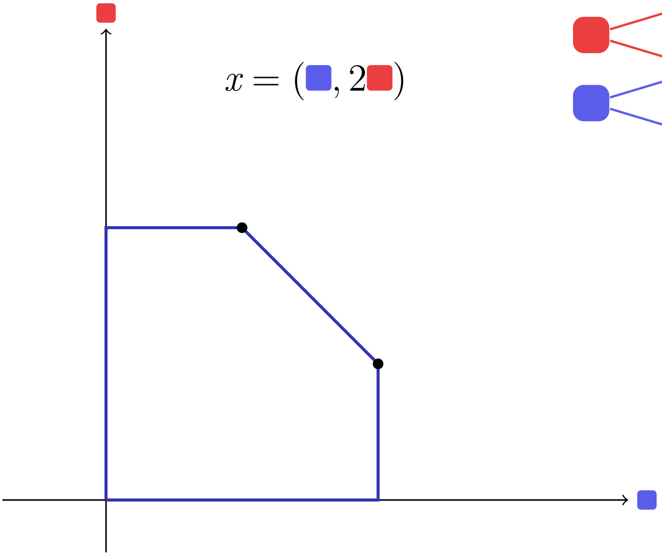


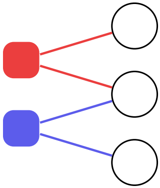
Average service rates



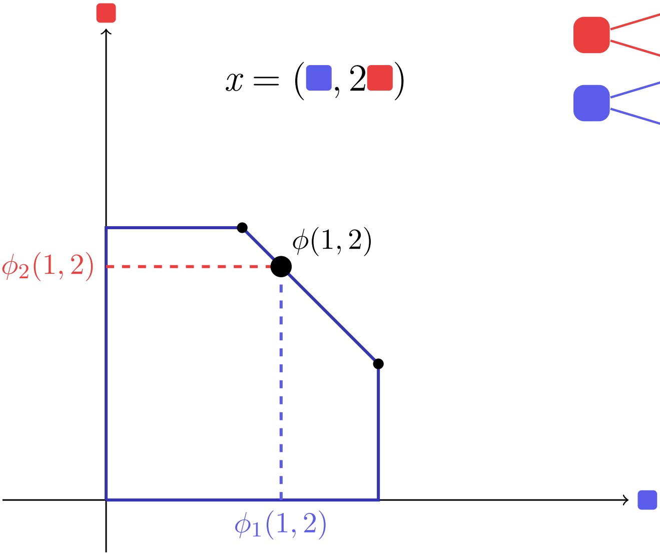


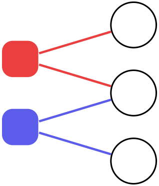
Average service rates





Average service rates





Balanced fairness

The average service rates are

- Balanced: $\dfrac{\phi_i(x-e_j)}{\phi_i(x)} = \dfrac{\phi_j(x-e_i)}{\phi_j(x)}, \quad \forall x \in \mathbb{N}^I, \quad \forall i,j \in A(x),$
- Efficient: $\displaystyle\sum_{i \in I} \phi_i(x) = \mu(A(x)), \quad \forall x \in \mathbb{N}^I.$

Balanced fairness in the capacity set

$$\mathcal{C} = \left\{ \phi \in \mathbb{R}_+^I : \quad \forall A \subset I, \quad \sum_{i \in A} \phi_i \le \mu(A) \right\}$$

Balanced fairness

The most efficient **insensitive** resource allocation

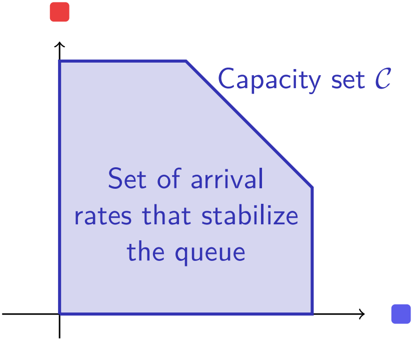
- Introduced for dimensioning data networks (Bonald and Proutière, 2003)
- Good approximation of proportional fairness
- Recently applied to Content Delivery Networks (Shah and de Veciana, 2015, 2016)

Stability condition

Theorem 2

The multi-server queue is stable if and only if

$$\forall A \subset I, \quad \sum_{i \in A} \lambda_i < \mu(A)$$



Aggregation

- Queue state c: $\pi(c) = \frac{\lambda_{c_n} \pi(c_1, \ldots, c_{n-1})}{\mu(c)}, \quad \forall c \neq \emptyset$
- Aggregate state x: $\pi(x) = \frac{\sum_{i \in A(x)} \lambda_i \pi(x - e_i)}{\mu(x)}, \quad \forall x \neq 0$
- Set of active classes A
  (Bonald et al., 2003; Shah and de Veciana, 2015, 2016)

$$\pi(A) = \frac{\sum_{i \in A} \lambda_i \pi(A \setminus \{i\})}{\mu(A) - \sum_{i \in A} \lambda_i}, \quad \forall A \neq \emptyset$$

$\rightarrow$ Closed-form expressions for the performance metrics

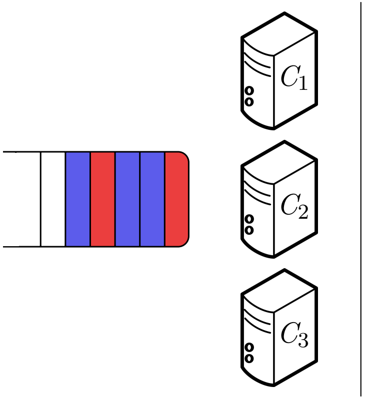
- Proportion of time the queue is idle
- Mean number of jobs of each class
- ...

Background on Order Independent queues

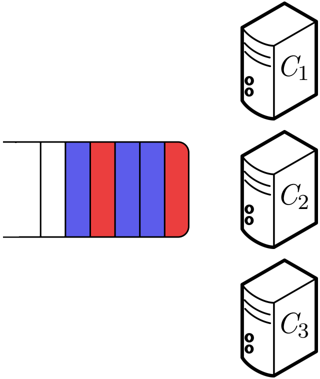
Multi-server queues with parallel processing

Scheduling algorithm

Computer cluster

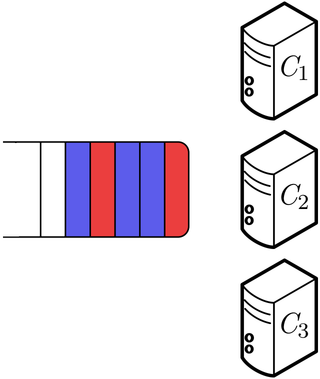


Computer cluster



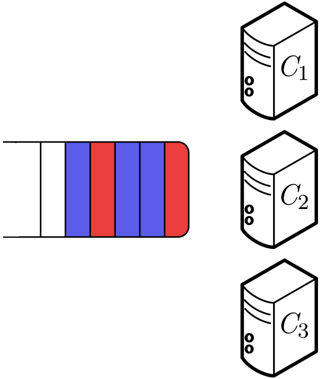
- Server s
  Fixed capacity C_s in flops

Computer cluster



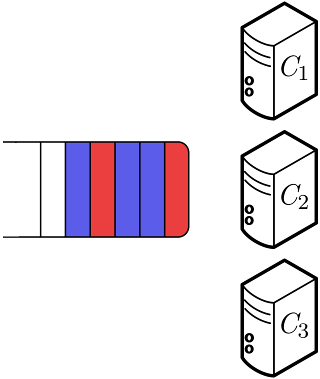
- Server s
  Fixed capacity C_s in flops
- An arriving job is assigned a set of computers *independently of the state of the cluster*

Computer cluster



- Server s
  Fixed capacity C_s in flops
- An arriving job is assigned a set of computers *independently of the state of the cluster*
- Service requirements
  General distribution with mean σ in floating-point operations

Computer cluster

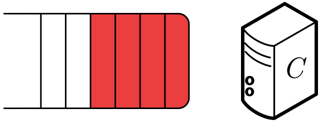


- Server s
  Fixed capacity C_s in flops
- An arriving job is assigned a set of computers *independently of the state of the cluster*
- Service requirements
  General distribution with mean σ in floating-point operations

Objective: Enforce balanced fairness

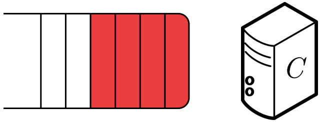
Single computer

Single-server mono-class cluster, service requirements with mean σ



Single computer

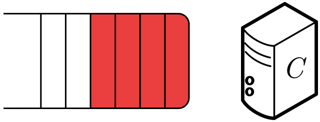
Single-server mono-class cluster, service requirements with mean σ



- FCFS at job scale
  Very sensitive

Single computer

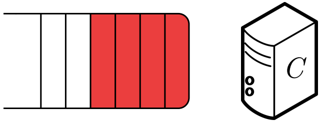
Single-server mono-class cluster, service requirements with mean σ



- FCFS at job scale
  Very sensitive
- Service interruption after θ floating point operations on average

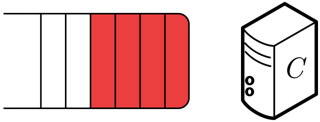
Single computer

Single-server mono-class cluster, service requirements with mean σ



Single computer

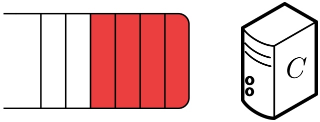
Single-server mono-class cluster, service requirements with mean σ



- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$

Single computer

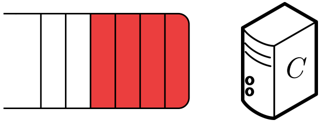
Single-server mono-class cluster, service requirements with mean σ



- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$
- Upon service completion, the job leaves the cluster

Single computer

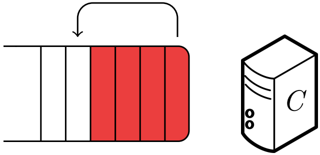
Single-server mono-class cluster, service requirements with mean σ



- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$
- Upon service completion, the job leaves the cluster
- When the timer expires, the service is interrupted

Single computer

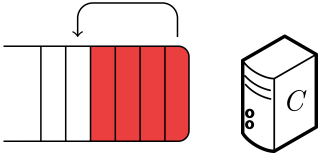
Single-server mono-class cluster, service requirements with mean σ



- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$
- Upon service completion, the job leaves the cluster
- When the timer expires, the service is interrupted

Single computer

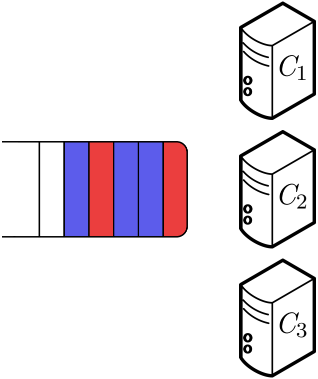
Single-server mono-class cluster, service requirements with mean σ



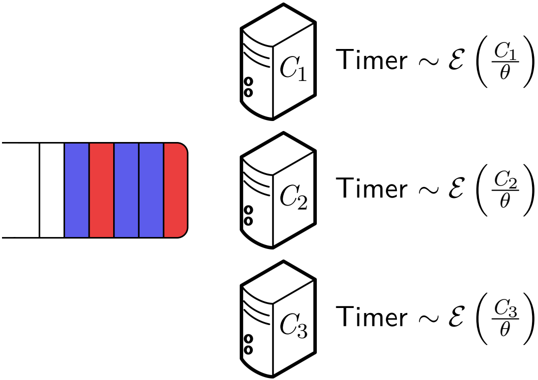
- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$
- Upon service completion, the job leaves the cluster
- When the timer expires, the service is interrupted

Parameter $m =$ mean number of interruptions per job
When $m \to \infty$, single-server queue under PS service discipline

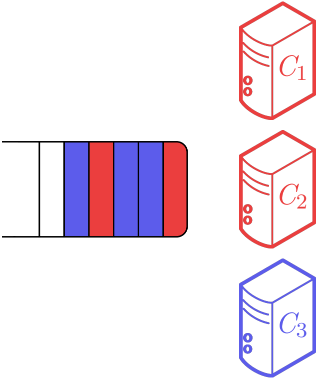
Computer cluster



Computer cluster



Computer cluster

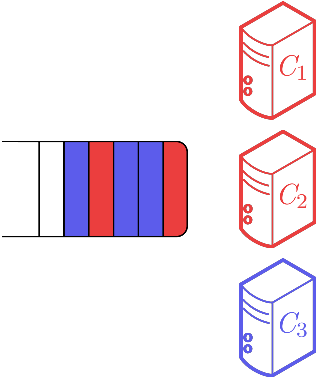


Timer $\sim \mathcal{E}\left(\frac{C_1}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_2}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

Computer cluster



Timer $\sim \mathcal{E}\left(\frac{C_1}{\theta}\right)$

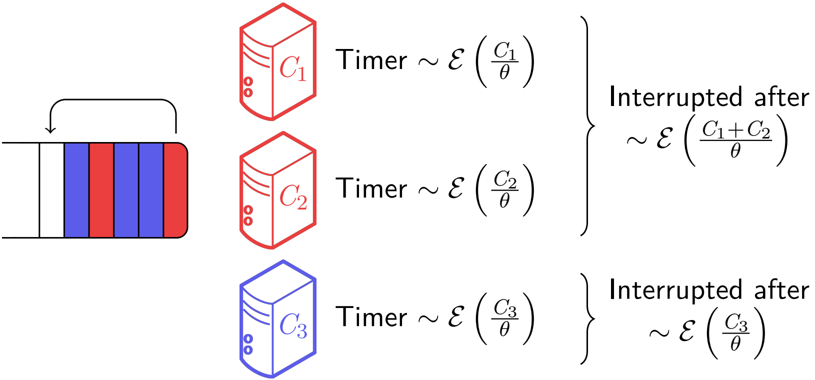
Timer $\sim \mathcal{E}\left(\frac{C_2}{\theta}\right)$

Interrupted after $\sim \mathcal{E}\left(\frac{C_1+C_2}{\theta}\right)$

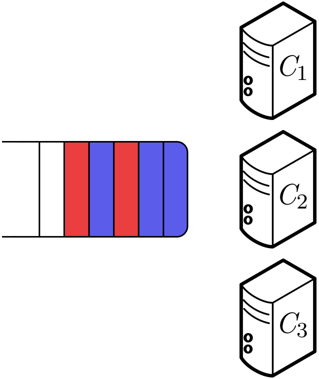
Timer $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

Interrupted after $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

Computer cluster



Computer cluster

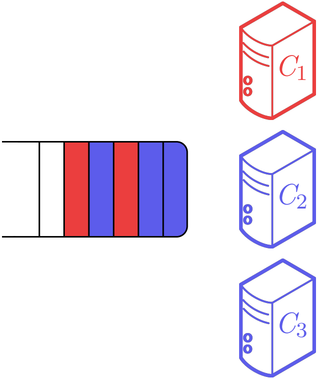


Timer $\sim \mathcal{E}\left(\frac{C_1}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_2}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

Computer cluster

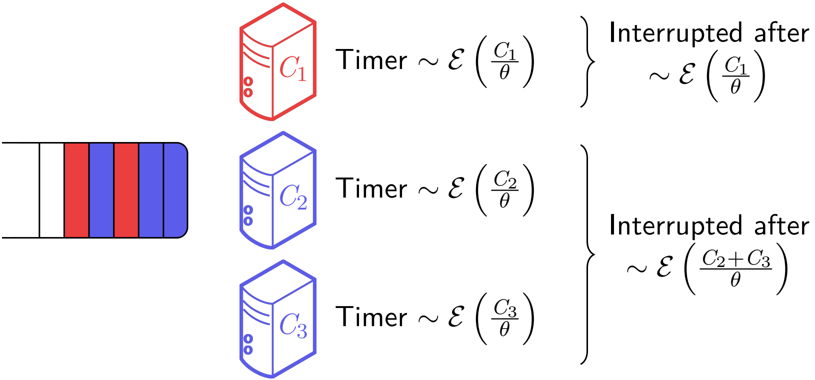


Timer $\sim \mathcal{E}\left(\frac{C_1}{\theta}\right)$

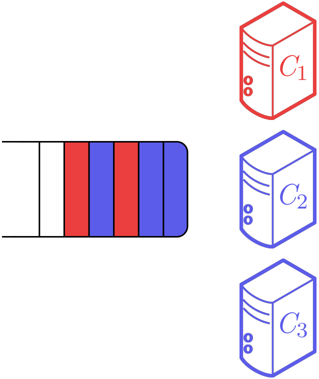
Timer $\sim \mathcal{E}\left(\frac{C_2}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

Computer cluster



Computer cluster



Timer $\sim \mathcal{E}\left(\frac{C_1}{\theta}\right)$ — Interrupted after $\sim \mathcal{E}\left(\frac{C_1}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_2}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

Interrupted after $\sim \mathcal{E}\left(\frac{C_2+C_3}{\theta}\right)$

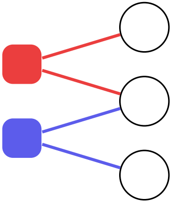
When $m \to \infty$, resources allocated according to balanced fairness

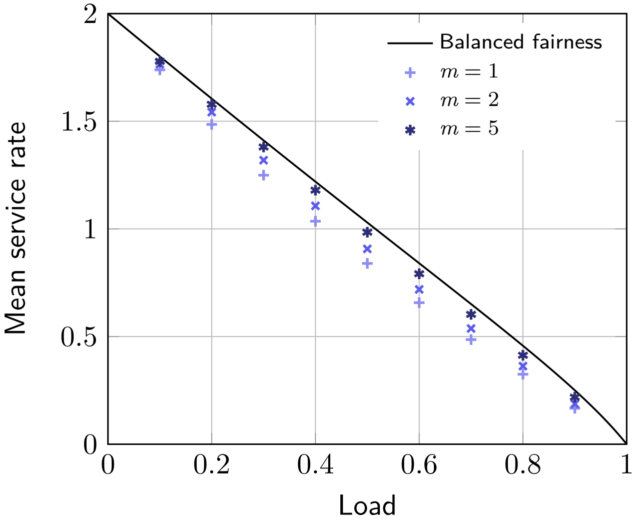
Numerical results

- Performance metric:
  Mean service rate seen by class-i jobs

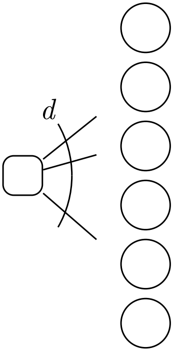
- Hyperexponential job size distribution
  $\rightarrow$ $\sim \mathcal{E}(1/5)$ with probability $1/6$
  $\rightarrow$ $\sim \mathcal{E}(5)$ with probability $5/6$

Numerical results: Shared pool





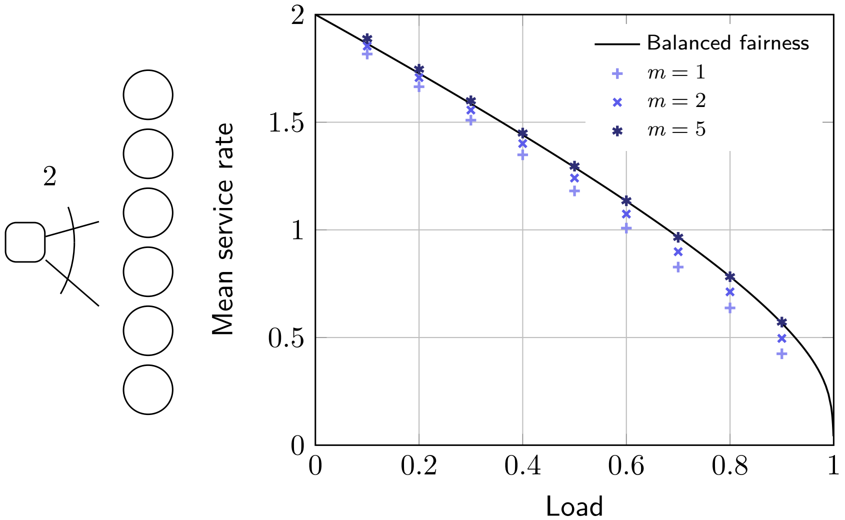
Numerical results: Random assignment



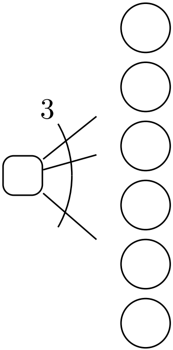
- d servers chosen uniformly and independently at random
- By (Gardner et al., 2016),

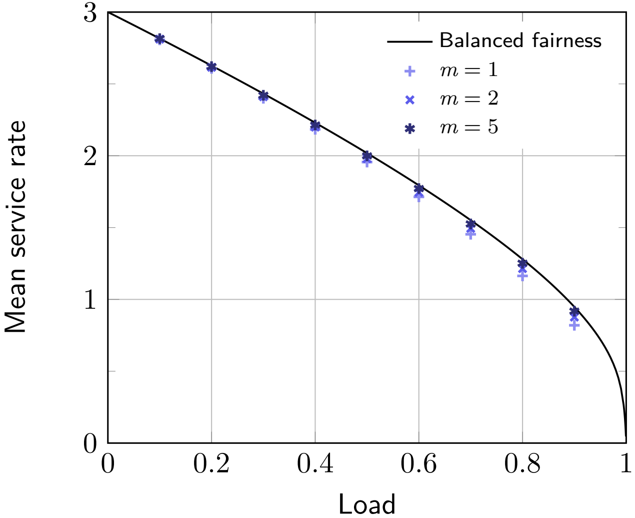
$$\frac{1}{\gamma} = \sum_{j=d}^{S} \frac{1}{S\mu \frac{\binom{S-1}{d-1}}{\binom{j-1}{d-1}} - S\lambda}$$

Numerical results: $S = 100$, $d = 2$



Numerical results: $S = 100$, $d = 3$





Conclusion

- Multi-server queues with parallel processing
  - Sequential version of a class of Whittle networks
  - Stable whenever each set of classes can handle its own load
  - Closed-form expressions for the performance metrics
- Scheduling algorithm in computer clusters
  - Service interruptions implemented by exponential timers
  - Insensitive resource allocation
- Future works
  - Generalize these results to Order Independent queues
  - Assert robustness of the algorithm

Bibliography

S. A. Berezner and A. E. Krzesinski (1996). "Order independent loss queues". In: *Queueing Systems* 23.1, pp. 331–335.

A. E. Krzesinski (2011). "Order Independent Queues". In: *Queueing Networks: A Fundamental Approach*. Ed. by R. J. Boucherie and N. M. van Dijk. Boston, MA: Springer US, pp. 85–120.

K. Gardner et al. (2015). "Reducing Latency via Redundant Requests: Exact Analysis". In: *Proceedings of ACM SIGMETRICS 2015*. Portland, Oregon, USA: ACM, pp. 347–360.

K. Gardner et al. (2016). "The Power of D Choices for Redundancy". In: *SIGMETRICS Perform. Eval. Rev.* 44.1, pp. 409–410.

Bibliography

T. Bonald and A. Proutière (2003). "Insensitive Bandwidth Sharing in Data Networks". In: *Queueing Syst.* 44.1, pp. 69–100.

T. Bonald et al. (2003). "Computational aspects of balanced fairness". In: Proceedings of the 18th International Teletraffic Congress. Berlin, Germany.

V. Shah and G. de Veciana (2015). "High-Performance Centralized Content Delivery Infrastructure: Models and Asymptotics". In: *IEEE/ACM Transactions on Networking* 23.5, pp. 1674–1687.

V. Shah and G. de Veciana (2016). "Impact of fairness and heterogeneity on delays in large-scale centralized content delivery systems". In: *Queueing Systems* 83.3, pp. 361–397.