Resource Management in Computer Clusters: Algorithm Design and Performance Analysis

Céline Comte

Nokia Bell Labs France – Télécom Paris

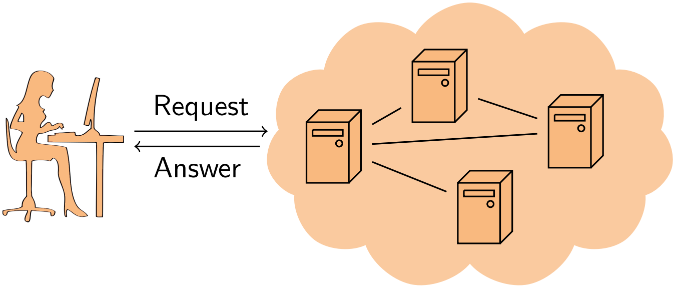
Ph.D. defense
September 24, 2019

The NIST Definition of Cloud Computing (Mell and Grance, 2011)

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a **shared pool of configurable computing resources**

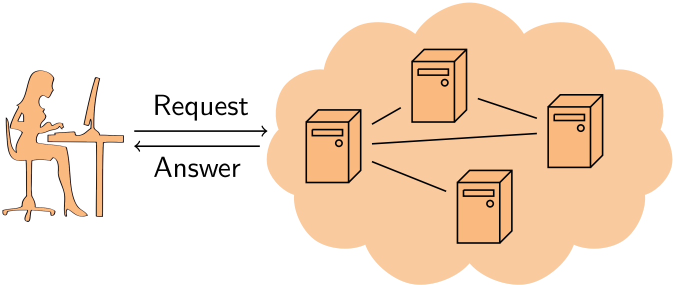
The NIST Definition of Cloud Computing (Mell and Grance, 2011)

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a **shared pool of configurable computing resources**



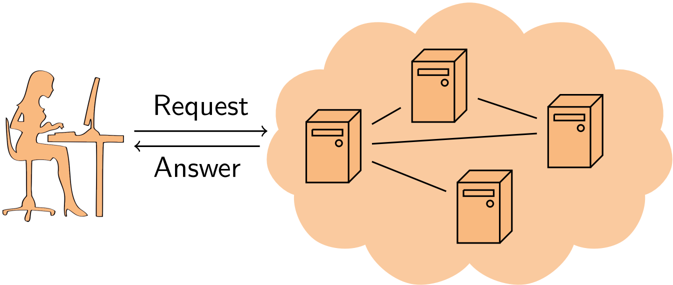
The NIST Definition of Cloud Computing (Mell and Grance, 2011)

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a **shared pool of configurable computing resources** (e.g., networks, servers, storage, applications, and services)



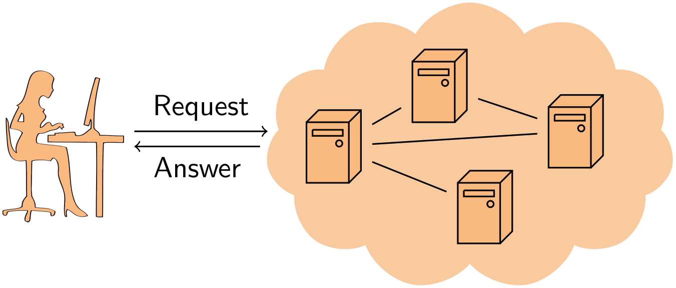
The NIST Definition of Cloud Computing (Mell and Grance, 2011)

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a **shared pool of configurable computing resources** (e.g., networks, servers, storage, applications, and services) that can be **rapidly provisioned and released** with minimal management effort or service provider interaction



The NIST Definition of Cloud Computing (Mell and Grance, 2011)

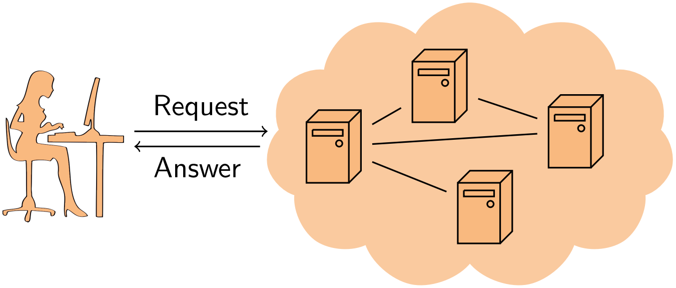
Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a **shared pool of configurable computing resources** (e.g., networks, servers, storage, applications, and services) that can be **rapidly provisioned and released** with minimal management effort or service provider interaction



- On-demand self-service
- Broad network access
- Rapid elasticity
- Measured service
- Resource pooling

The NIST Definition of Cloud Computing (Mell and Grance, 2011)

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a **shared pool of configurable computing resources** (e.g., networks, servers, storage, applications, and services) that can be **rapidly provisioned and released** with minimal management effort or service provider interaction



- On-demand self-service
- Broad network access
- Rapid elasticity
- Measured service
- **Resource pooling**

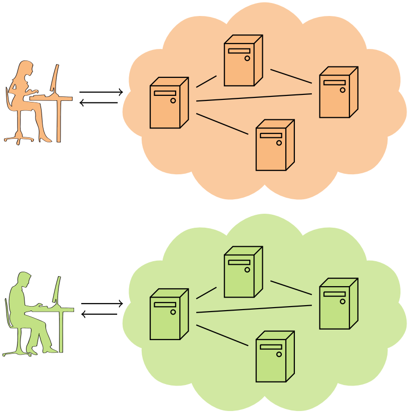
Economies of scale

Without resource pooling

With resource pooling

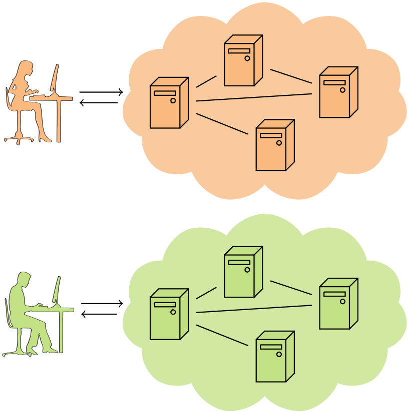
Economies of scale

Without resource pooling

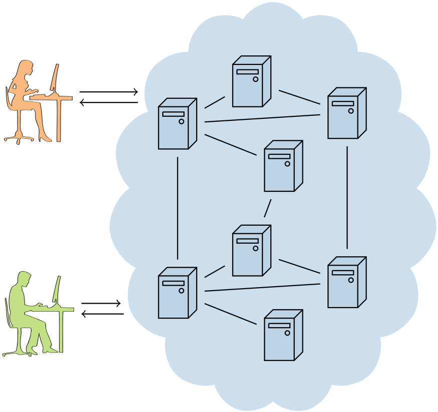


With resource pooling

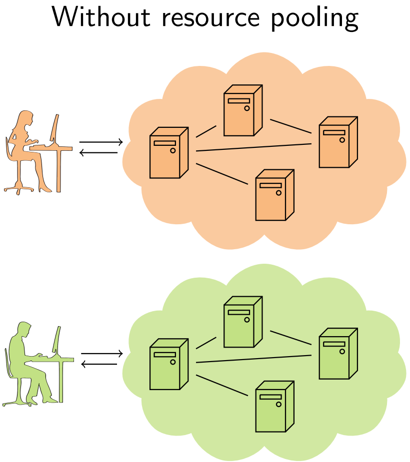
Without resource pooling

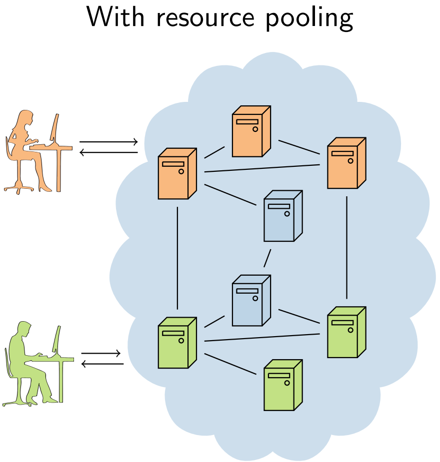


With resource pooling



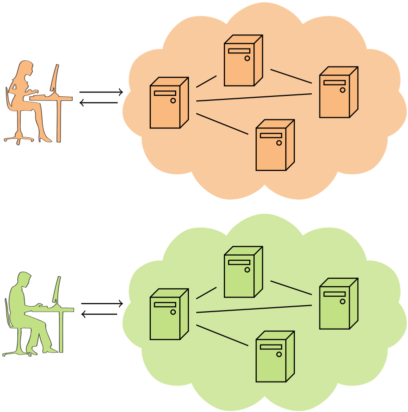
Economies of scale



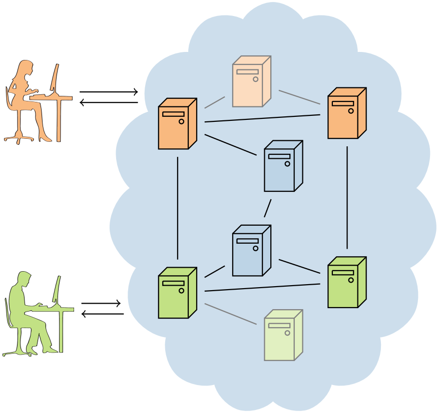


Economies of scale

Without resource pooling

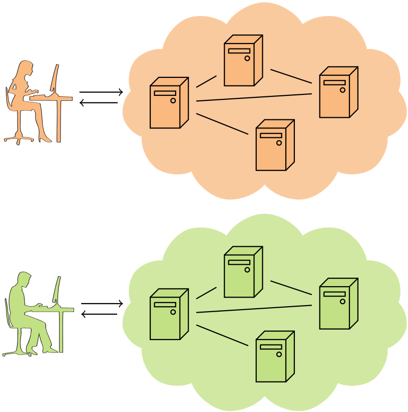


With resource pooling

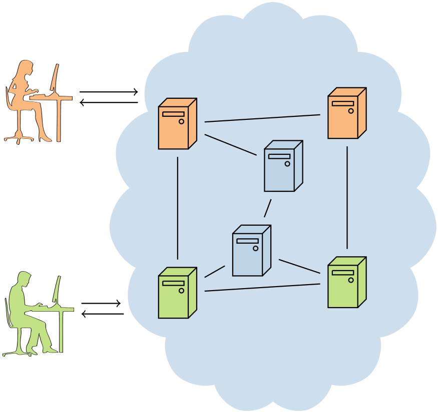


Economies of scale

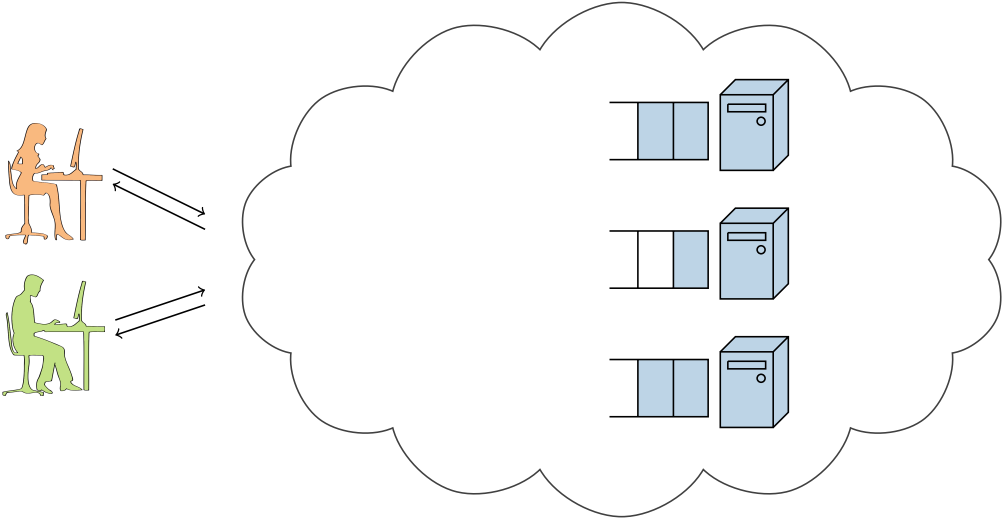
Without resource pooling



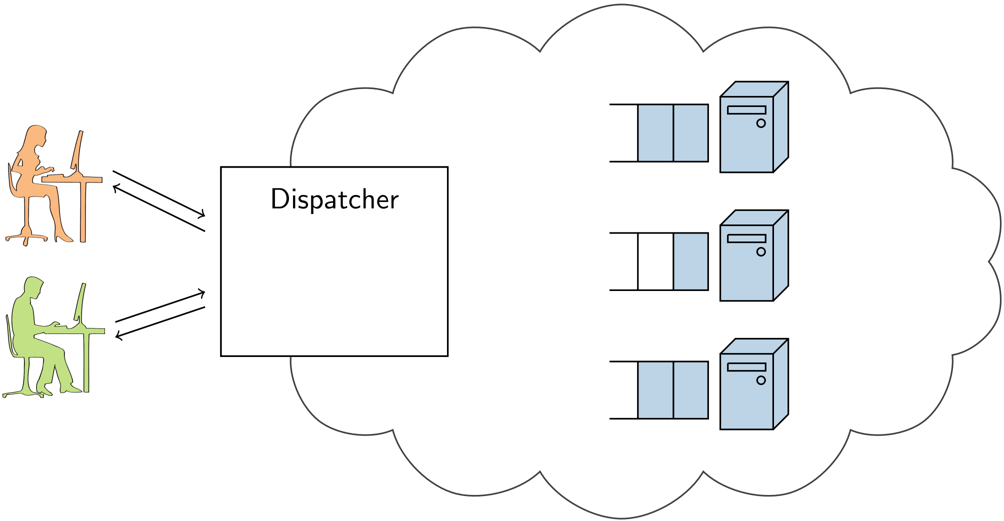
With resource pooling

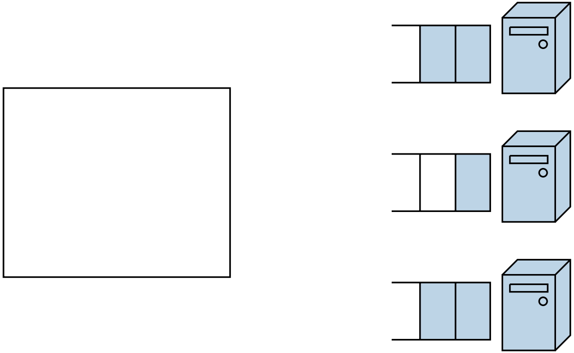


Resource-management algorithms

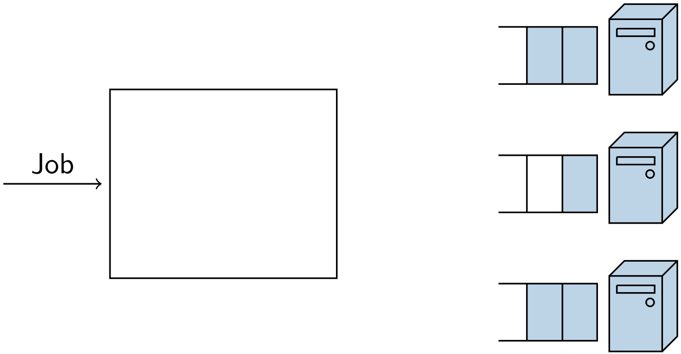


Resource-management algorithms

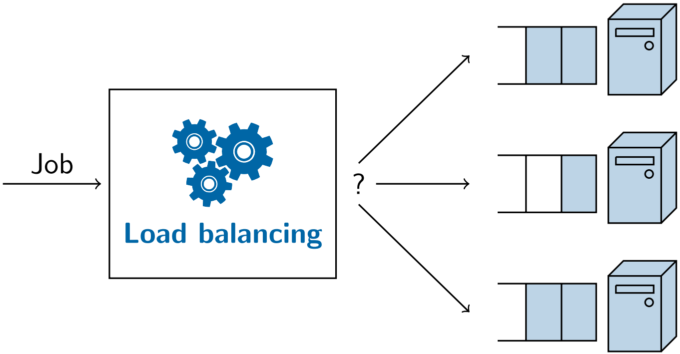




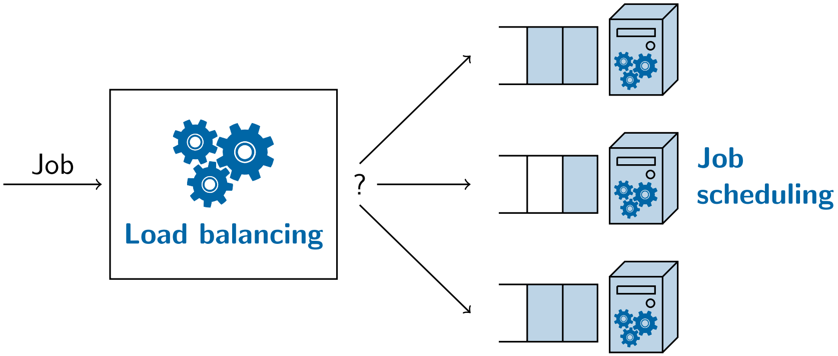
Resource-management algorithms



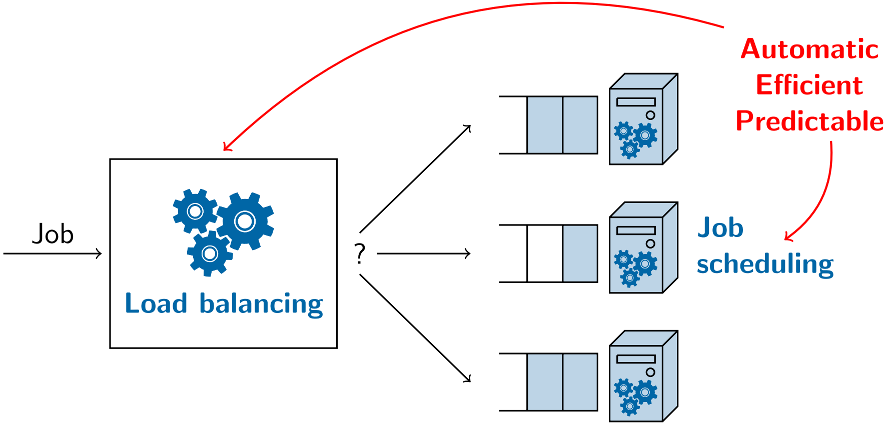
Resource-management algorithms



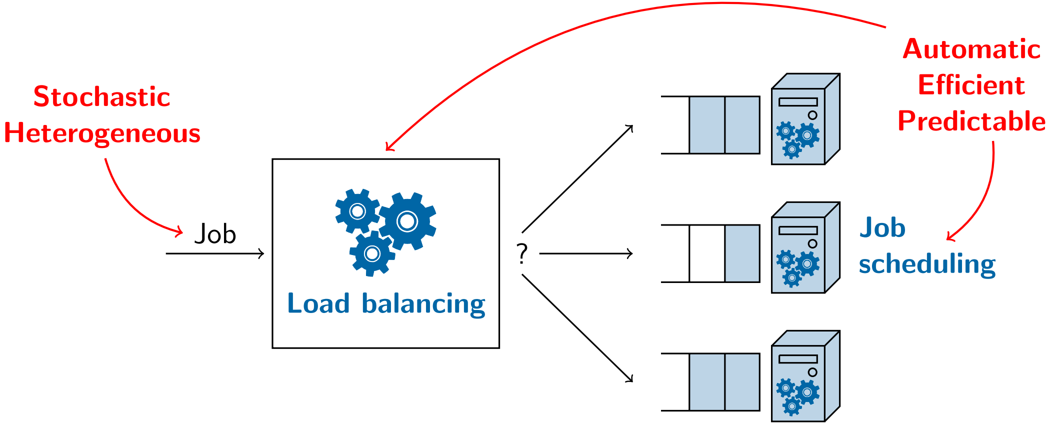
Resource-management algorithms



Resource-management algorithms



Resource-management algorithms



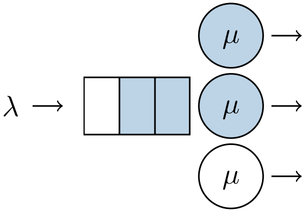
Queueing theory: the early days

Erlang formulas
(Erlang, 1917)

→ Telephone networks

→ Optical networks

Erlang formulas
(Erlang, 1917)

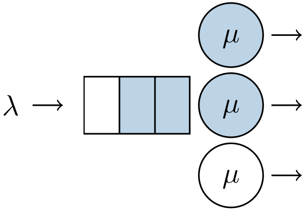


→ Telephone networks

→ Optical networks

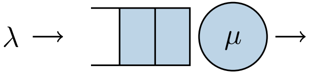
Queueing theory: the early days

Erlang formulas
(Erlang, 1917)



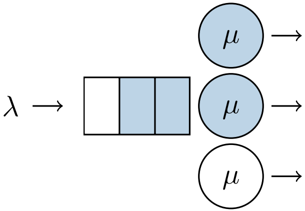
→ Telephone networks
→ Optical networks

Single-server queue
(Kendall, 1951, 1953)

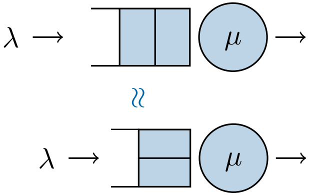


Queueing theory: the early days

Erlang formulas
(Erlang, 1917)



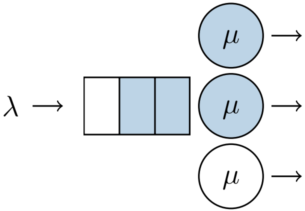
Single-server queue
(Kendall, 1951, 1953)



→ Telephone networks

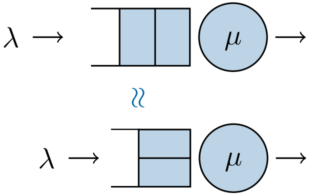
→ Optical networks

Erlang formulas
(Erlang, 1917)



→ Telephone networks
→ Optical networks

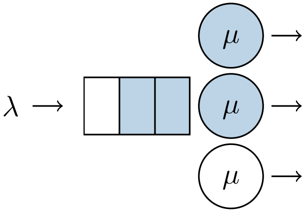
Single-server queue
(Kendall, 1951, 1953)



→ Process schedulers
→ Network schedulers

Queueing theory: the early days

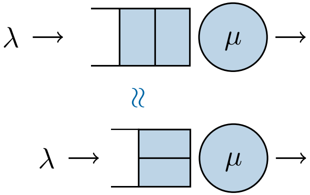
Erlang formulas
(Erlang, 1917)



→ Telephone networks

→ Optical networks

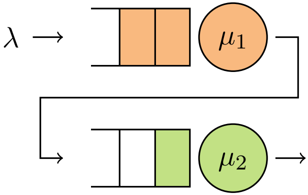
Single-server queue
(Kendall, 1951, 1953)



→ Process schedulers

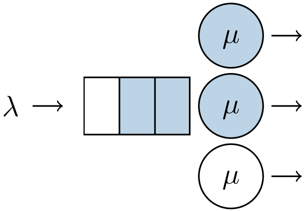
→ Network schedulers

Networks of queues
(Jackson, 1957)



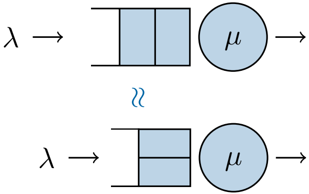
Queueing theory: the early days

Erlang formulas
(Erlang, 1917)



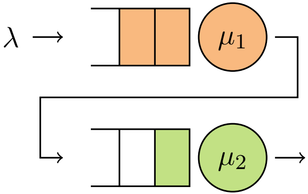
→ Telephone networks
→ Optical networks

Single-server queue
(Kendall, 1951, 1953)



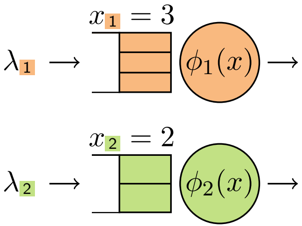
→ Process schedulers
→ Network schedulers

Networks of queues
(Jackson, 1957)



→ Hospital planning
→ Manufacturing

Whittle networks
(Whittle, 1986)

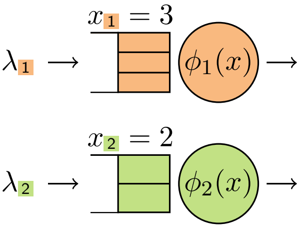


→ Data networks

→ Computer systems

Queueing theory: abstract models

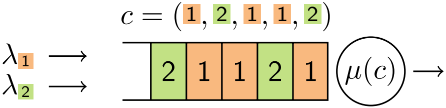
Whittle networks
(Whittle, 1986)



→ Data networks

→ Computer systems

Order-independent queues
(Berezner et al., 1995)

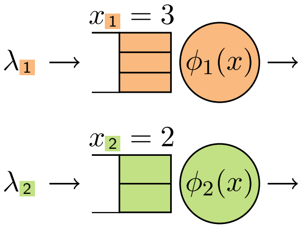


→ Partitioned bus systems

→ Circuit-switched networks

Whittle networks

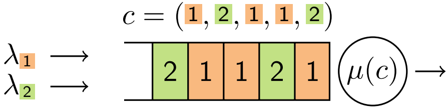
(Whittle, 1986)



$\approx$

Order-independent queues

(Berezner et al., 1995)

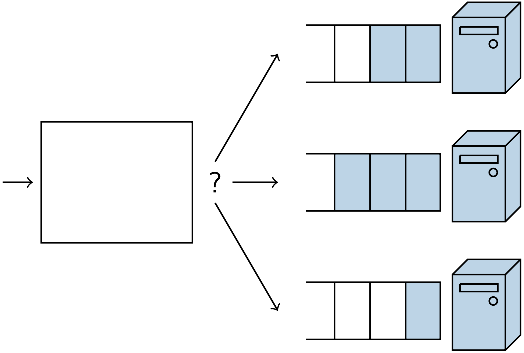


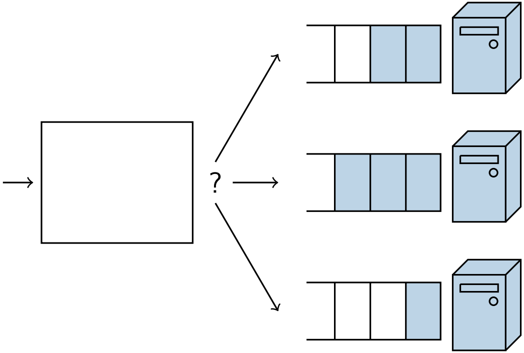
→ Data networks

→ Computer systems

→ Partitioned bus systems

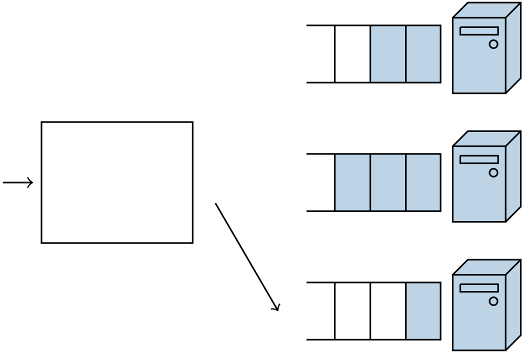
→ Circuit-switched networks





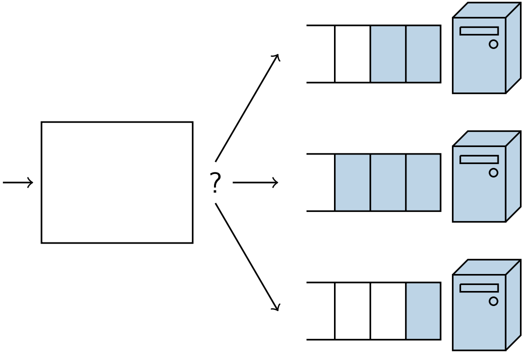
Classical algorithms

- Join-the-shortest-queue



Classical algorithms

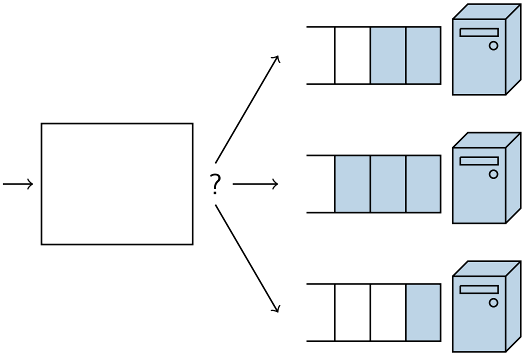
- Join-the-shortest-queue



Classical algorithms

- Join-the-shortest-queue

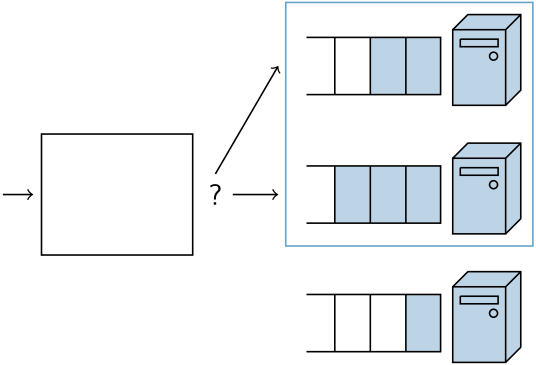
Queueing theory: application to load balancing in computer clusters



Classical algorithms

- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)

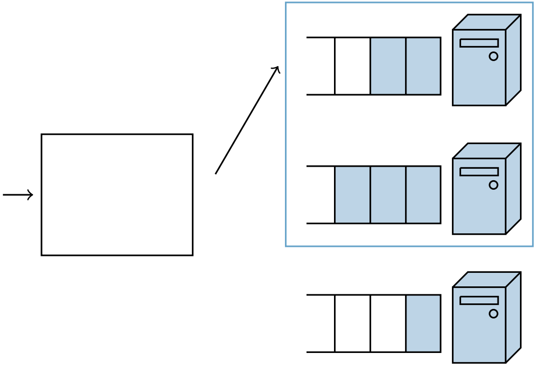
Queueing theory: application to load balancing in computer clusters



Classical algorithms

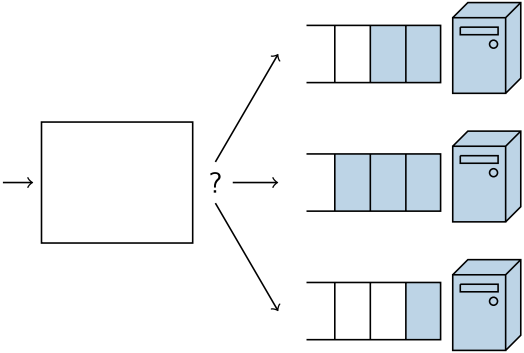
- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)

Queueing theory: application to load balancing in computer clusters



Classical algorithms

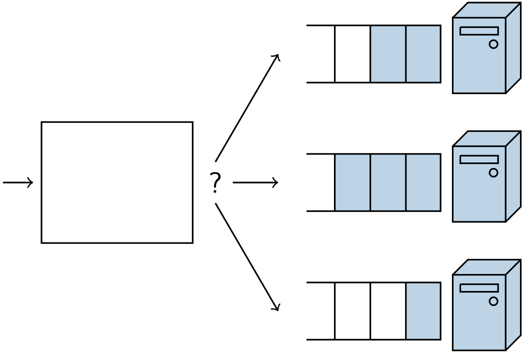
- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)



Classical algorithms

- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)

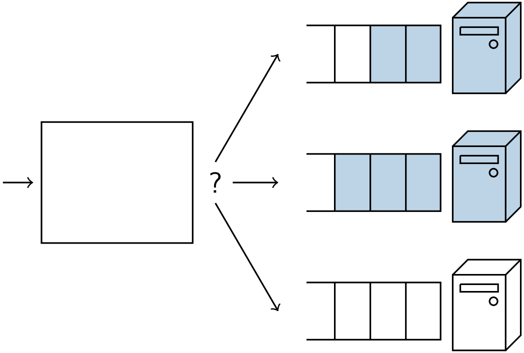
Queueing theory: application to load balancing in computer clusters



Classical algorithms

- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)
- Join-idle-queue (Lu et al., 2011)

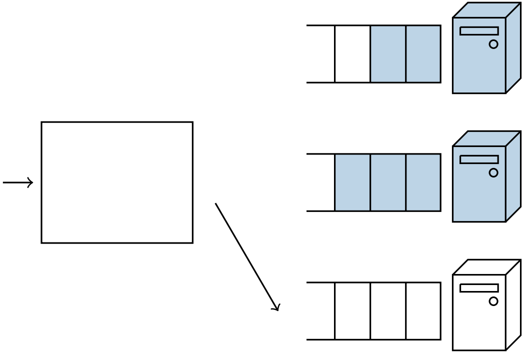
Queueing theory: application to load balancing in computer clusters



Classical algorithms

- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)
- Join-idle-queue (Lu et al., 2011)

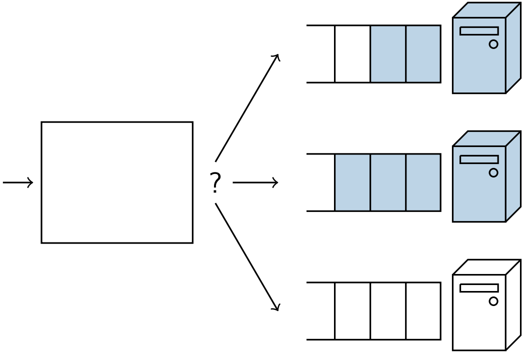
Queueing theory: application to load balancing in computer clusters



Classical algorithms

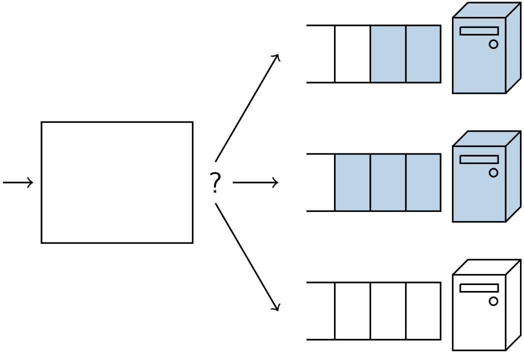
- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)
- Join-idle-queue (Lu et al., 2011)

Queueing theory: application to load balancing in computer clusters



Classical algorithms

- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)
- Join-idle-queue (Lu et al., 2011)



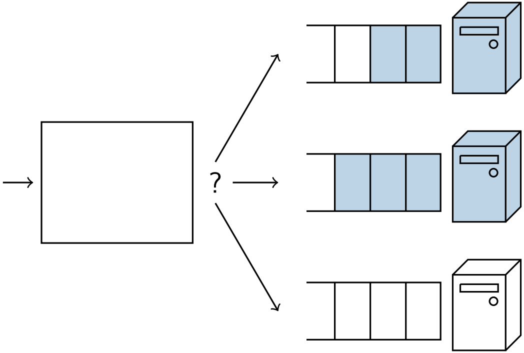
Classical algorithms

- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)
- Join-idle-queue (Lu et al., 2011)

Exact analyses with two computers and approximations otherwise

(Gupta et al., 2007)

Queueing theory: application to load balancing in computer clusters



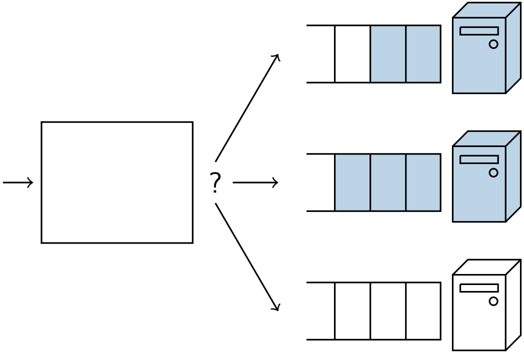
Classical algorithms

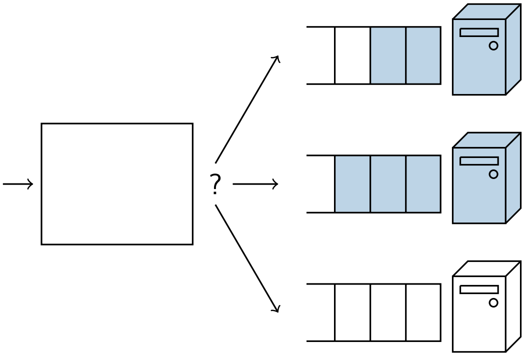
- Join-the-shortest-queue
- Power-of-two-choices (Mitzenmacher, 1996)
- Join-idle-queue (Lu et al., 2011)

Exact analyses with two computers and approximations otherwise
(Gupta et al., 2007)

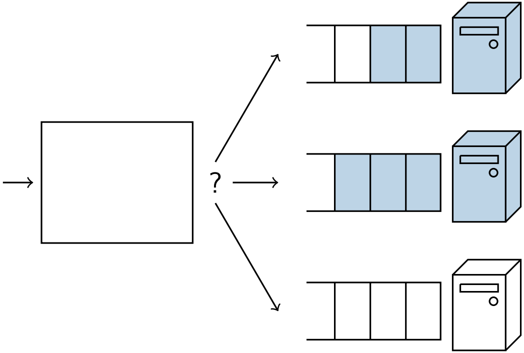
Asymptotic scaling regimes
(van der Boor et al., 2018)

Queueing theory: application to load balancing in computer clusters



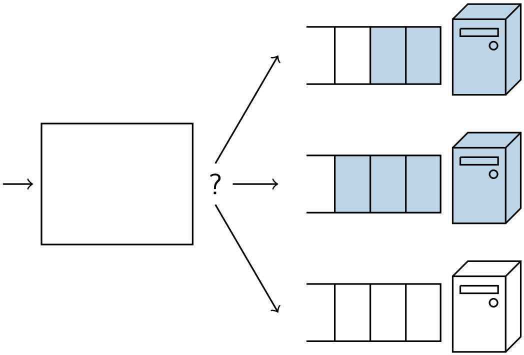


Insensitive algorithms



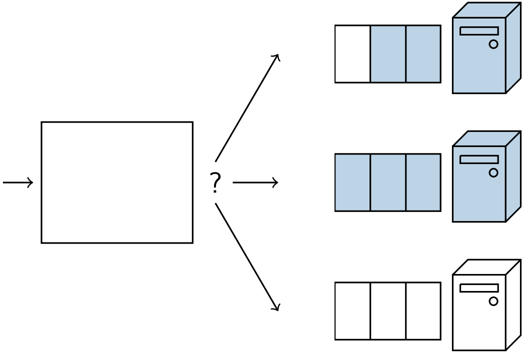
Insensitive algorithms

- Static random



Insensitive algorithms

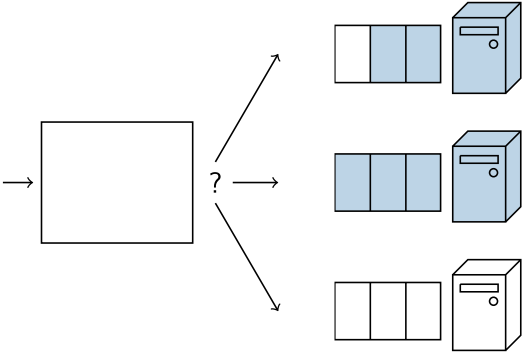
- Static random
- Dynamic random (Bonald et al., 2004)



Insensitive algorithms

- Static random
- Dynamic random (Bonald et al., 2004)

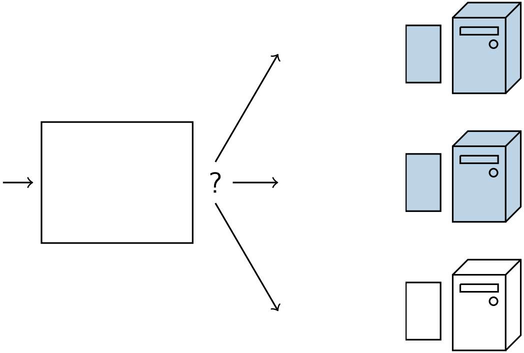
Queueing theory: application to load balancing in computer clusters



Insensitive algorithms

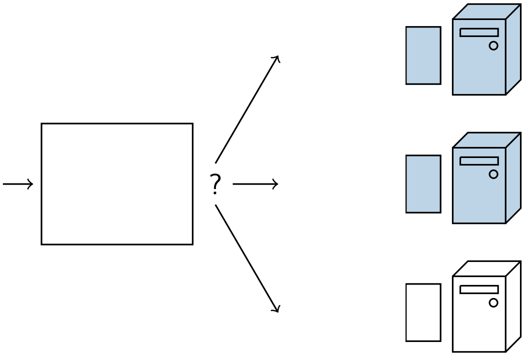
- Static random
- Dynamic random (Bonald et al., 2004)
- Assign-to-the-longest-idle-server (Adan and Weiss, 2012)

Queueing theory: application to load balancing in computer clusters



Insensitive algorithms

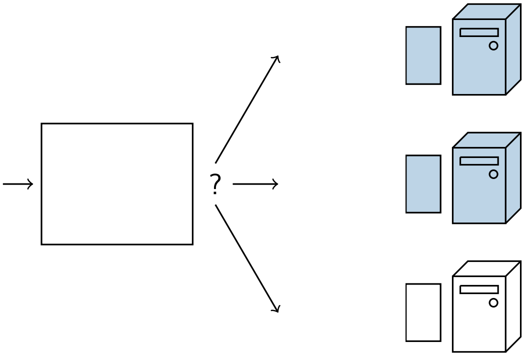
- Static random
- Dynamic random (Bonald et al., 2004)
- Assign-to-the-longest-idle-server (Adan and Weiss, 2012)



Insensitive algorithms

- Static random
- Dynamic random (Bonald et al., 2004)
- Assign-to-the-longest-idle-server (Adan and Weiss, 2012)

Product-form queueing networks



Insensitive algorithms

- Static random
- Dynamic random (Bonald et al., 2004)
- Assign-to-the-longest-idle-server (Adan and Weiss, 2012)

Product-form queueing networks

Asymptotic scaling regimes

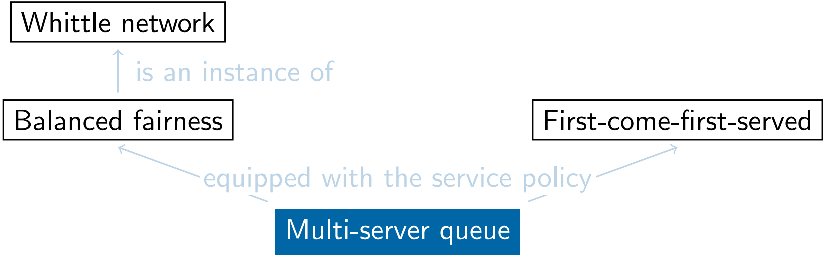
(Jonckheere and Prabhu, 2016)

Roadmap of the manuscrit

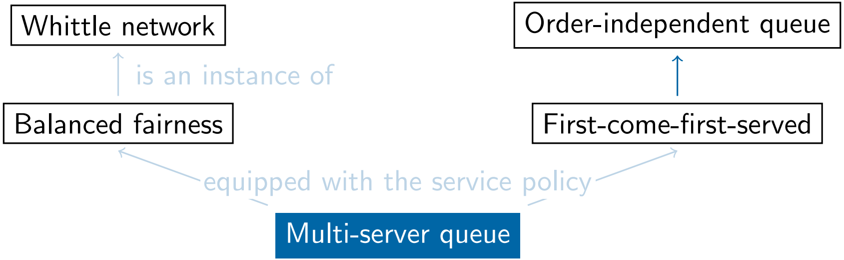
Multi-server queue

Roadmap of the manuscrit

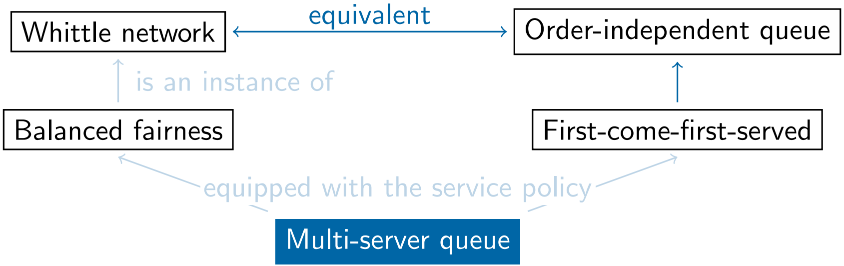
Roadmap of the manuscrit



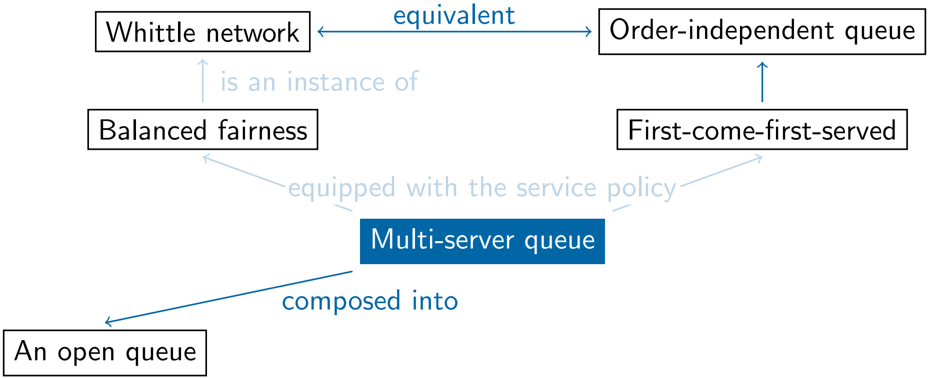
Roadmap of the manuscrit



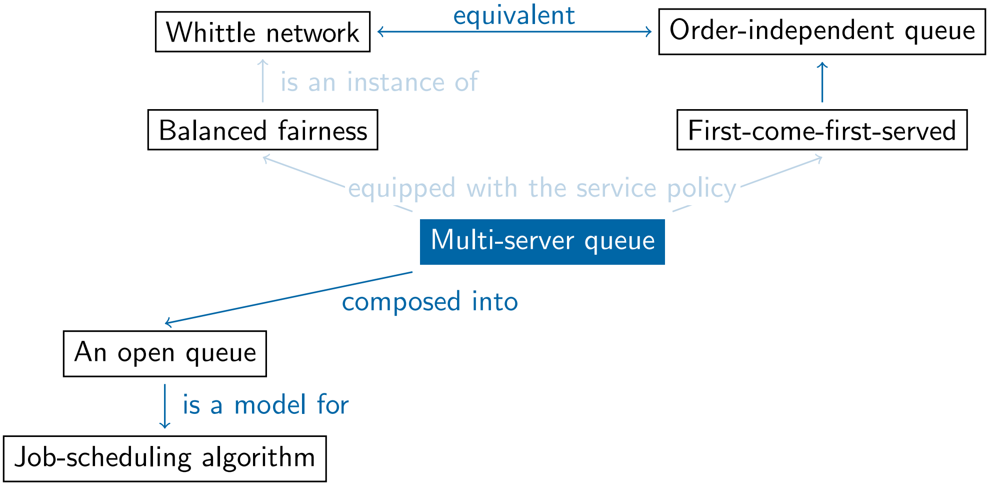
Roadmap of the manuscrit



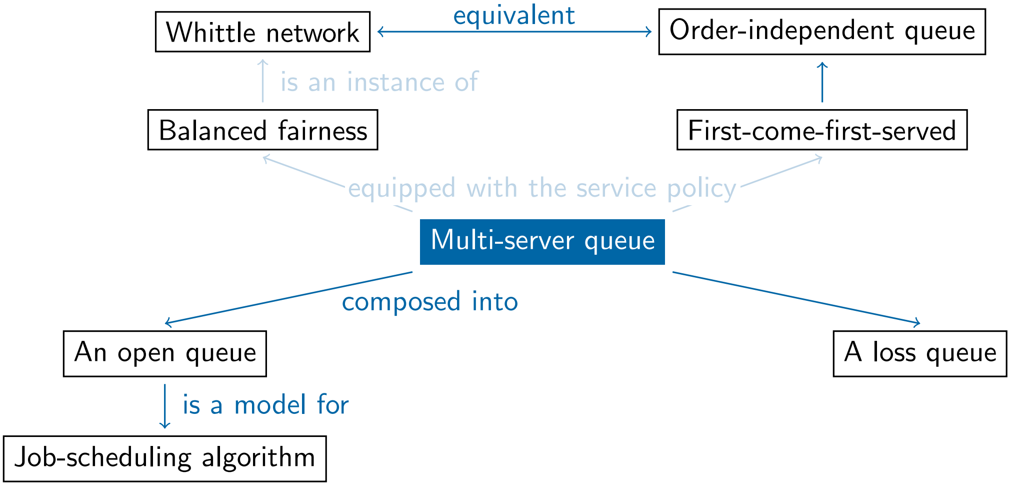
Roadmap of the manuscrit



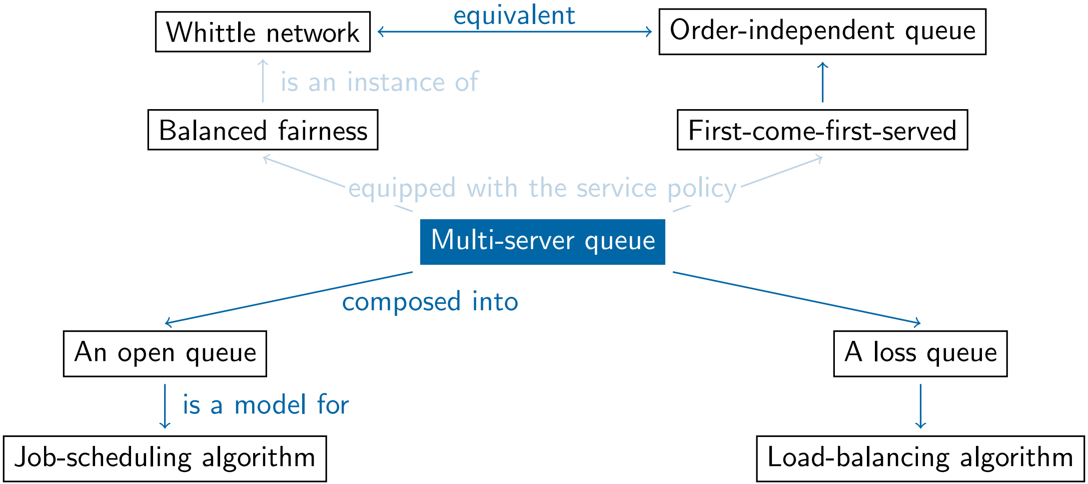
Roadmap of the manuscrit



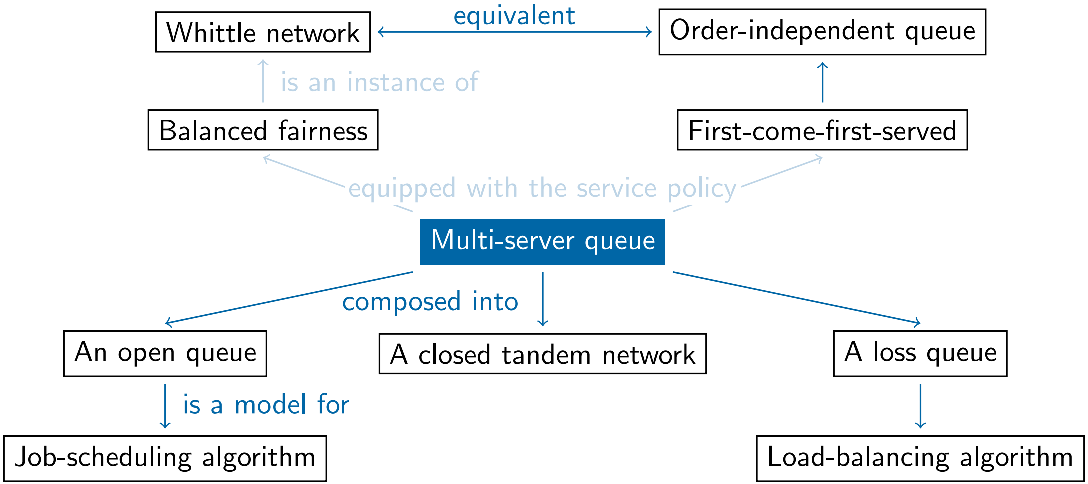
Roadmap of the manuscrit



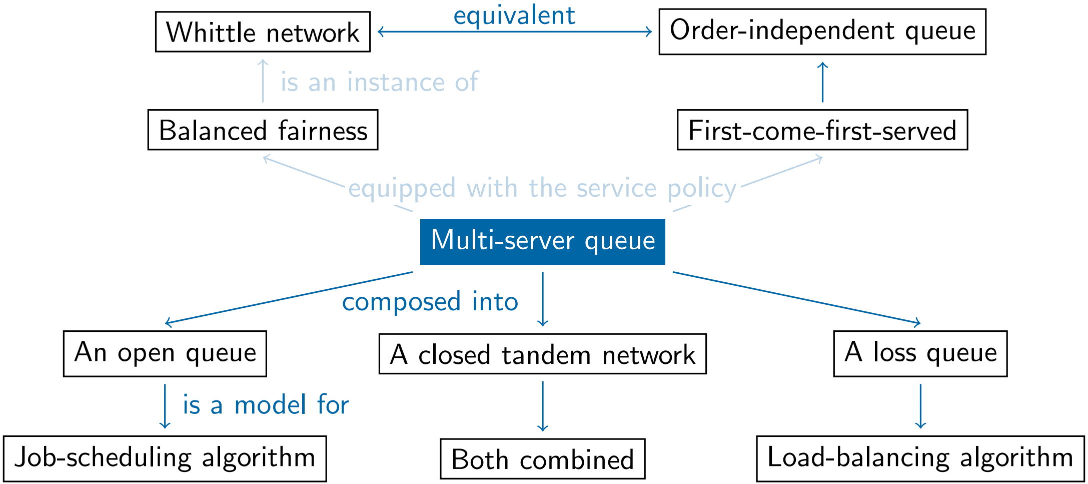
Roadmap of the manuscrit



Roadmap of the manuscrit



Roadmap of the manuscrit



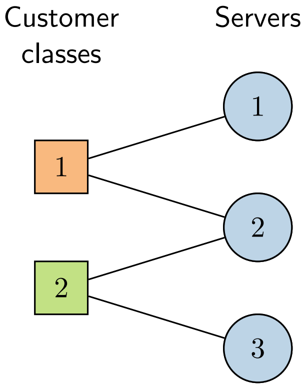
Outline

1. Equivalence of balanced fairness and first-come-first-served
2. Performance analysis of the open queue
3. Applications in algorithm design

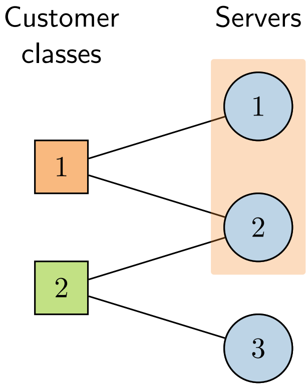
Outline

1. Equivalence of balanced fairness and first-come-first-served
2. Performance analysis of the open queue
3. Applications in algorithm design

Compatibility graph

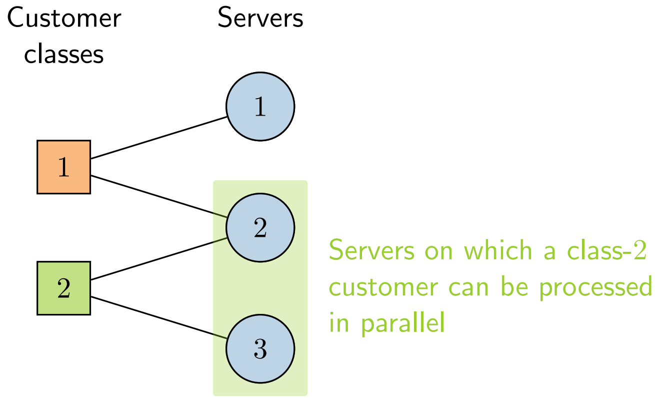


Compatibility graph

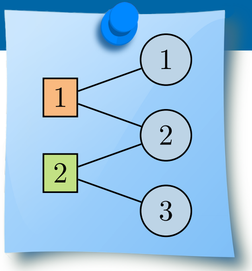


Servers on which a class-1 customer can be processed in parallel

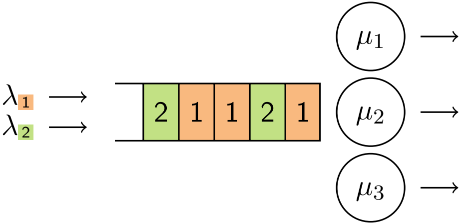
Compatibility graph

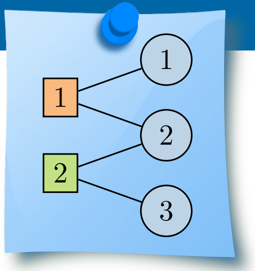


The multi-server queue

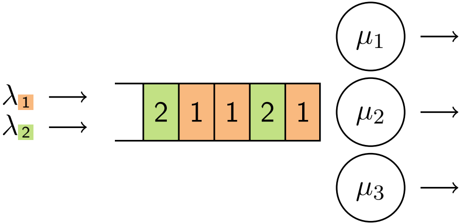


The multi-server queue

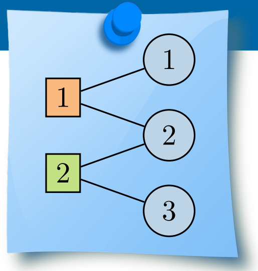




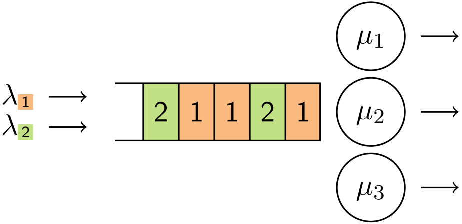
The multi-server queue

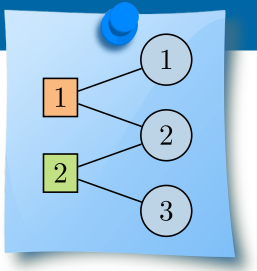


- **Markovian assumptions**
  - → Class-i customers arrive according to a Poisson process with rate λ_i
  - → Server s has capacity μ_s
  - → Service requirements are independent and exponentially distributed with unit mean



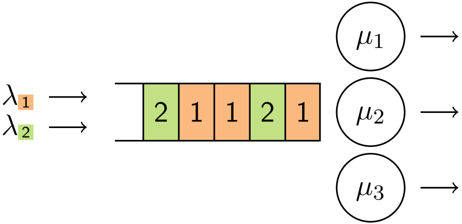
The multi-server queue

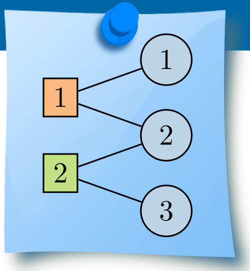




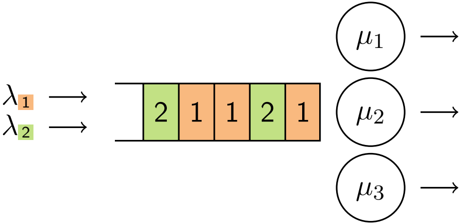
- **Markovian assumptions**
  - → Class-i customers arrive according to a Poisson process with rate λ_i
  - → Server s has capacity μ_s
  - → Service requirements are independent and exponentially distributed with unit mean
- **Queue state**
  - → Microstate $c = (1, 2, 1, 1, 2)$
  - → Macrostate $x = \begin{pmatrix} 3 & 1 \\ 2 & 2 \end{pmatrix}$

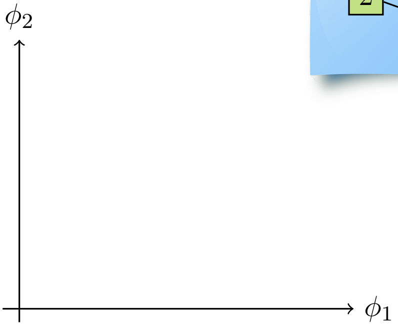
Capacity region

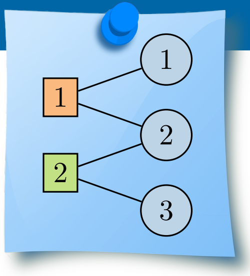




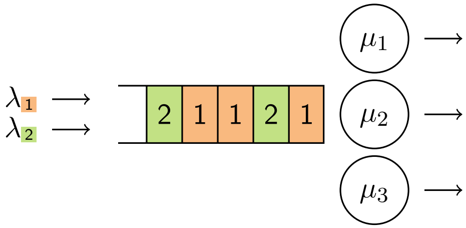
Capacity region

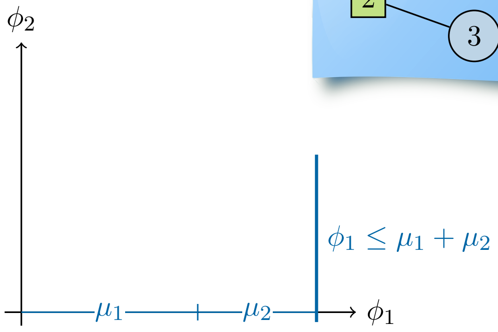


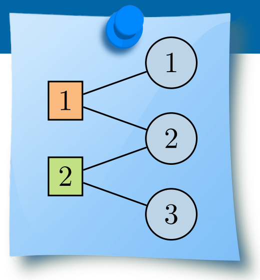




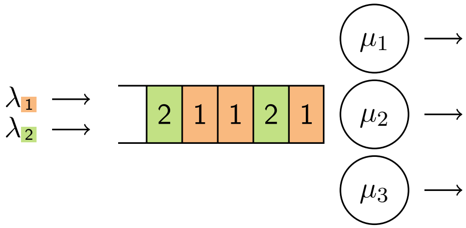
Capacity region

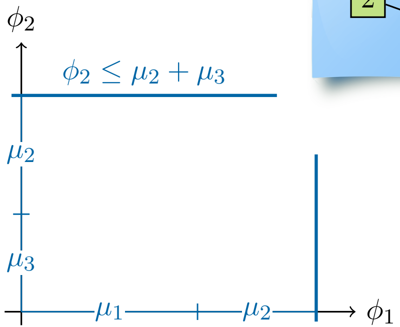


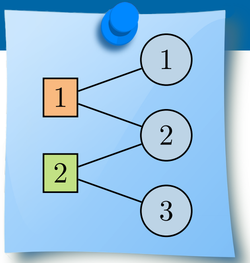




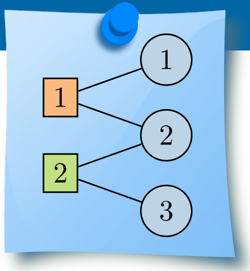
Capacity region

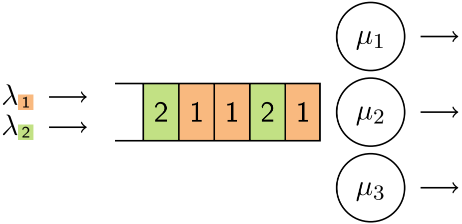


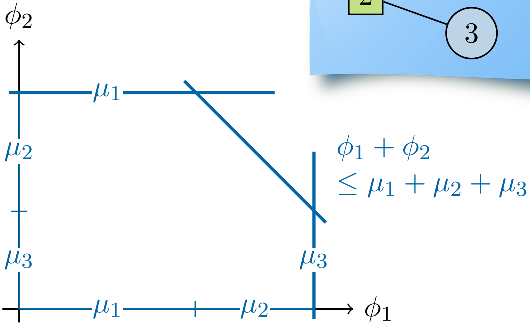




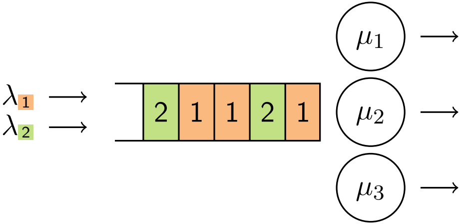
Capacity region

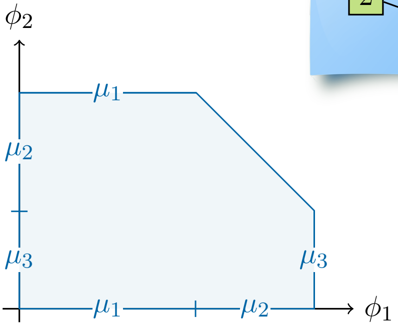


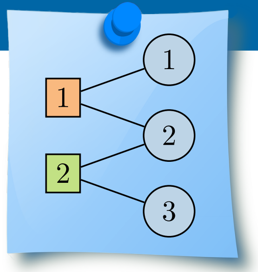




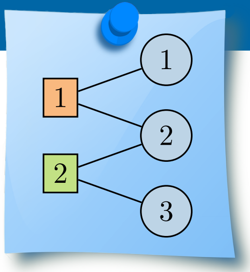
Capacity region

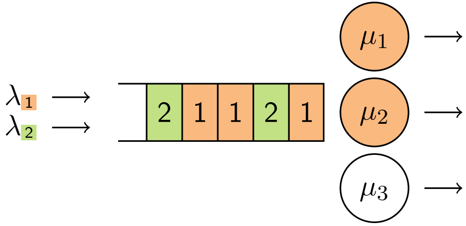


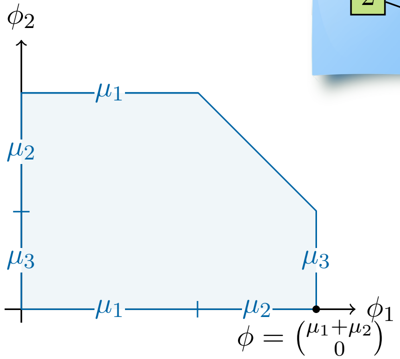




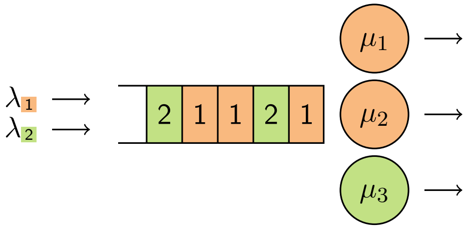
Capacity region

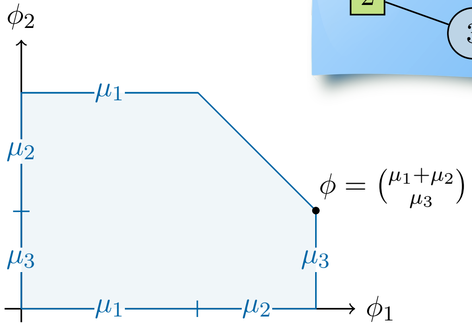


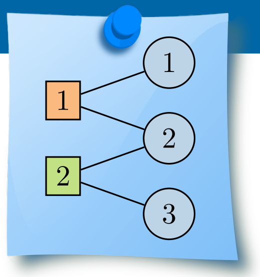




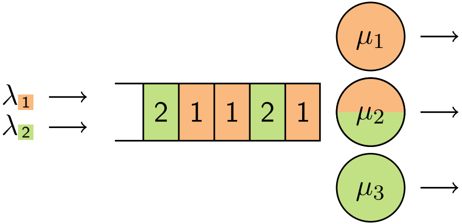
Capacity region

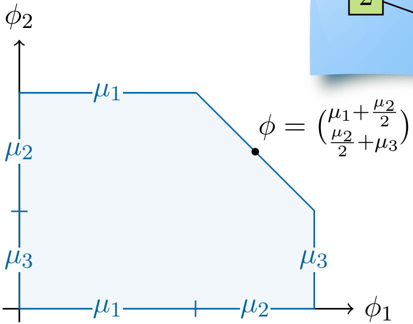


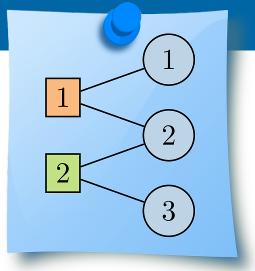




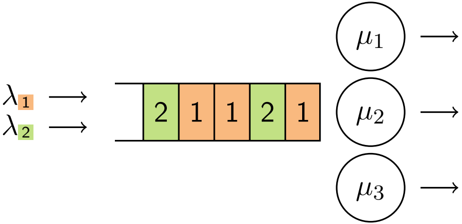
Capacity region

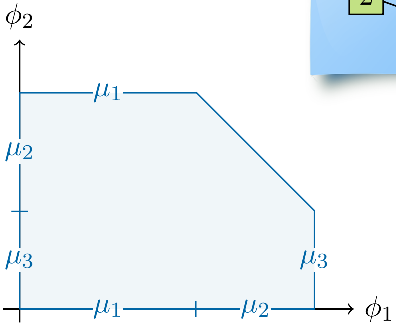


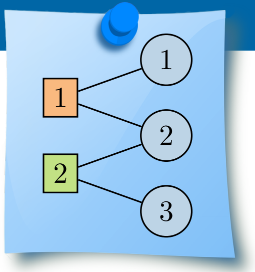




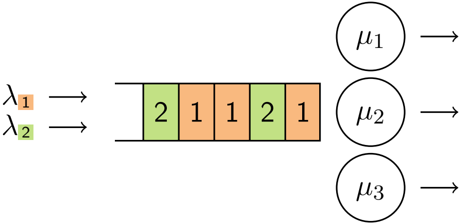
Capacity region

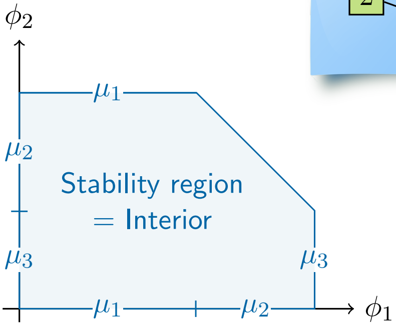


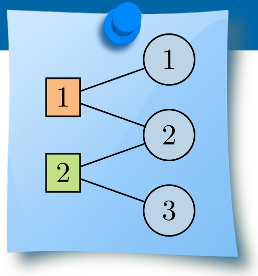




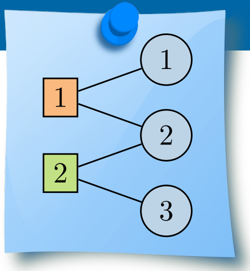
Capacity region





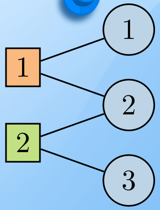


First-come-first-served policy



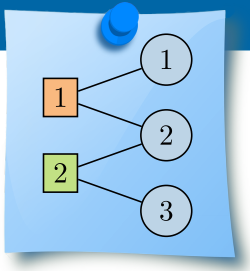
- Time-sharing policy considered in (Gardner et al., 2016)

First-come-first-served policy

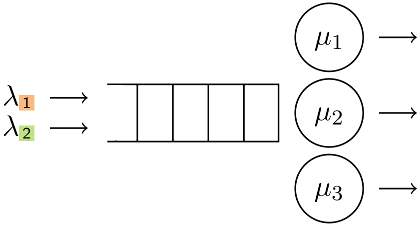


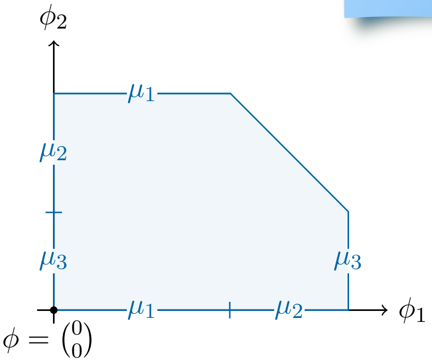
- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers

First-come-first-served policy



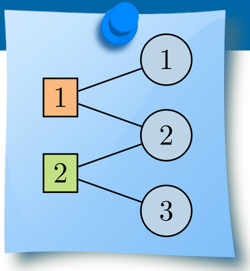
- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers

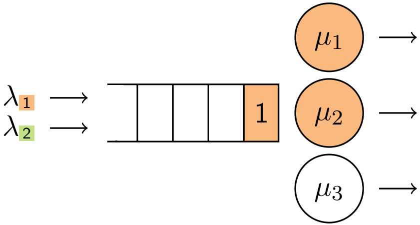


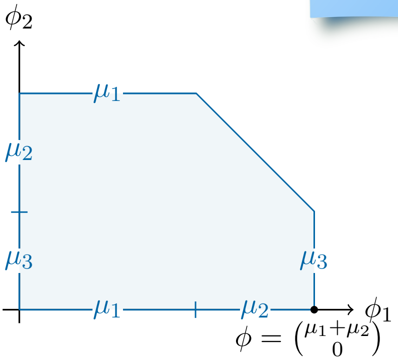


First-come-first-served policy

- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers

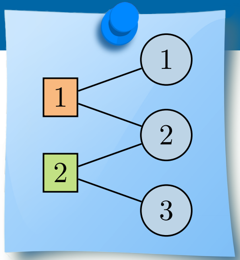


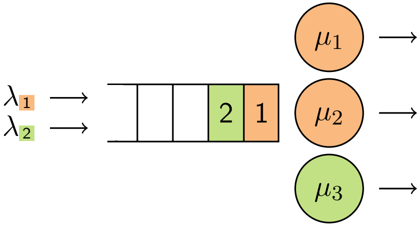


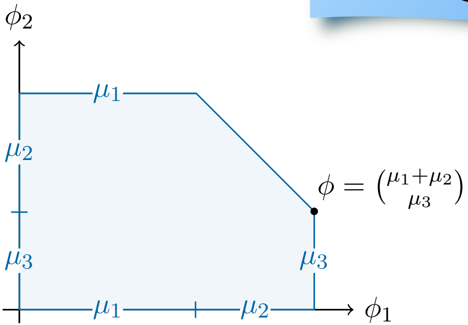


First-come-first-served policy

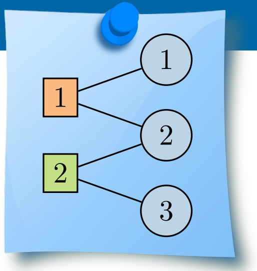
- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers



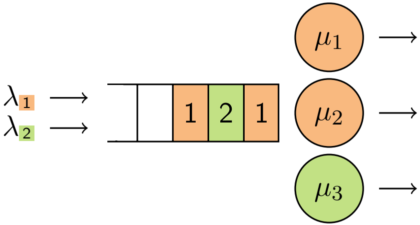


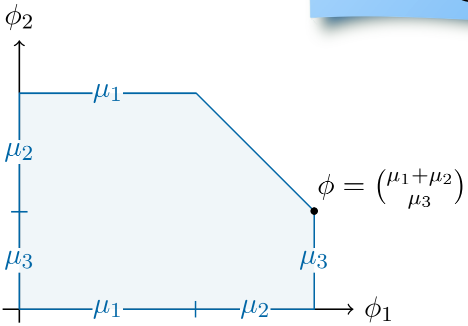


First-come-first-served policy

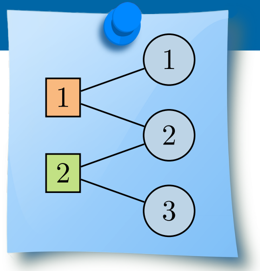


- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers

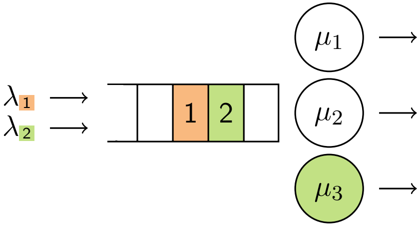


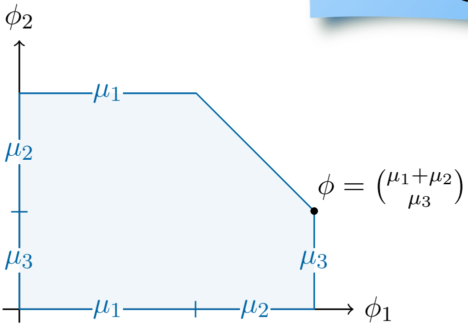


First-come-first-served policy



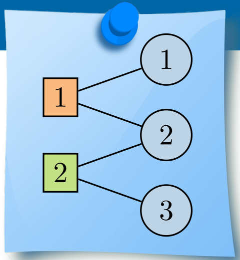
- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers

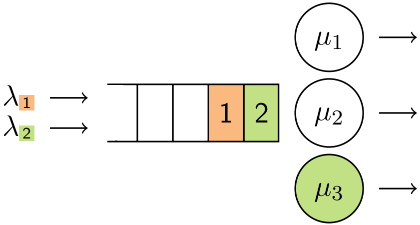


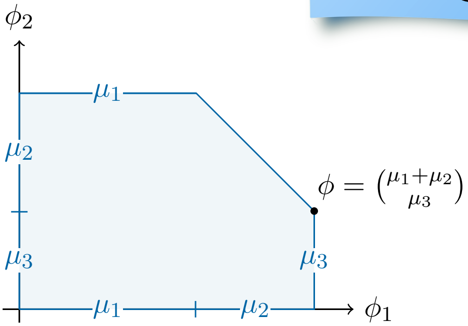


First-come-first-served policy

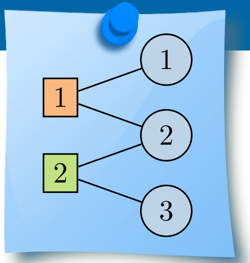
- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers



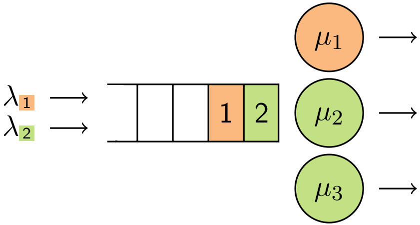


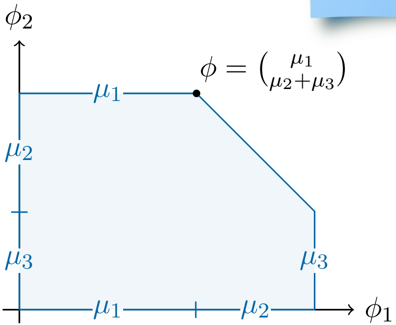


First-come-first-served policy

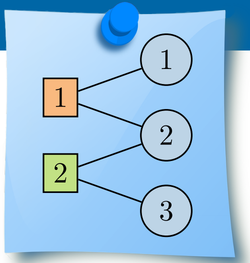


- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers

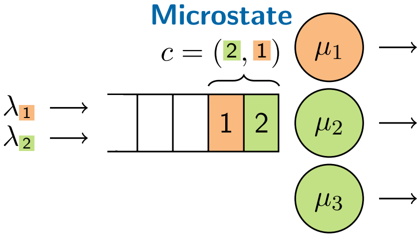


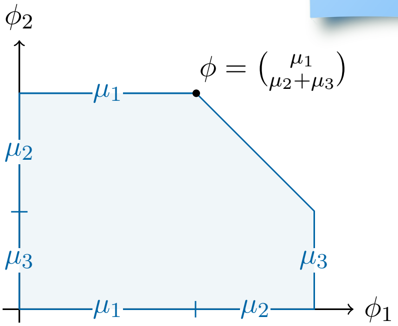


First-come-first-served policy

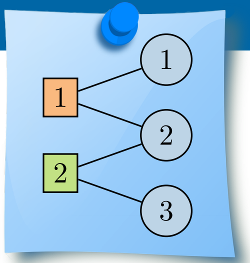


- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers

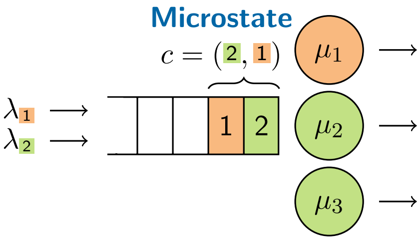


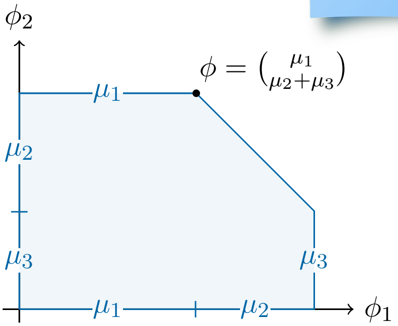


First-come-first-served policy

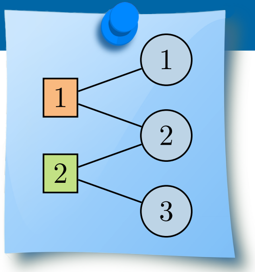


- Time-sharing policy considered in (Gardner et al., 2016)
- Servers are greedily assigned to customers
- The queue is **order-independent**



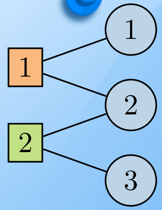


Balanced fairness



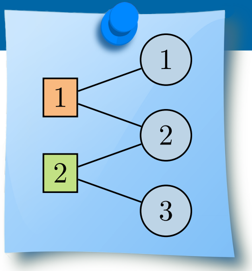
- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)

Balanced fairness

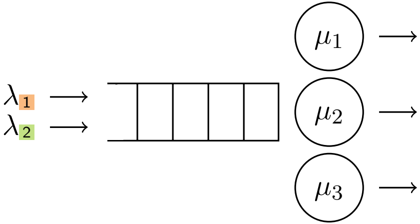


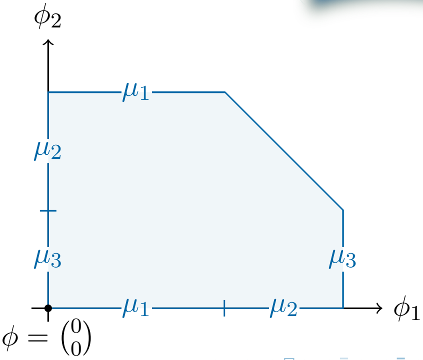
- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

Balanced fairness



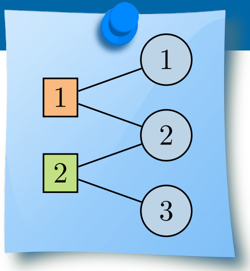
- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

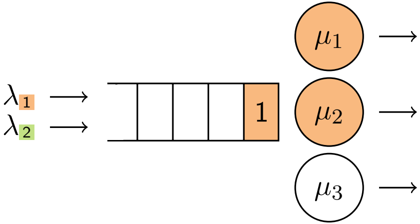


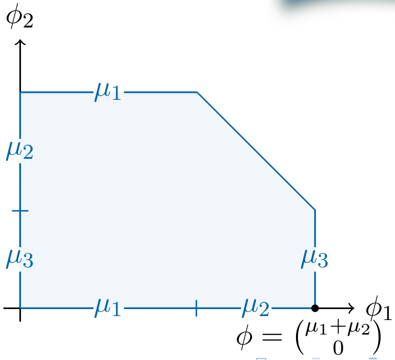


Balanced fairness

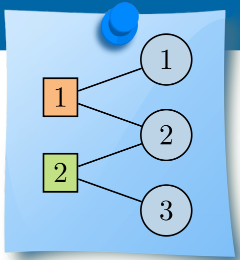
- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order



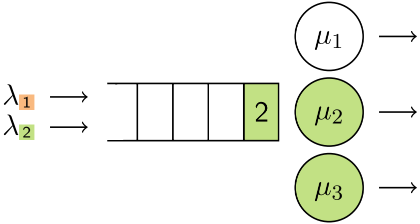


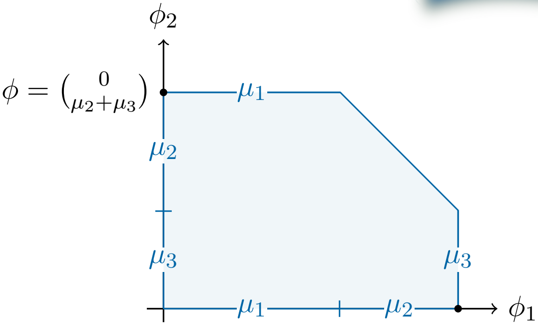


Balanced fairness

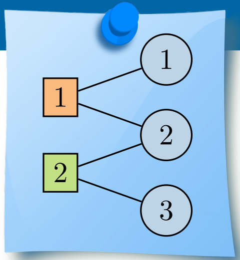


- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

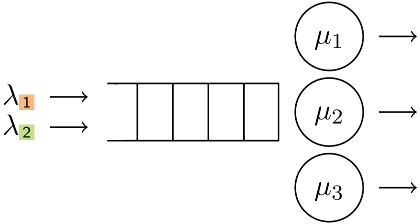


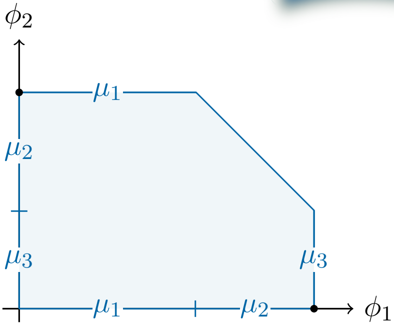


Balanced fairness

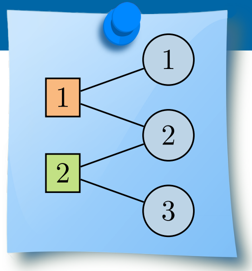


- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

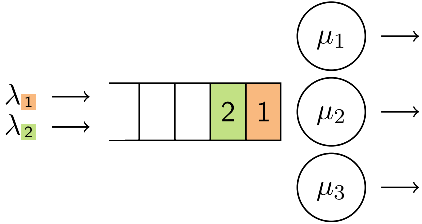


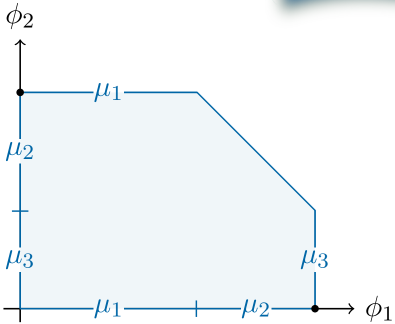


Balanced fairness



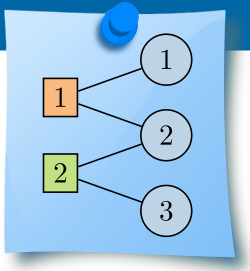
- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

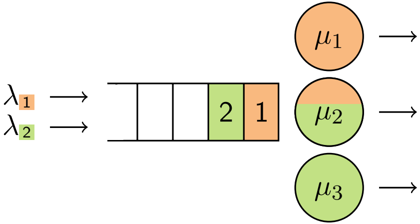


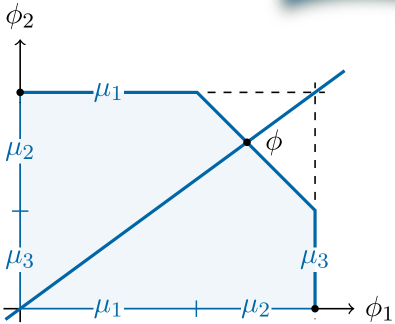


Balanced fairness

- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

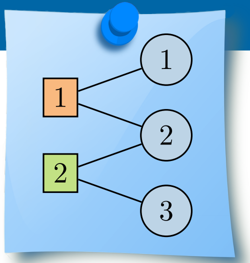


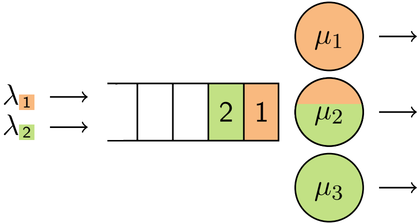


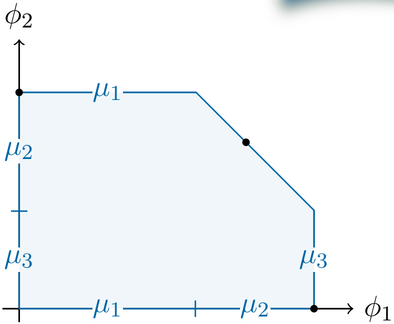


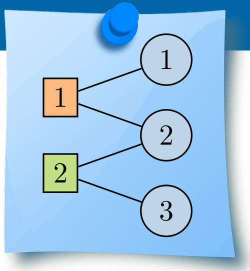
Balanced fairness

- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

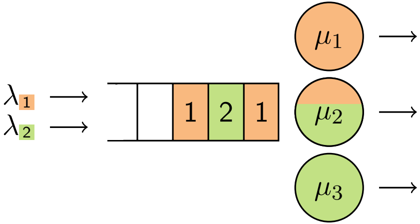


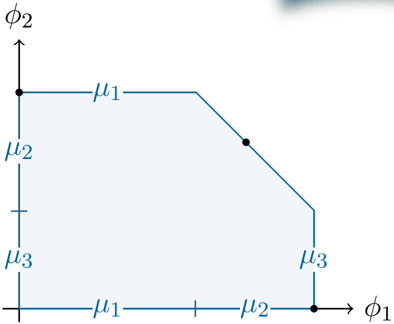






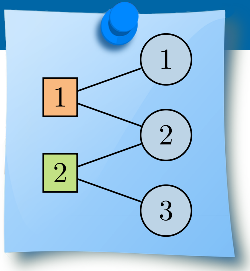
- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

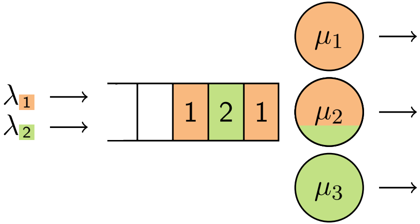


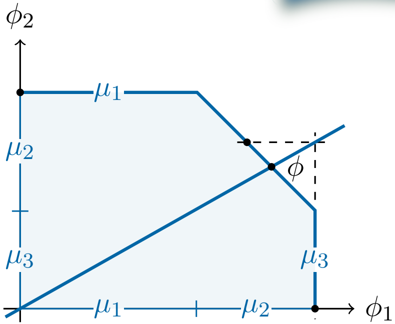


Balanced fairness

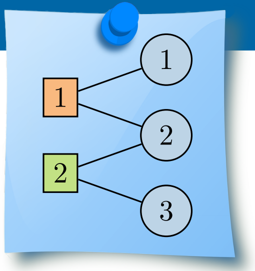
- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order



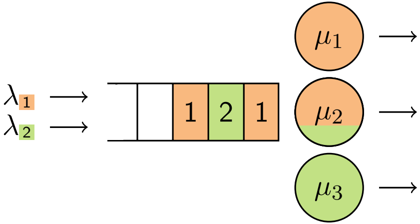


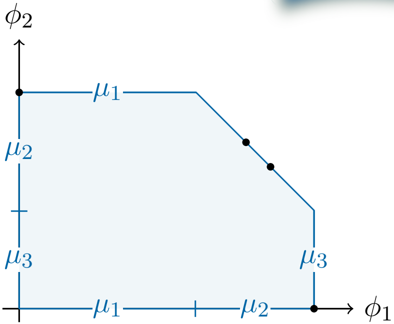


Balanced fairness

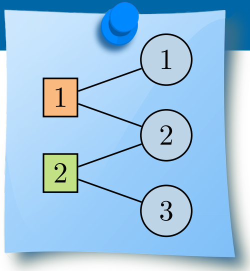


- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order



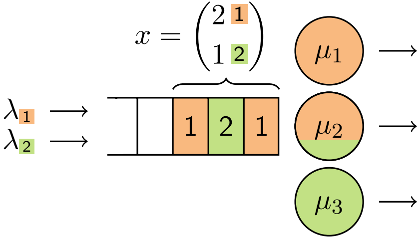


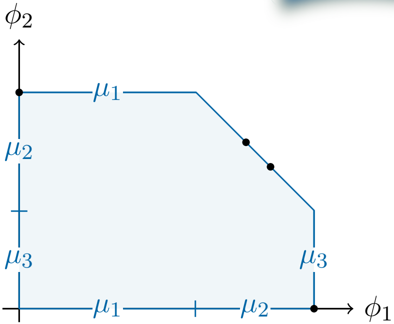
Balanced fairness

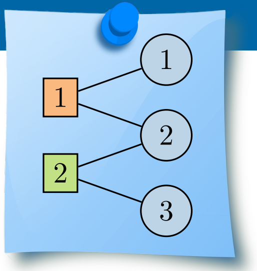


- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order

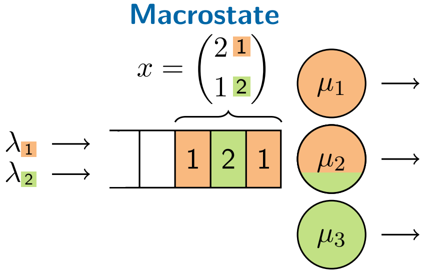
Macrostate

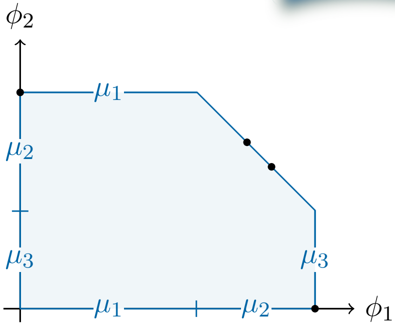




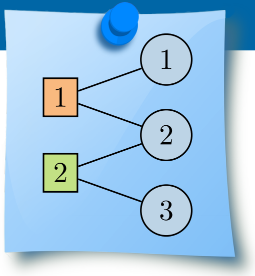


- Resource-sharing policy introduced in (Bonald and Proutière, 2003) and applied to the multi-server queue in (Shah and de Veciana, 2015)
- Independent of the customer arrival order
- Dynamics described by a **Whittle network**





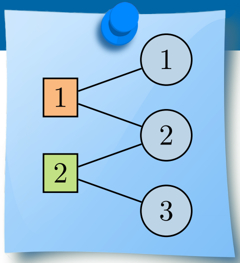
Equivalence



- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

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

Equivalence

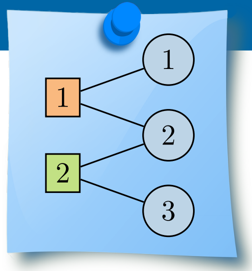


- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

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

Stationary probability of being in macrostate x under balanced fairness

Equivalence



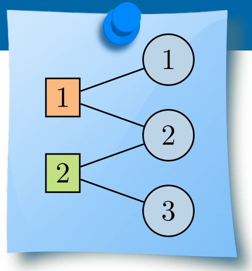
- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

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

Stationary probability of being in macrostate x under balanced fairness

Microstates c that correspond to macrostate x

Equivalence



- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

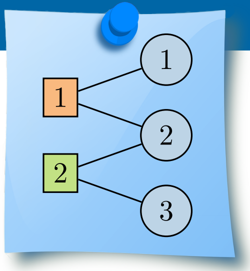
$$\pi(x) = \sum_{c:|c|=x} \pi(c)$$

Stationary probability of being in macrostate x under balanced fairness

Stationary probability of being in microstate c under first-come-first-served

Microstates c that correspond to macrostate x

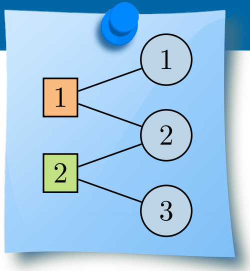
Equivalence



- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

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

Equivalence



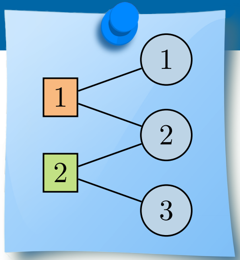
- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

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

- The service rate of class i under balanced fairness is equal to the **average service rate** of class i under first-come-first-served

$$\phi_i(x) = \sum_{c : |c| = x} \phi_i(c) \frac{\pi(c)}{\pi(x)}$$

Equivalence

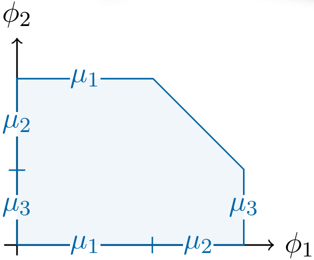


- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

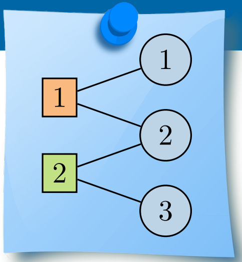
$$\pi(x) = \sum_{c:|c|=x} \pi(c)$$

- The service rate of class i under balanced fairness is equal to the **average service rate** of class i under first-come-first-served

$$\phi_i(x) = \sum_{c:|c|=x} \phi_i(c) \frac{\pi(c)}{\pi(x)}$$



Equivalence

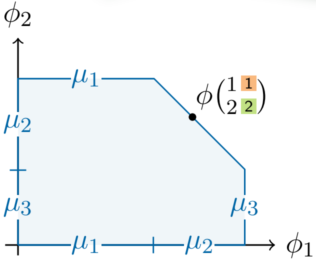


- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

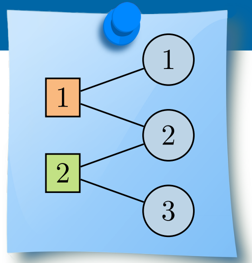
$$\pi(x) = \sum_{c:|c|=x} \pi(c)$$

- The service rate of class i under balanced fairness is equal to the **average service rate** of class i under first-come-first-served

$$\boxed{\phi_i(x)} = \sum_{c:|c|=x} \phi_i(c) \frac{\pi(c)}{\pi(x)}$$



Equivalence

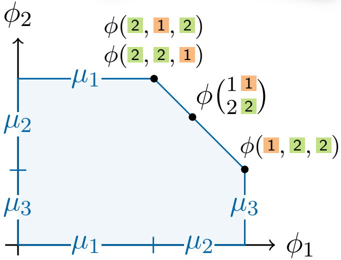


- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

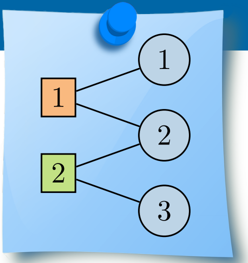
$$\pi(x) = \sum_{c:|c|=x} \pi(c)$$

- The service rate of class i under balanced fairness is equal to the **average service rate** of class i under first-come-first-served

$$\phi_i(x) = \sum_{c:|c|=x} \phi_i(c) \frac{\pi(c)}{\pi(x)}$$



Equivalence

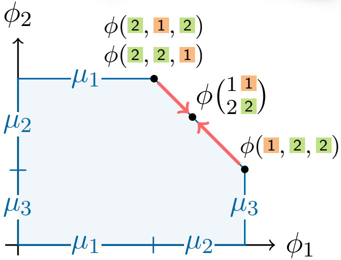


- Balanced fairness and first-come-first-served are equivalent with respect to the **stationary distribution** of the macrostate

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

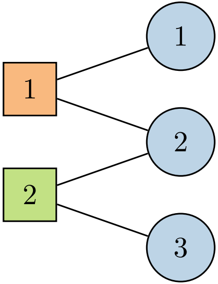
- The service rate of class i under balanced fairness is equal to the **average service rate** of class i under first-come-first-served

$$\phi_i(x) = \sum_{c:|c|=x} \phi_i(c)\frac{\pi(c)}{\pi(x)}$$

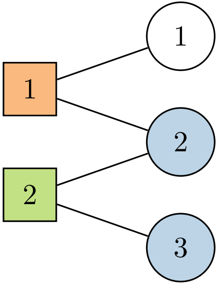


Outline

Server elimination

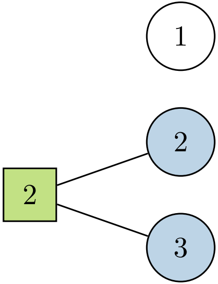


Conditionally on server s being idle, the stationary queue behaves like the restricted queue without traffic generated by the classes compatible with server s.



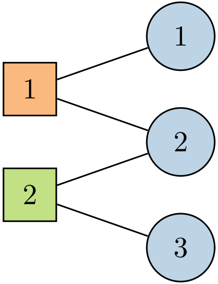
Conditionally on server s being idle, the stationary queue behaves like the restricted queue without traffic generated by the classes compatible with server s.

Server elimination



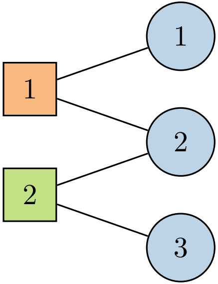
Conditionally on server s being idle, the stationary queue behaves like the restricted queue without traffic generated by the classes compatible with server s.

Server elimination



Conditionally on server s being idle, the stationary queue behaves like the restricted queue without traffic generated by the classes compatible with server s.

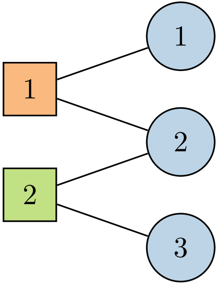
Server elimination



Conditionally on server s being idle, the stationary queue behaves like the restricted queue without traffic generated by the classes compatible with server s.

$$\psi_{|-s} = \mathbb{P}\left(\begin{array}{c|c} \text{the queue} & \text{server } s \\ \text{is empty} & \text{is idle} \end{array}\right)$$

Server elimination

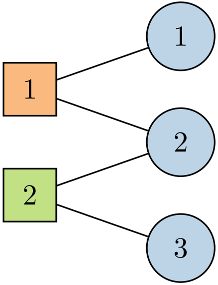


Conditionally on server s being idle, the stationary queue behaves like the restricted queue without traffic generated by the classes compatible with server s.

$$\psi_{|-s} = \mathbb{P}\left(\begin{matrix}\text{the queue}\\ \text{is empty}\end{matrix} \,\middle|\, \begin{matrix}\text{server } s\\ \text{is idle}\end{matrix}\right)$$

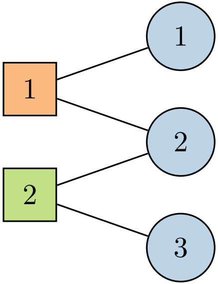
$$= \mathbb{P}\left(\begin{matrix}\text{the restricted queue, without server } s\\ \text{and its compatible classes, is empty}\end{matrix}\right)$$

Performance metrics



- Probability that the queue is empty

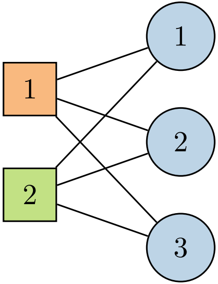
$$\psi = (1-\rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$



- Probability that the queue is empty

$$\psi = \boxed{(1-\rho)} \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

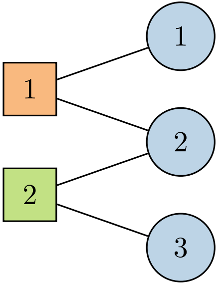
Complete pooling



- Probability that the queue is empty

$$\psi = (1 - \rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

Complete pooling

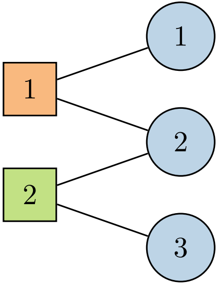


- Probability that the queue is empty

$$\psi = \boxed{(1-\rho)} \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

Complete pooling

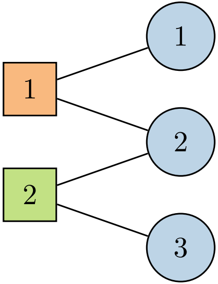
Performance metrics



- Probability that the queue is empty

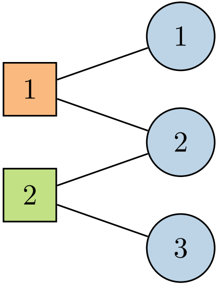
$$\psi = \underbrace{(1-\rho)}_{\text{Complete pooling}} \times \underbrace{\frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}}}_{\text{Overhead due to incomplete pooling}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

Performance metrics



- Probability that the queue is empty

$$\psi = (1-\rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

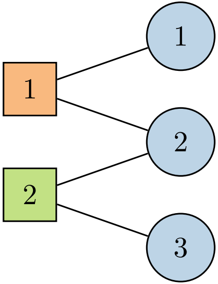


- Probability that the queue is empty

$$\psi = (1-\rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

- Expected number of customers

$$L = \frac{\rho}{1-\rho} + \frac{1}{1-\rho} \frac{\sum_s \mu_s \frac{\psi}{\psi_{|-s}} L_{|-s}}{\sum_s \mu_s}$$



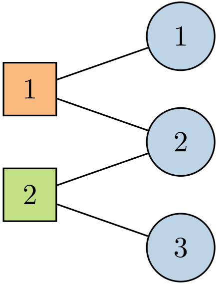
- Probability that the queue is empty

$$\psi = (1-\rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

- Expected number of customers

$$L = \boxed{\frac{\rho}{1-\rho}} + \frac{1}{1-\rho} \frac{\sum_s \mu_s \frac{\psi}{\psi_{|-s}} L_{|-s}}{\sum_s \mu_s}$$

Complete pooling



- Probability that the queue is empty

$$\psi = (1-\rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

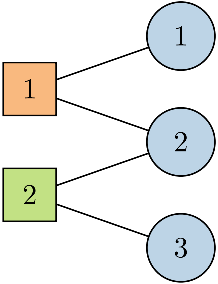
- Expected number of customers

$$L = \frac{\rho}{1-\rho} + \frac{1}{1-\rho} \frac{\sum_s \mu_s \frac{\psi}{\psi_{|-s}} L_{|-s}}{\sum_s \mu_s}$$

Complete pooling

Overhead due to incomplete pooling

Performance metrics



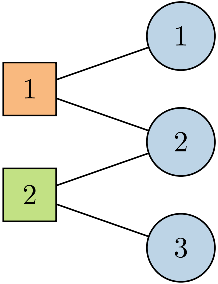
- Probability that the queue is empty

$$\psi = (1-\rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

- Expected number of customers

$$L = \frac{\rho}{1-\rho} + \frac{1}{1-\rho} \frac{\sum_s \mu_s \frac{\psi}{\psi_{|-s}} L_{|-s}}{\sum_s \mu_s}$$

Performance metrics



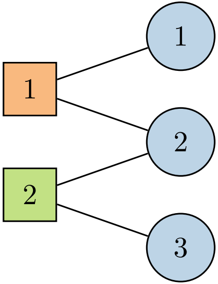
- Probability that the queue is empty

$$\psi = (1 - \rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

- Expected number of customers

$$L = \frac{\rho}{1 - \rho} + \frac{1}{1 - \rho} \frac{\sum_s \mu_s \frac{\psi}{\psi_{|-s}} L_{|-s}}{\sum_s \mu_s}$$

- Time complexity exponential in the number of servers



- Probability that the queue is empty

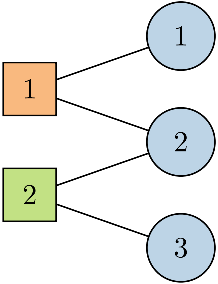
$$\psi = (1-\rho) \times \frac{\sum_s \mu_s}{\sum_s \frac{\mu_s}{\psi_{|-s}}} \quad \text{with} \quad \rho = \frac{\sum_i \lambda_i}{\sum_s \mu_s}$$

- Expected number of customers

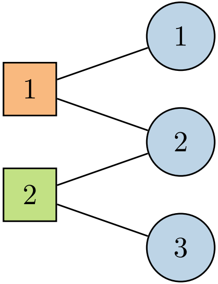
$$L = \frac{\rho}{1-\rho} + \frac{1}{1-\rho} \frac{\sum_s \mu_s \frac{\psi}{\psi_{|-s}} L_{|-s}}{\sum_s \mu_s}$$

- Time complexity exponential in the number of servers
- Polynomial in interesting cases

Toy example



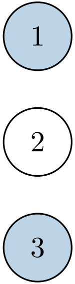
$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = \\ \psi_{|-2} = \\ \psi_{|-3} = \end{cases}$$

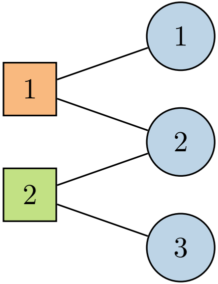
Toy example



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = \\ \psi_{|-2} = \\ \psi_{|-3} = \end{cases}$$

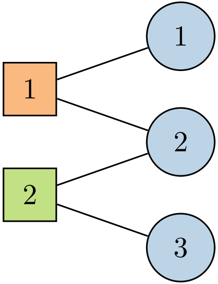
Toy example



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = \\ \psi_{|-2} = \\ \psi_{|-3} = \end{cases}$$

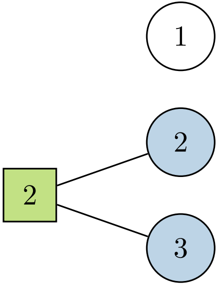
Toy example



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = \\ \psi_{|-2} = 1 \\ \psi_{|-3} = \end{cases}$$

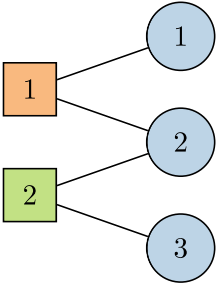
Toy example



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = \\ \psi_{|-2} = 1 \\ \psi_{|-3} = \end{cases}$$

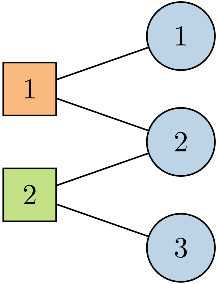
Toy example



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = \\ \psi_{|-2} = 1 \\ \psi_{|-3} = \end{cases}$$

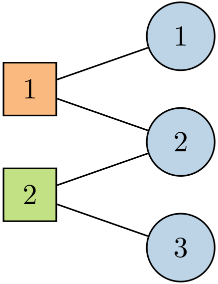
Toy example



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = 1 - \dfrac{\lambda_2}{\mu_2 + \mu_3} \\ \psi_{|-2} = 1 \\ \psi_{|-3} = \end{cases}$$

Toy example



$$\psi = (1-\rho) \times \frac{\mu_1 + \mu_2 + \mu_3}{\frac{\mu_1}{\psi_{|-1}} + \frac{\mu_2}{\psi_{|-2}} + \frac{\mu_3}{\psi_{|-3}}} \text{ with } \rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2 + \mu_3}$$

$$\begin{cases} \psi_{|-1} = 1 - \dfrac{\lambda_2}{\mu_2 + \mu_3} \\ \psi_{|-2} = 1 \\ \psi_{|-3} = 1 - \dfrac{\lambda_1}{\mu_1 + \mu_2} \end{cases}$$

Proof

- **Law of total probability**

$$\mathbb{P}\begin{pmatrix}\text{the queue}\\ \text{is empty}\end{pmatrix} = \mathbb{P}\begin{pmatrix}\text{server } s\\ \text{is idle}\end{pmatrix} \times \mathbb{P}\left(\begin{matrix}\text{the queue}\\ \text{is empty}\end{matrix} \,\middle|\, \begin{matrix}\text{server } s\\ \text{is idle}\end{matrix}\right)$$
$$+ \mathbb{P}\begin{pmatrix}\text{server } s\\ \text{is active}\end{pmatrix} \times \mathbb{P}\left(\begin{matrix}\text{the queue}\\ \text{is empty}\end{matrix} \,\middle|\, \begin{matrix}\text{server } s\\ \text{is active}\end{matrix}\right)$$

Proof

- **Law of total probability**

$$\mathbb{P}\begin{pmatrix}\text{the queue}\\ \text{is empty}\end{pmatrix} = \mathbb{P}\begin{pmatrix}\text{server } s\\ \text{is idle}\end{pmatrix} \times \mathbb{P}\left(\begin{matrix}\text{the queue}\\ \text{is empty}\end{matrix} \,\middle|\, \begin{matrix}\text{server } s\\ \text{is idle}\end{matrix}\right)$$

Proof

- **Law of total probability**

$$\underbrace{\mathbb{P}\begin{pmatrix}\text{the queue}\\ \text{is empty}\end{pmatrix}}_{\psi} = \mathbb{P}\begin{pmatrix}\text{server } s\\ \text{is idle}\end{pmatrix} \times \mathbb{P}\left(\begin{matrix}\text{the queue}\\ \text{is empty}\end{matrix} \,\middle|\, \begin{matrix}\text{server } s\\ \text{is idle}\end{matrix}\right)$$

Proof

- **Law of total probability**

$$\underbrace{\mathbb{P}\begin{pmatrix}\text{the queue} \\ \text{is empty}\end{pmatrix}}_{\psi} = \underbrace{\mathbb{P}\begin{pmatrix}\text{server } s \\ \text{is idle}\end{pmatrix}}_{\psi_s} \times \mathbb{P}\left(\begin{matrix}\text{the queue} \\ \text{is empty}\end{matrix} \,\middle|\, \begin{matrix}\text{server } s \\ \text{is idle}\end{matrix}\right)$$

$$\psi = \psi_s$$

Proof

- **Law of total probability**

$$\underbrace{\mathbb{P}\begin{pmatrix}\text{the queue}\\ \text{is empty}\end{pmatrix}}_{\psi} = \underbrace{\mathbb{P}\begin{pmatrix}\text{server } s\\ \text{is idle}\end{pmatrix}}_{\psi_s} \times \underbrace{\mathbb{P}\left(\begin{matrix}\text{the queue}\\ \text{is empty}\end{matrix} \,\middle|\, \begin{matrix}\text{server } s\\ \text{is idle}\end{matrix}\right)}_{\psi_{|-s}}$$

$$\psi = \psi_s \times \psi_{|-s}$$

Proof

- **Law of total probability**

$$\underbrace{\mathbb{P}\begin{pmatrix}\text{the queue}\\ \text{is empty}\end{pmatrix}}_{\psi} = \underbrace{\mathbb{P}\begin{pmatrix}\text{server } s\\ \text{is idle}\end{pmatrix}}_{\psi_s} \times \underbrace{\mathbb{P}\left(\begin{matrix}\text{the queue}\\ \text{is empty}\end{matrix}\,\middle|\,\begin{matrix}\text{server } s\\ \text{is idle}\end{matrix}\right)}_{\psi_{|-s}}$$

- **Conservation equation**

$$\sum_i \lambda_i = \sum_s \mu_s (1 - \psi_s)$$

Proof

- **Law of total probability**

$$\underbrace{\mathbb{P}\begin{pmatrix}\text{the queue}\\\text{is empty}\end{pmatrix}}_{\psi} = \underbrace{\mathbb{P}\begin{pmatrix}\text{server } s\\\text{is idle}\end{pmatrix}}_{\psi_s} \times \underbrace{\mathbb{P}\left(\begin{matrix}\text{the queue}\\\text{is empty}\end{matrix}\,\middle|\,\begin{matrix}\text{server } s\\\text{is idle}\end{matrix}\right)}_{\psi_{|-s}}$$

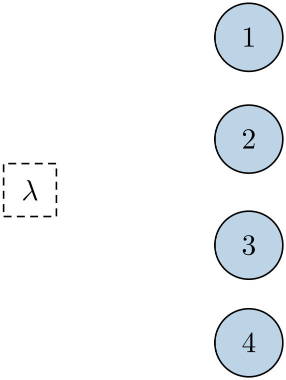
- **Conservation equation**

$$\sum_i \lambda_i = \sum_s \mu_s (1 - \psi_s) \qquad \square$$

Queues where complexity is polynomial in the number of servers

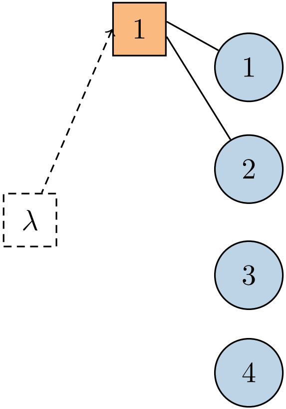
Global static random assignment

Global static random assignment



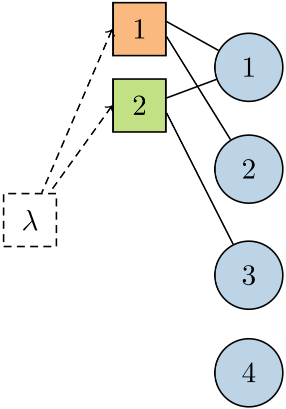
Queues where complexity is polynomial in the number of servers

Global static random assignment



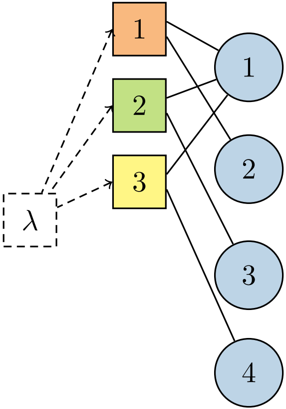
Queues where complexity is polynomial in the number of servers

Global static random assignment



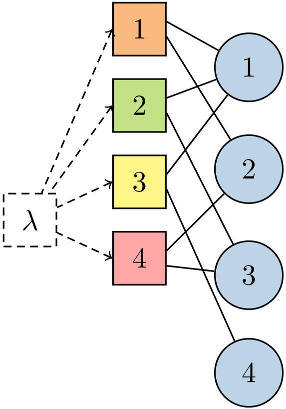
Queues where complexity is polynomial in the number of servers

Global static random assignment



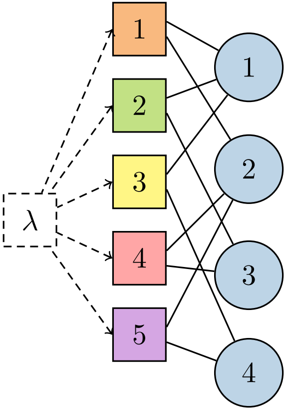
Queues where complexity is polynomial in the number of servers

Global static random assignment

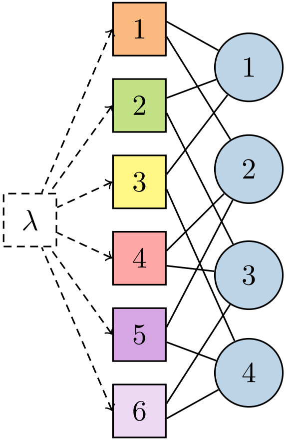


Queues where complexity is polynomial in the number of servers

Global static random assignment

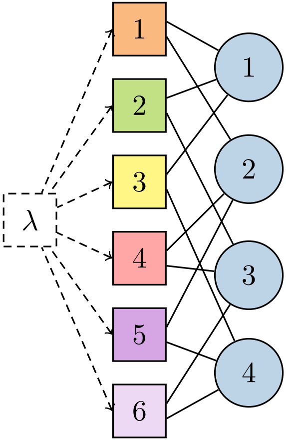


Global static random assignment



Queues where complexity is polynomial in the number of servers

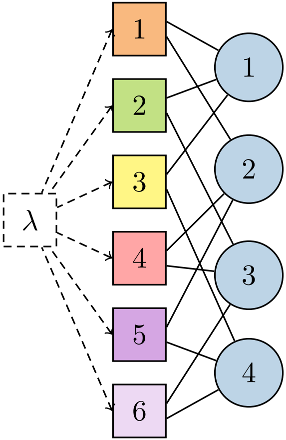
Global static random assignment



Line structure

Queues where complexity is polynomial in the number of servers

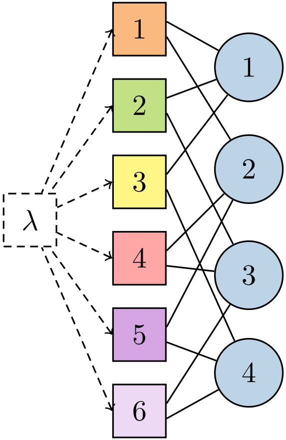
Global static random assignment



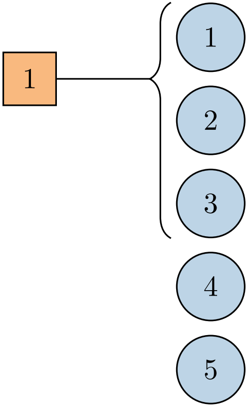
Line structure

Queues where complexity is polynomial in the number of servers

Global static random assignment

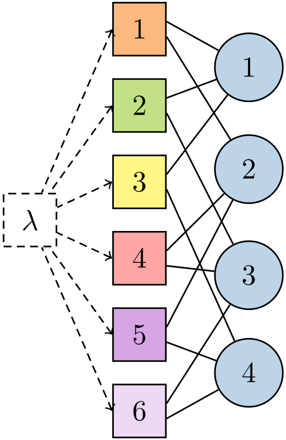


Line structure

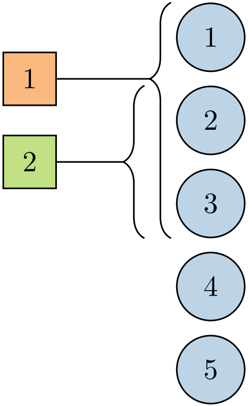


Queues where complexity is polynomial in the number of servers

Global static random assignment

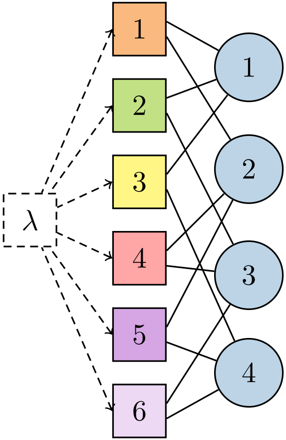


Line structure

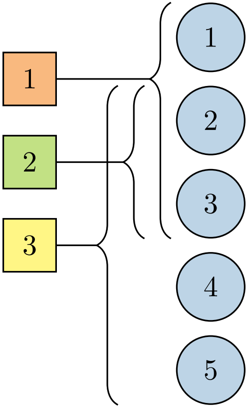


Queues where complexity is polynomial in the number of servers

Global static random assignment

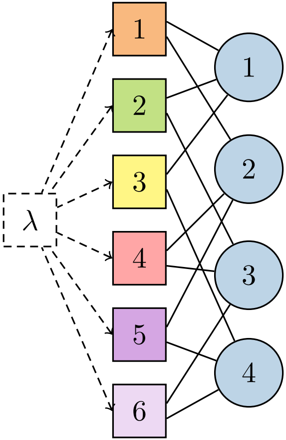


Line structure

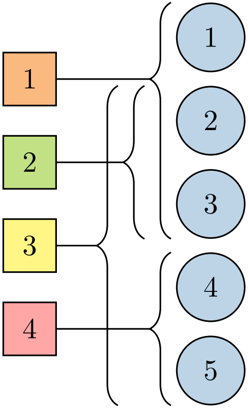


Queues where complexity is polynomial in the number of servers

Global static random assignment

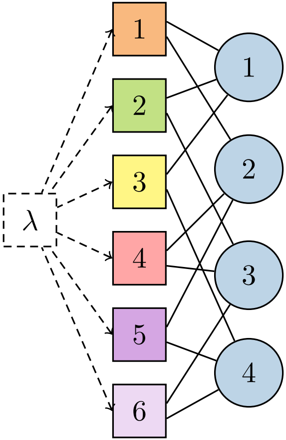


Line structure

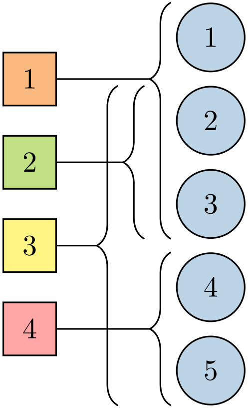


Queues where complexity is polynomial in the number of servers

Global static random assignment



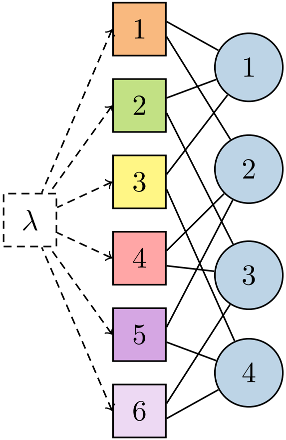
Line structure



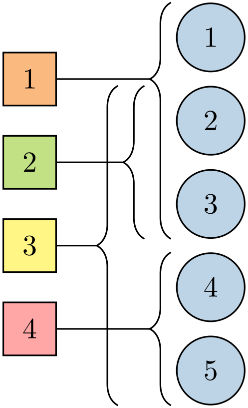
Ring structure

Queues where complexity is polynomial in the number of servers

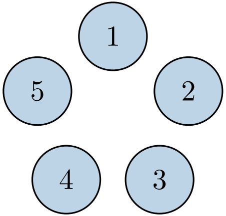
Global static random assignment



Line structure

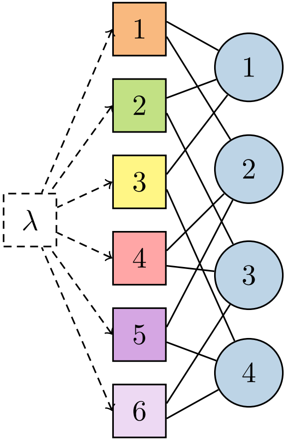


Ring structure

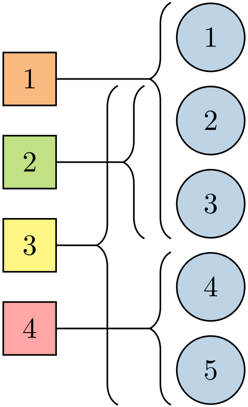


Queues where complexity is polynomial in the number of servers

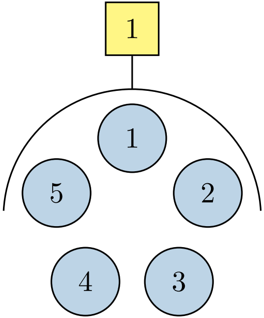
Global static random assignment



Line structure

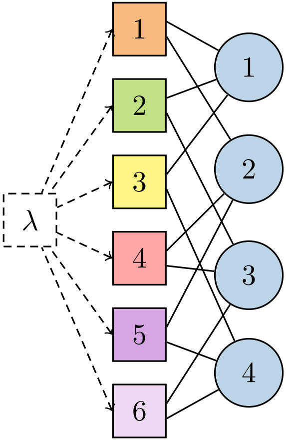


Ring structure

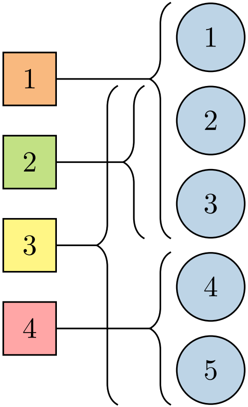


Queues where complexity is polynomial in the number of servers

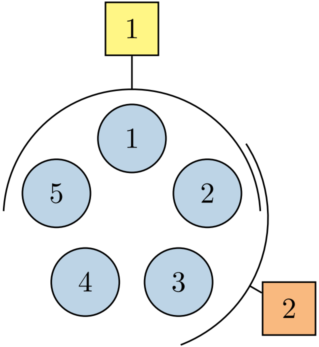
Global static random assignment



Line structure

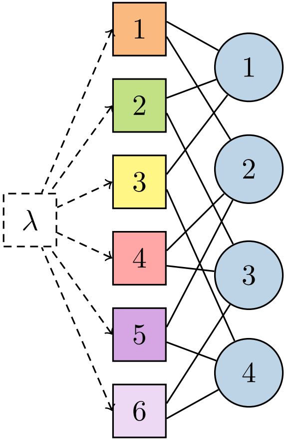


Ring structure

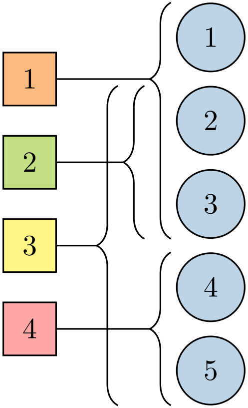


Queues where complexity is polynomial in the number of servers

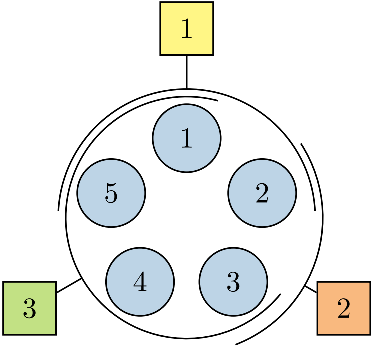
Global static random assignment



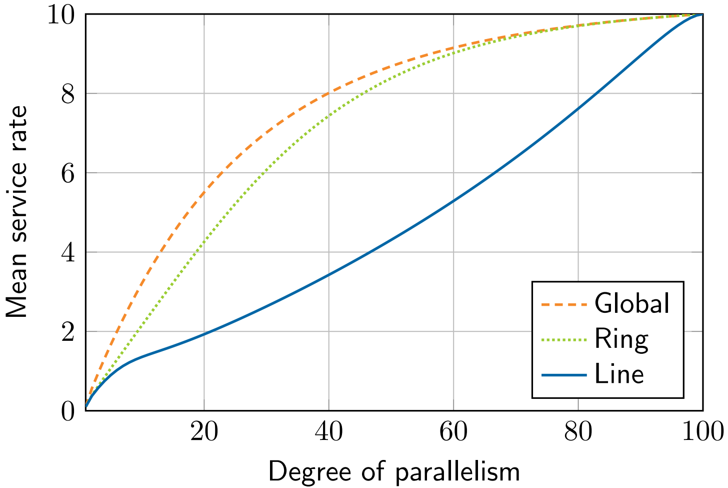
Line structure



Ring structure

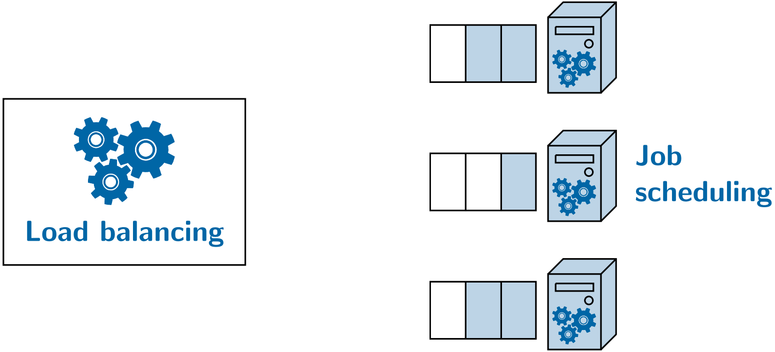


Static random assignment (100 servers, load $\rho = 0.9$)

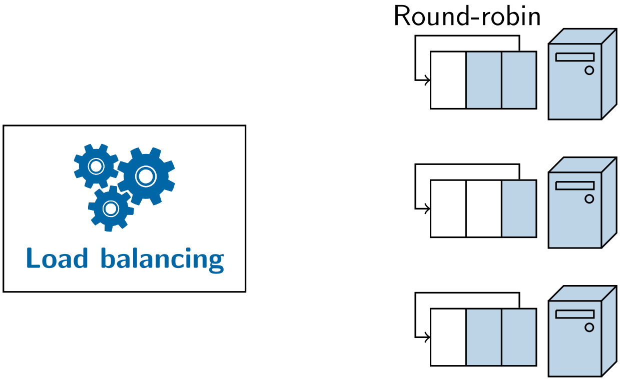


Outline

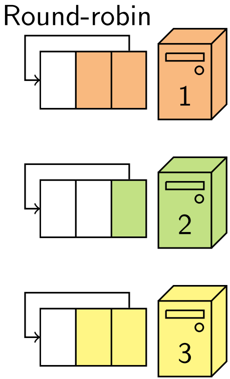
Load-balancing algorithm



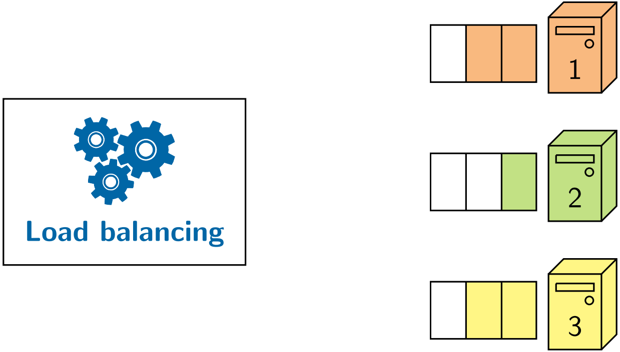
Load-balancing algorithm



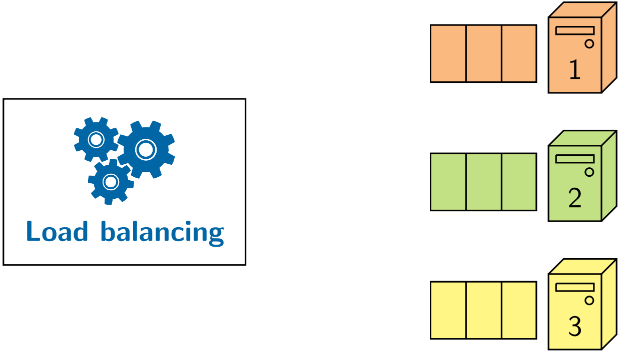
Load-balancing algorithm



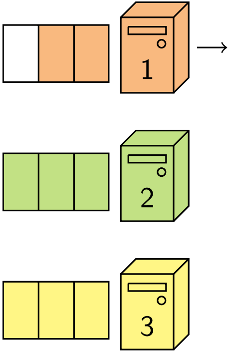
Load-balancing algorithm



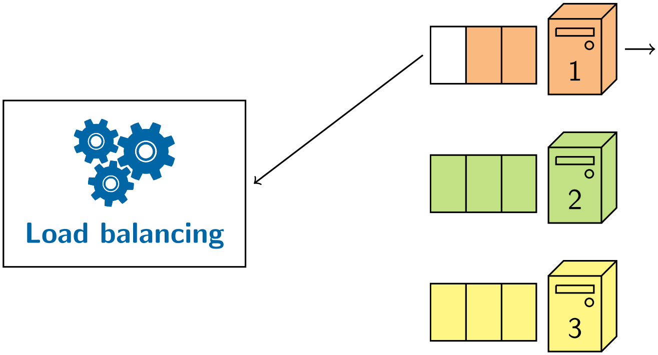
Load-balancing algorithm



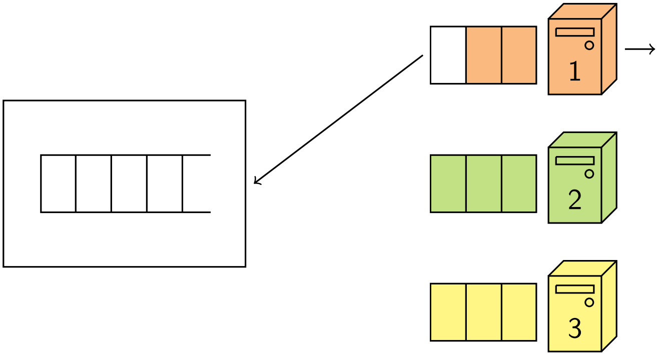
Load-balancing algorithm



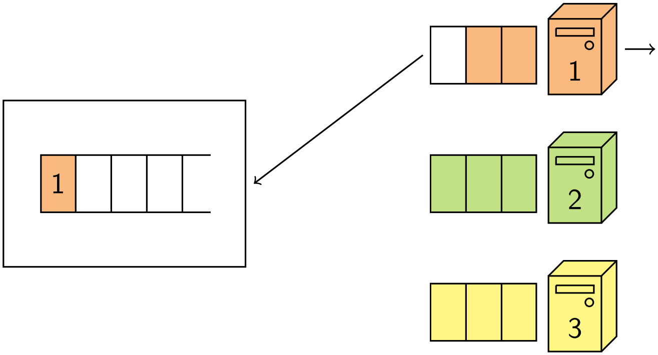
Load-balancing algorithm



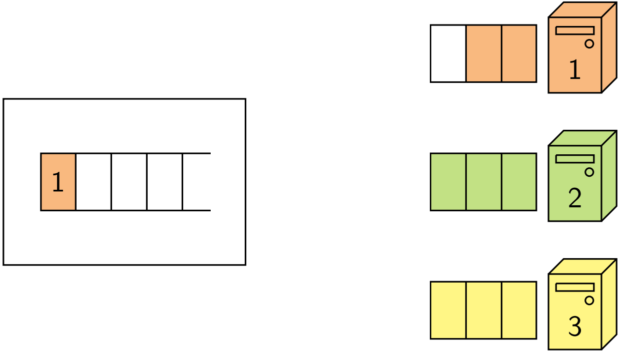
Load-balancing algorithm



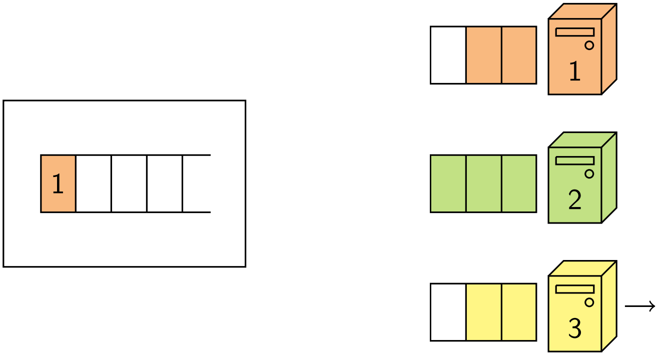
Load-balancing algorithm



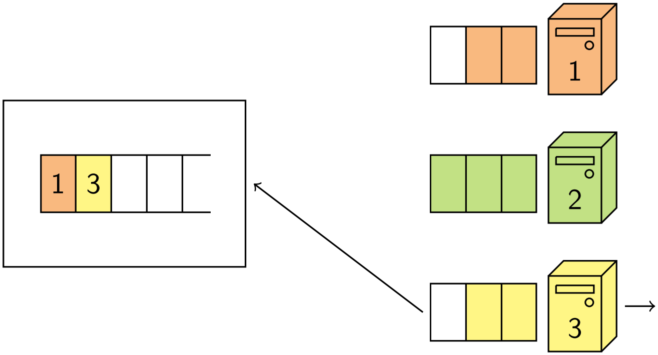
Load-balancing algorithm



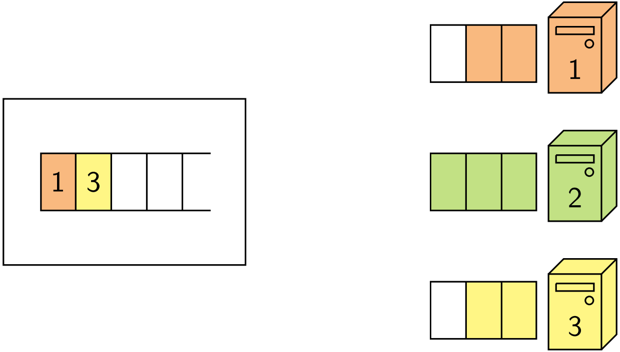
Load-balancing algorithm



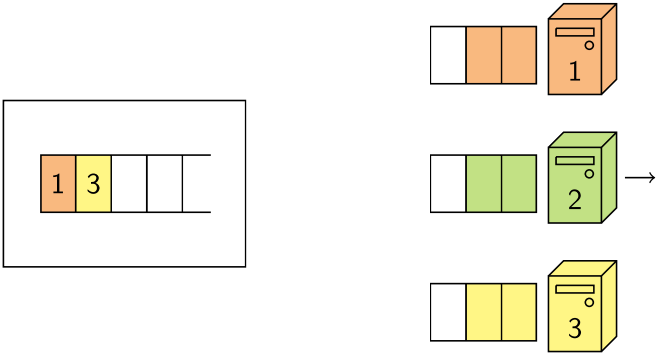
Load-balancing algorithm



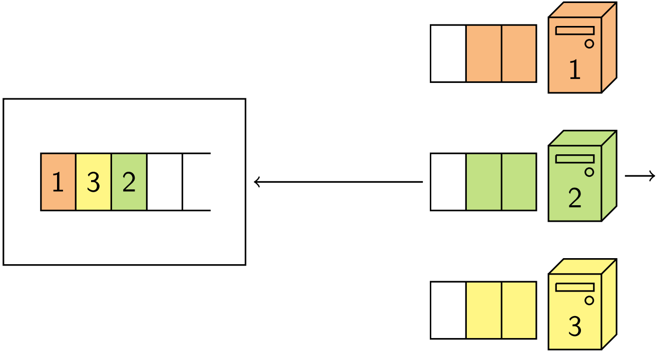
Load-balancing algorithm



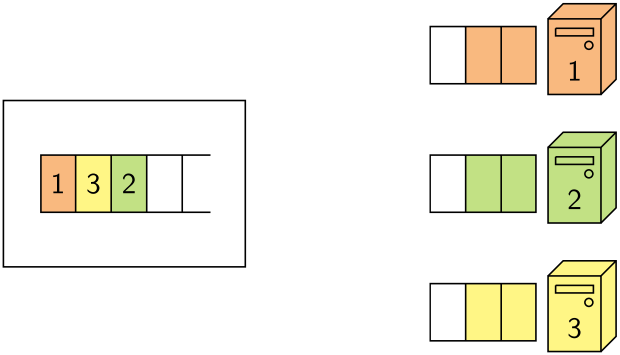
Load-balancing algorithm



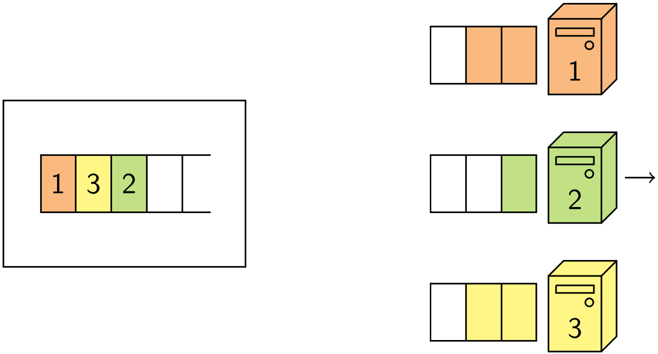
Load-balancing algorithm



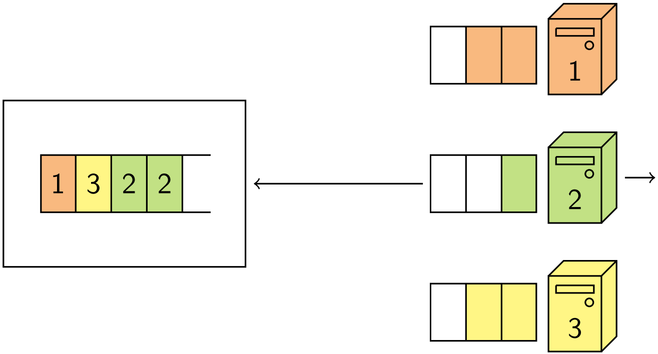
Load-balancing algorithm



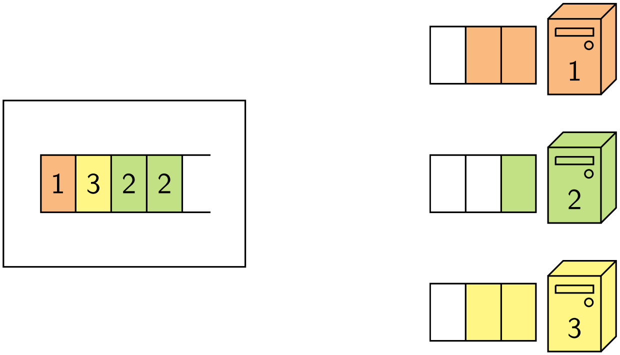
Load-balancing algorithm



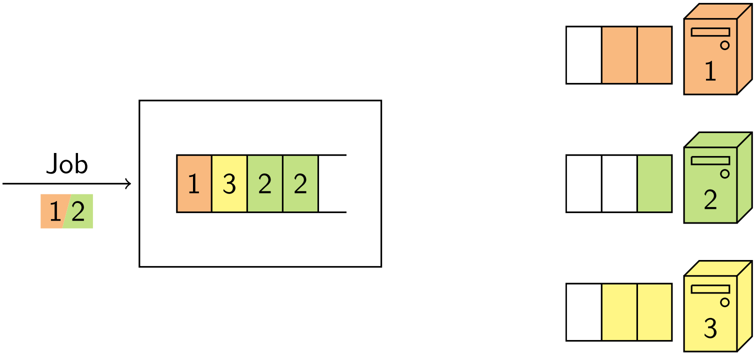
Load-balancing algorithm



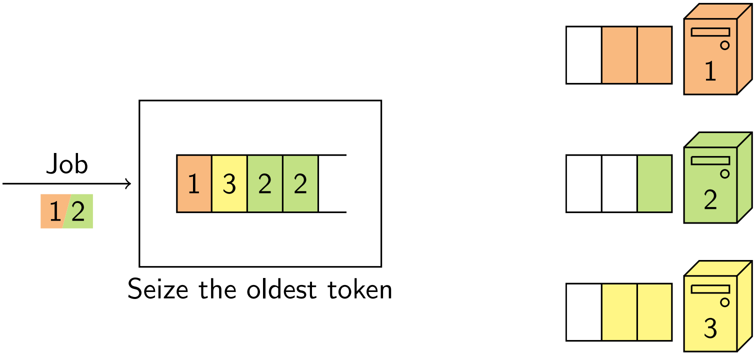
Load-balancing algorithm



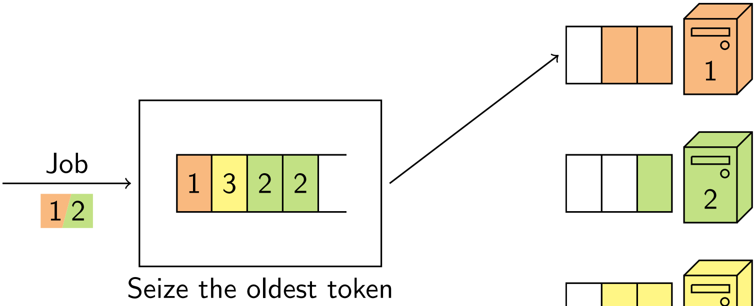
Load-balancing algorithm



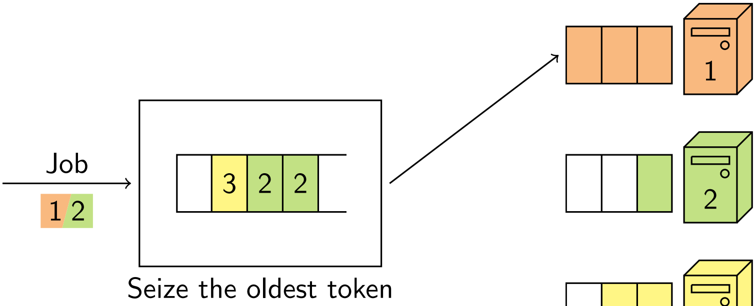
Load-balancing algorithm



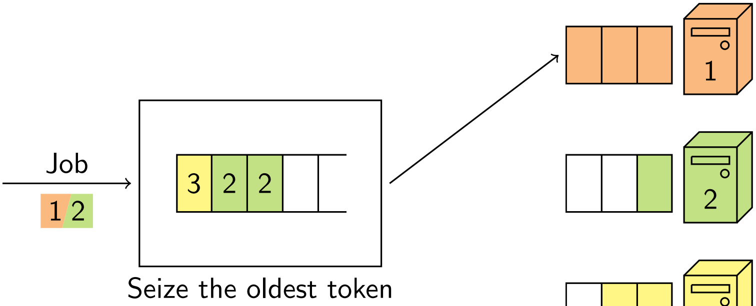
Load-balancing algorithm



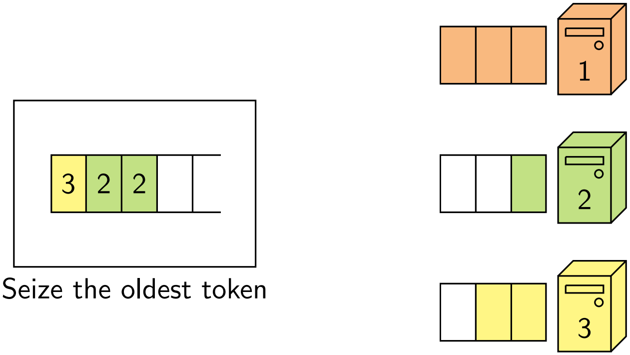
Load-balancing algorithm



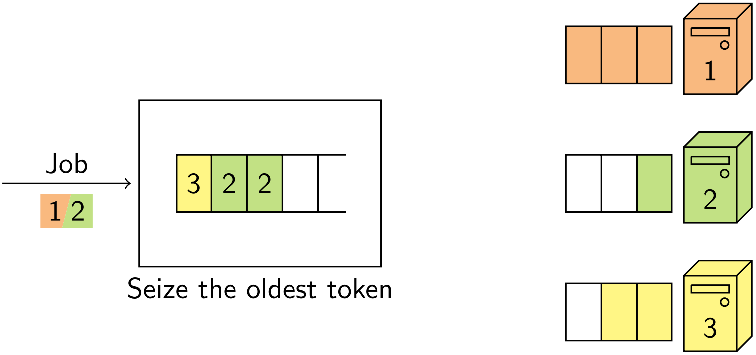
Load-balancing algorithm

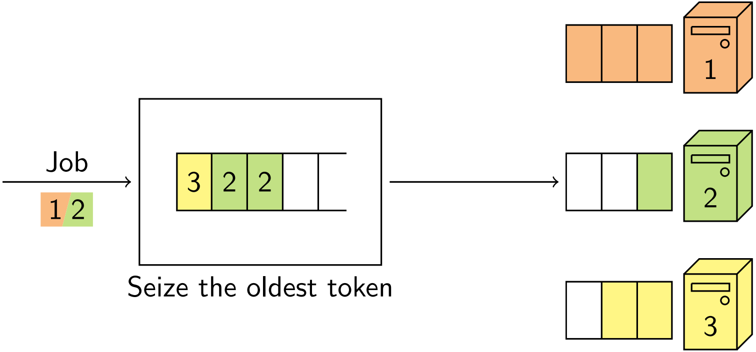


Load-balancing algorithm

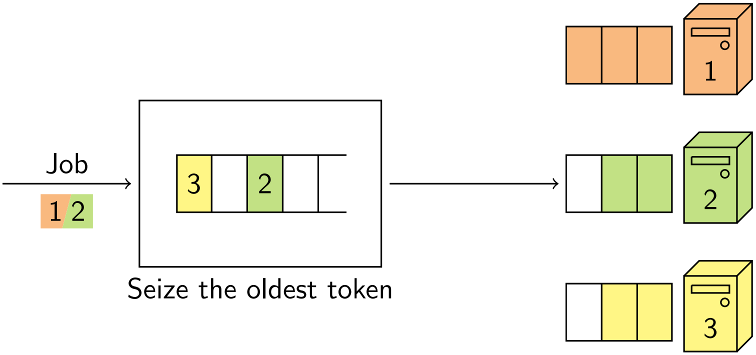


Load-balancing algorithm

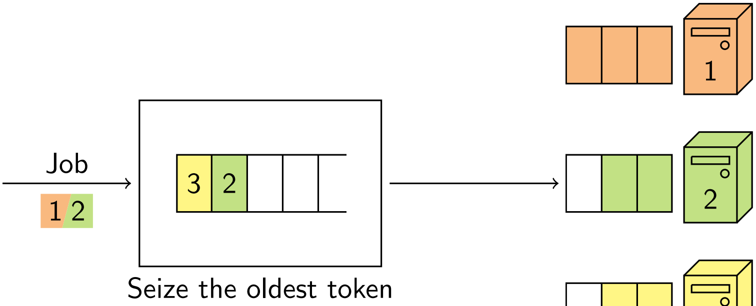



Job
1 2
3 2 2
Seize the oldest token
1
2
3

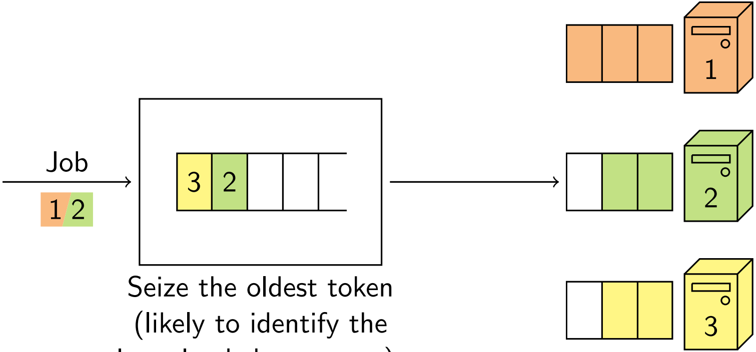
Load-balancing algorithm



Load-balancing algorithm

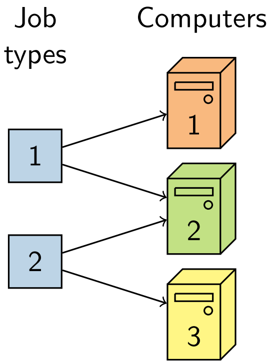


Load-balancing algorithm

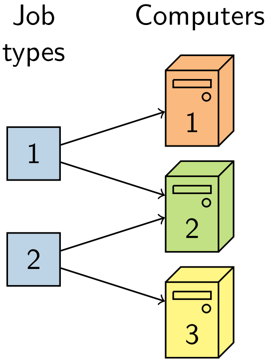


Seize the oldest token
(likely to identify the
least loaded computer)

Cluster model

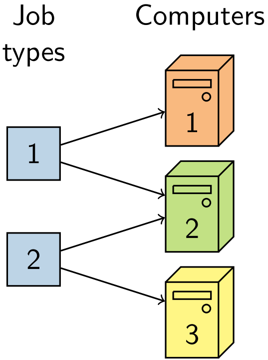


Cluster model



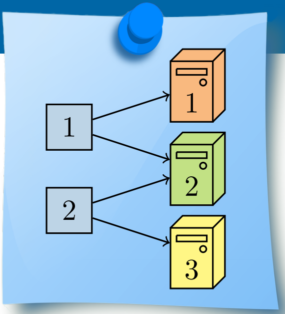
- **Markovian assumptions**
  - → Type-i jobs arrive according to a Poisson process with rate ν_i
  - → Computer s has capacity μ_s
  - → Job sizes are independent and exponentially distributed with unit mean

Cluster model

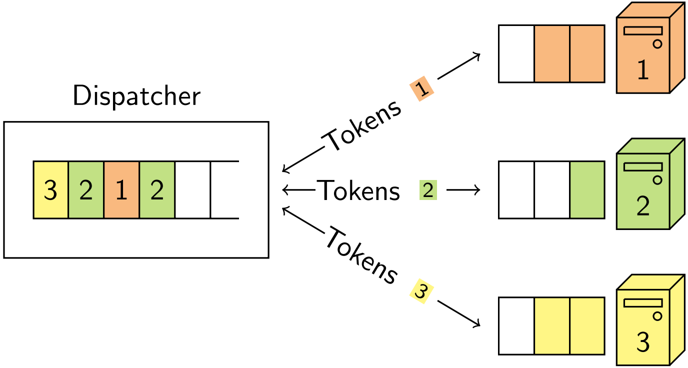


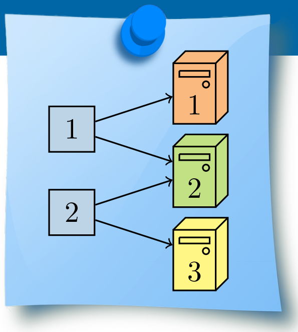
- **Markovian assumptions**
  - → Type-i jobs arrive according to a Poisson process with rate ν_i
  - → Computer s has capacity μ_s
  - → Job sizes are independent and exponentially distributed with unit mean

- **Admission limit**
  - → Computer s has ℓ_s tokens

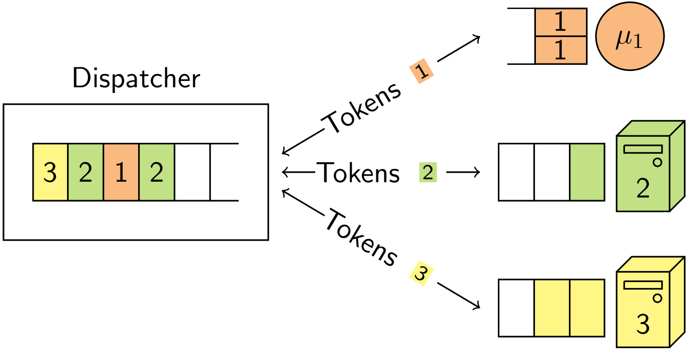

1
2
1
2
3

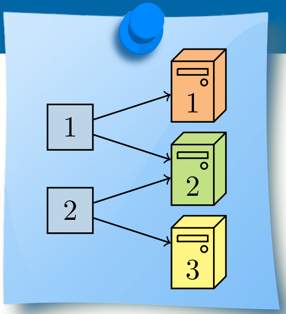
Queueing model



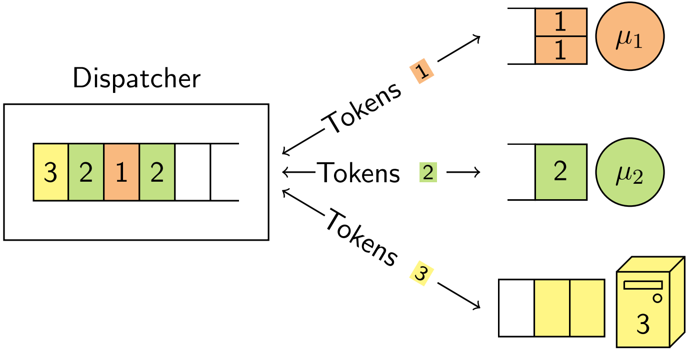


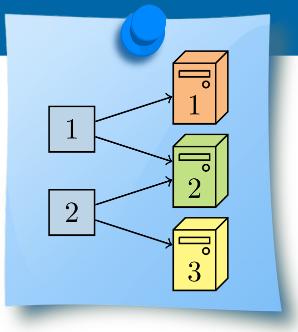
Queueing model



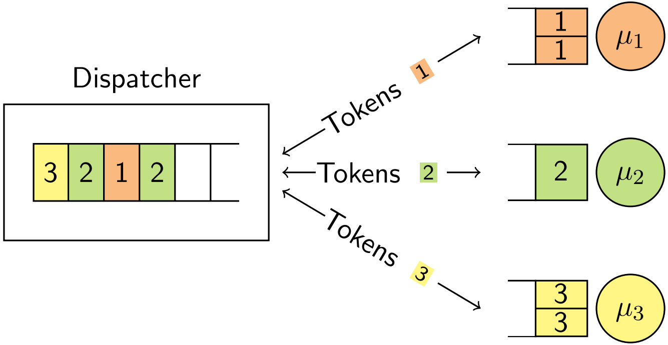


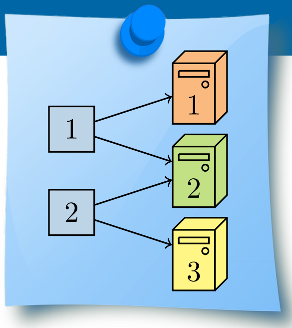
Queueing model



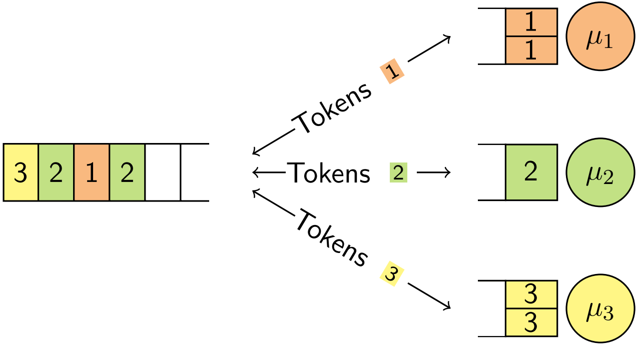


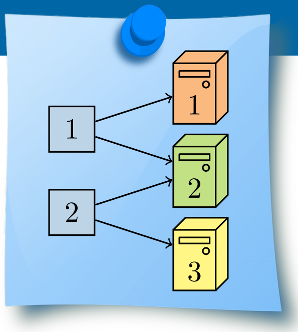
Queueing model

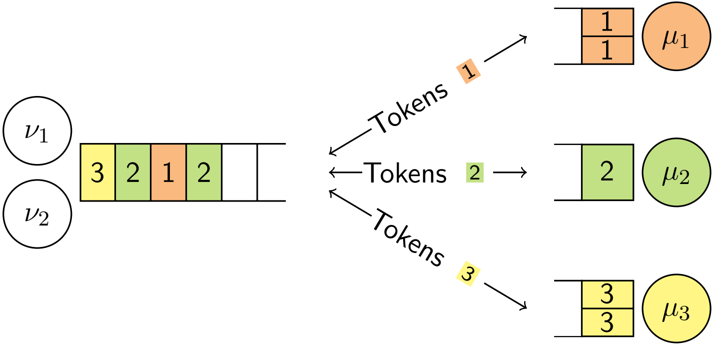




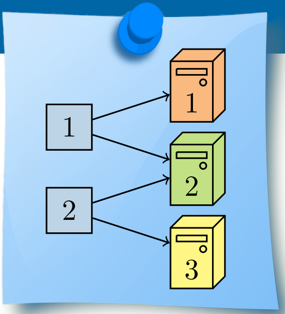
Queueing model





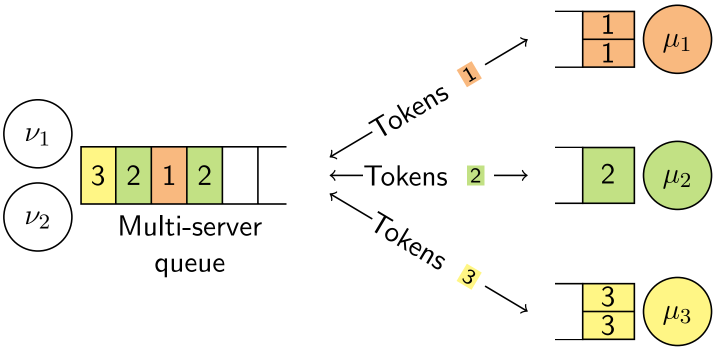


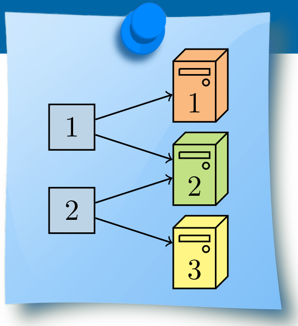
ν_1
ν_2
3 2 1 2
Tokens 1
Tokens 2
Tokens 3
1 1
2
3 3
μ_1
μ_2
μ_3



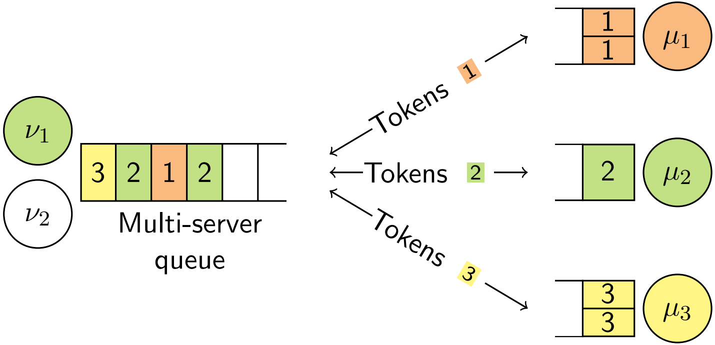
1
2
1
2
3

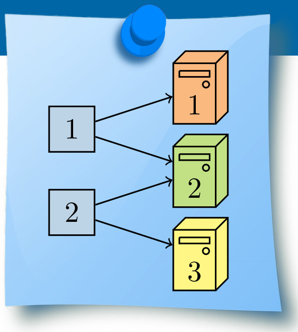
Queueing model

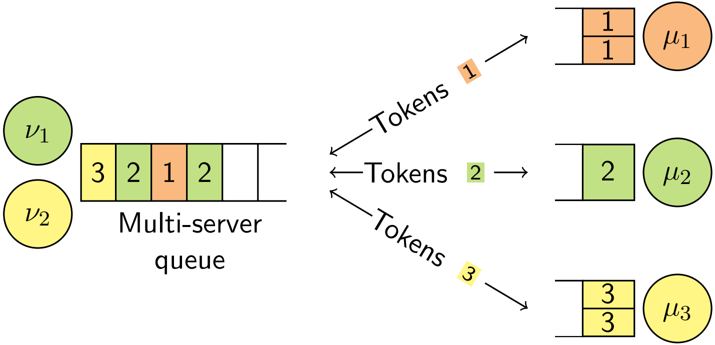




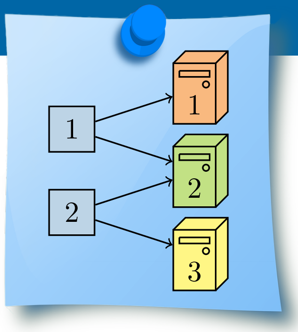
Queueing model





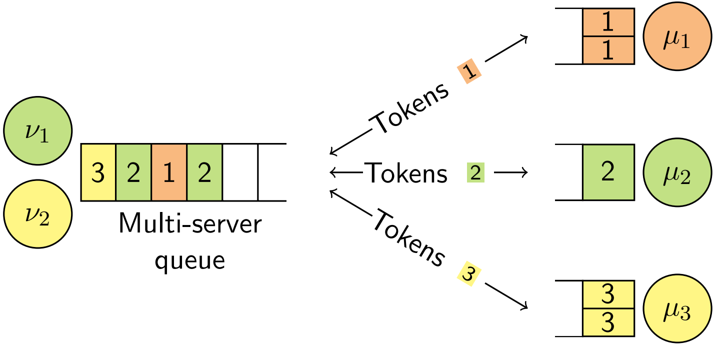


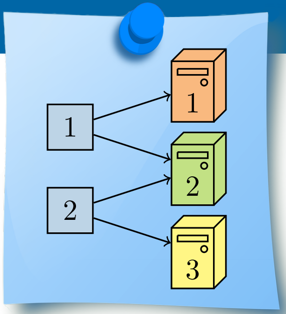
ν_1
ν_2
3 2 1 2
Multi-server queue
Tokens 1
Tokens 2
Tokens 3
1
1
μ_1
2
μ_2
3
3
μ_3



1
2
1
2
3

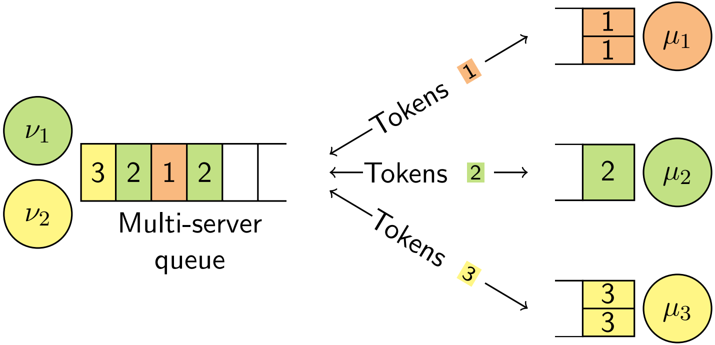
Queueing model

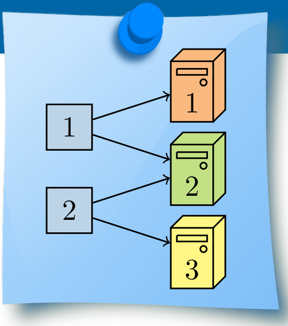




- Closed network of multi-server queues

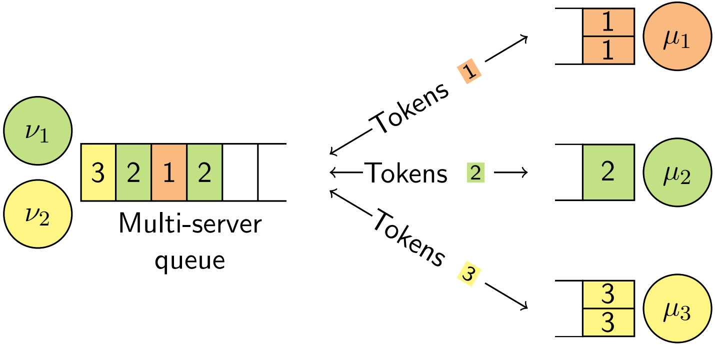
Queueing model

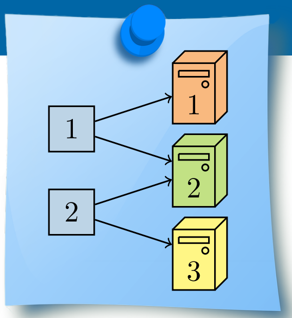




- Closed network of multi-server queues
- Closed-form expression for the stationary distribution

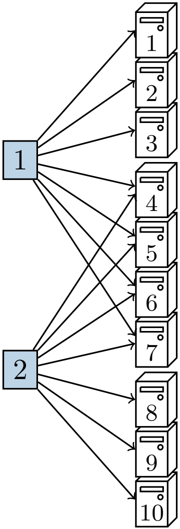
Queueing model



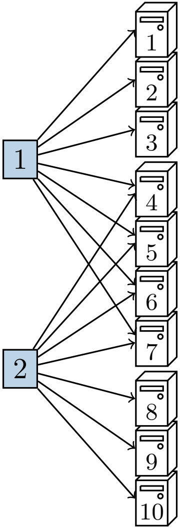


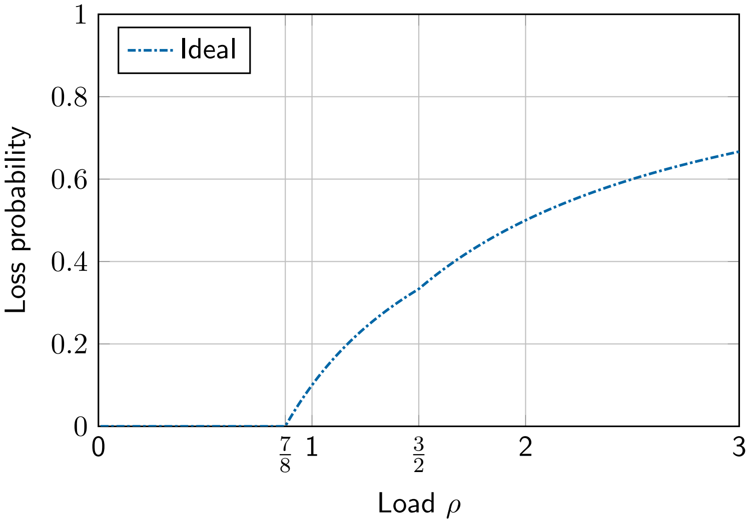
- Closed network of multi-server queues
- Closed-form expression for the stationary distribution
- Insensitivity to the job size distribution

Assignment constraints

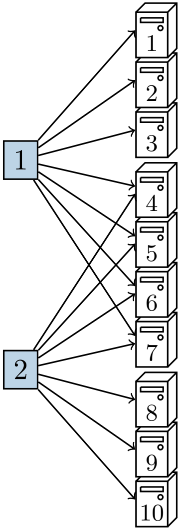


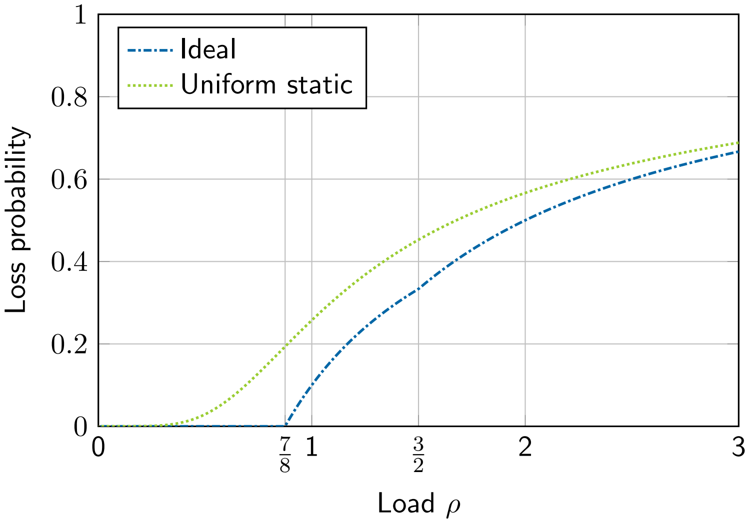
Assignment constraints



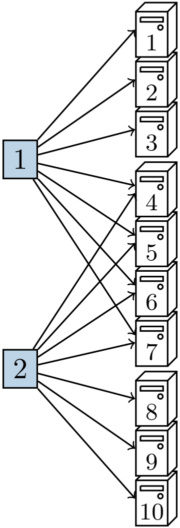


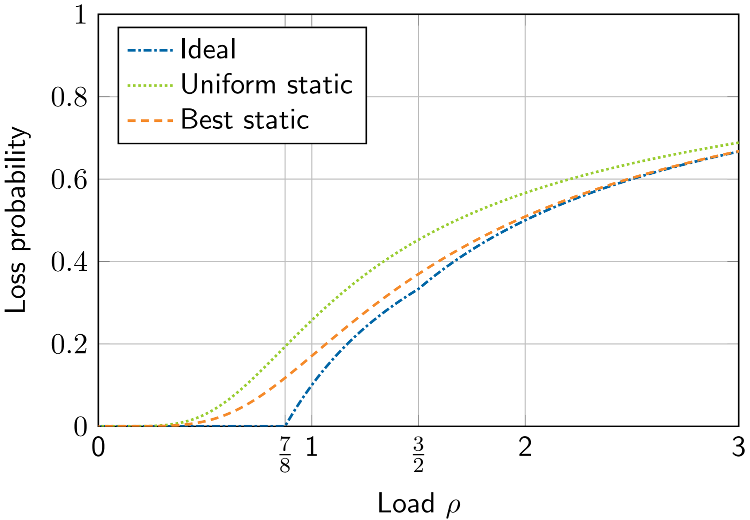
Assignment constraints



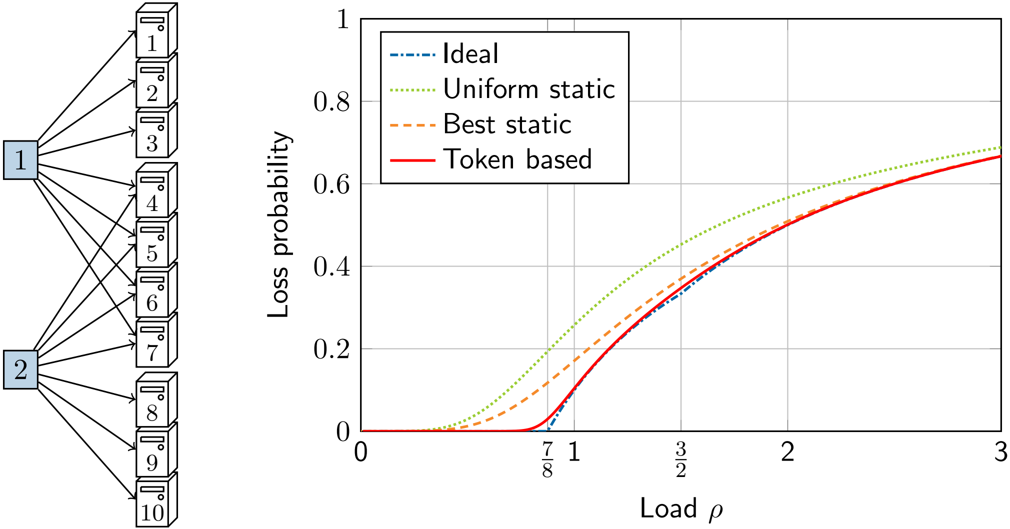


Assignment constraints





Assignment constraints



Conclusion

Takeaways

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Excursions during my Ph.D. thesis

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Excursions during my Ph.D. thesis

- Analytic combinatorics and queueing theory

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Excursions during my Ph.D. thesis

- Analytic combinatorics and queueing theory
- Simulating Kleinberg's grid

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Excursions during my Ph.D. thesis

- Analytic combinatorics and queueing theory
- Simulating Kleinberg's grid

Perspectives

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Excursions during my Ph.D. thesis

- Analytic combinatorics and queueing theory
- Simulating Kleinberg's grid

Perspectives

- Sensitive vs. insensitive algorithms

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Excursions during my Ph.D. thesis

- Analytic combinatorics and queueing theory
- Simulating Kleinberg's grid

Perspectives

- Sensitive vs. insensitive algorithms
- Multi-resource sharing

Conclusion

Takeaways

- Equivalence of balanced fairness and first-come-first-served with respect to the stationary distribution of the macrostate (extension to Whittle networks and order-independent queues)
- Closed-form expressions to compute performance metrics valid under both policies
- Applications in design and analysis of resource-management algorithms in computer clusters

Excursions during my Ph.D. thesis

- Analytic combinatorics and queueing theory
- Simulating Kleinberg's grid

Perspectives

- Sensitive vs. insensitive algorithms
- Multi-resource sharing
- Integrate learning into the process

International publications

- Balanced fair resource sharing in computer clusters.
  T. Bonald and C. Comte. Performance Evaluation (2017).
- Poly-symmetry in processor-sharing systems.
  T. Bonald, C. Comte, V. Shah, and G. de Veciana. Queueing Systems (2017).
- Performance of Balanced Fairness in Resource Pools: A Recursive Approach.
  T. Bonald, C. Comte, and F. Mathieu. SIGMETRICS (2018).
- Of Kernels and Queues: When Network Calculus Meets Analytic Combinatorics.
  A. Bouillard, C. Comte, E. de Panafieu, and F. Mathieu. NetCal (2018).
- Kleinberg's grid unchained.
  C. Comte and F. Mathieu. Theoretical Computer Science (2018).
- Dynamic Load Balancing with Tokens. C. Comte. IFIP Networking (2018).
- Dynamic Load Balancing with Tokens. C. Comte. Computer Communications (2019).