Dynamic Load Balancing with Tokens

Céline Comte

Young European Queueing Theorists XII
December 3, 2018

Introduction

Video on demand

ichokedonmylatte.wordpress.com

Introduction

Video on demand

ichokedonmylatte.wordpress.com

Computer clusters

www.cnet.com

Introduction

Video on demand

ichokedonmylatte.wordpress.com

Computer clusters

www.cnet.com

www.selserschaefer.com

Supermarket

Introduction

Video on demand

ichokedonmylatte.wordpress.com

Computer clusters

www.cnet.com

www.selserschaefer.com

Supermarket

www.bostonglobe.com

Airport

Introduction

Video on demand

ichokedonmylatte.wordpress.com

Computer clusters

www.cnet.com

Heterogeneity + Compatibilities

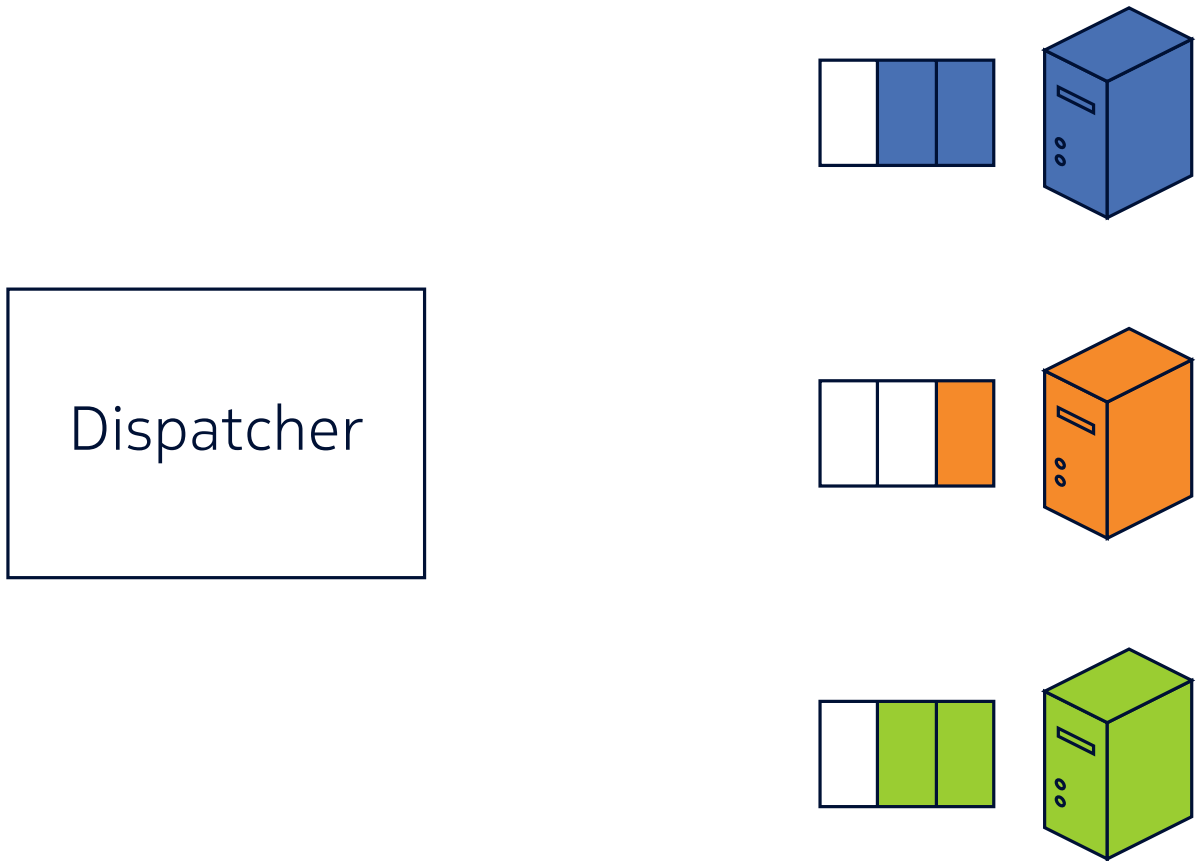
www.selserschaefer.com

Supermarket

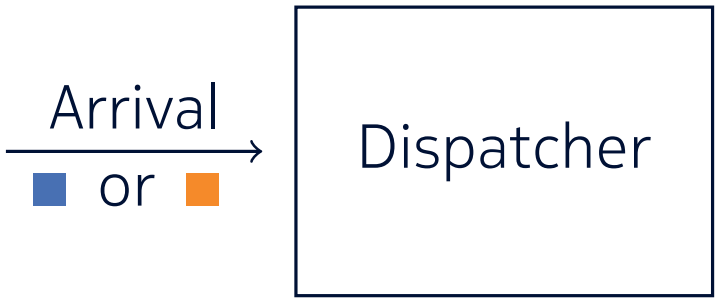
www.bostonglobe.com

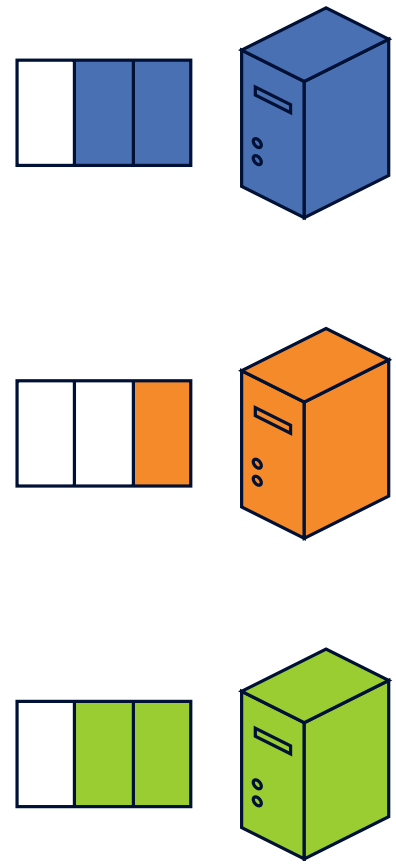
Airport

Abstraction

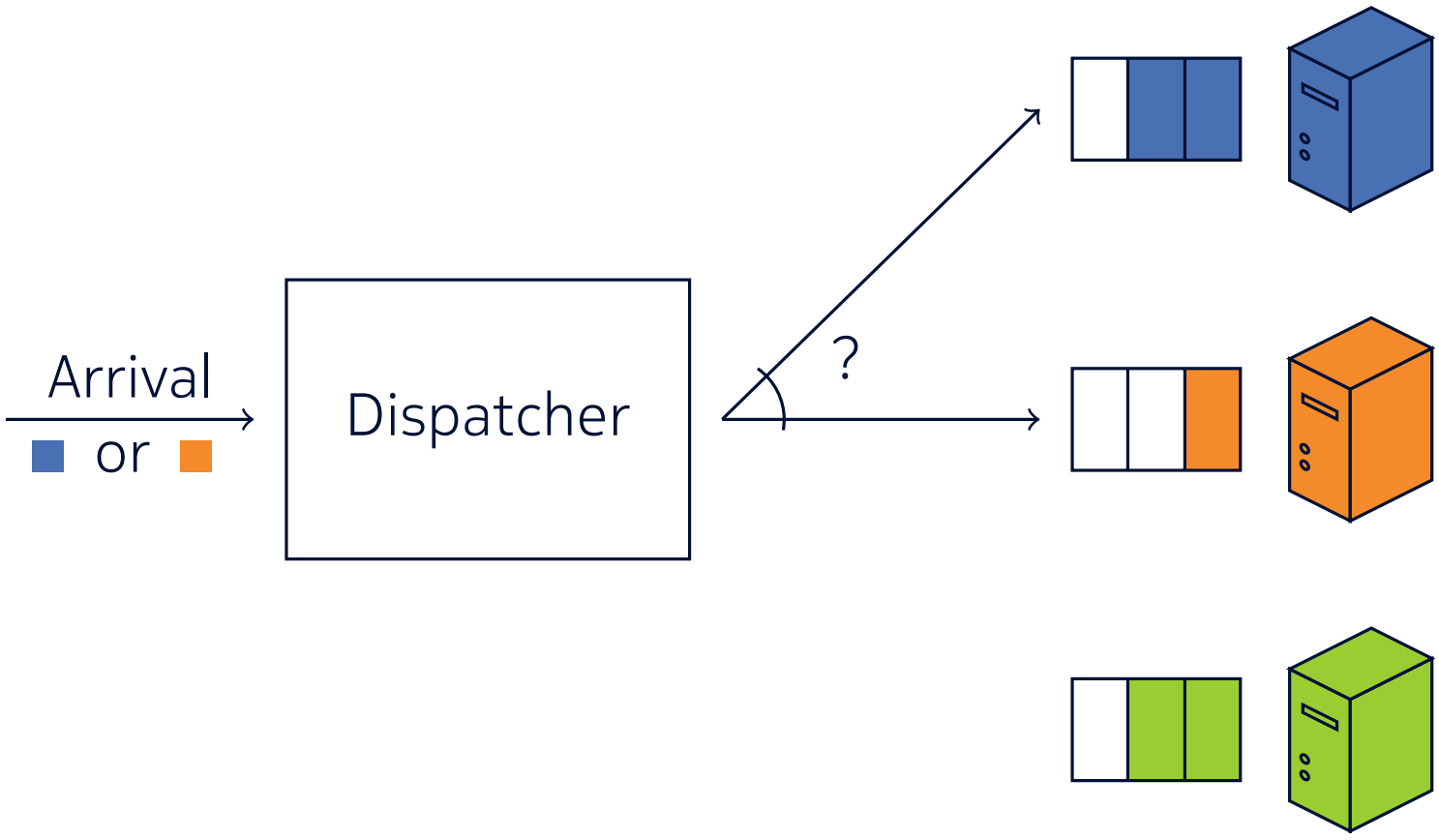


Abstraction





Abstraction



Our contributions

Load balancing algorithm

Robust and adaptative, yet simple

Our contributions

Load balancing algorithm
Robust and adaptative, yet simple

Queueing analysis
Under a Poisson arrival process

Our contributions

Load balancing algorithm
Robust and adaptative, yet simple

Queueing analysis
Under a Poisson arrival process

Relate several existing works

- Join-Idle-Queue (Lu et al., 2011)
- Assign to the longest idle server (ALIS) and FCFS-ALIS (Adan and Weiss, 2012) based on order-independent queues (Berezner et al., 1995)
- Insensitive load balancing (Bonald et al., 2004) based on Whittle networks (Whittle, 1985)

Outline

Algorithm

Queueing analysis

Numerical results

Extensions

Outline

Algorithm

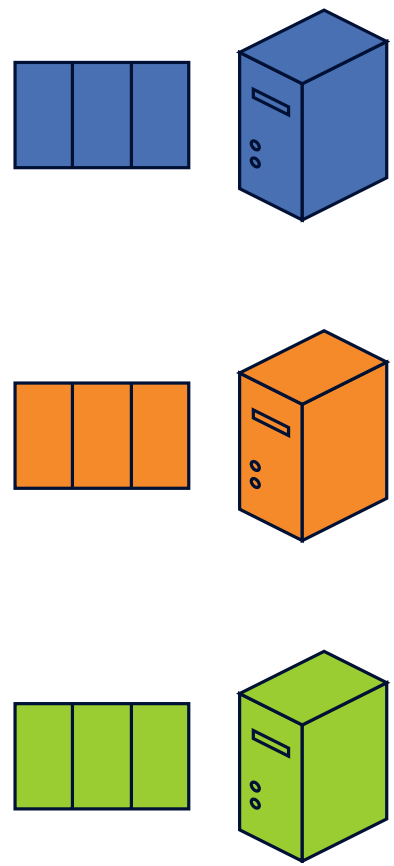
Queueing analysis

Numerical results

Extensions

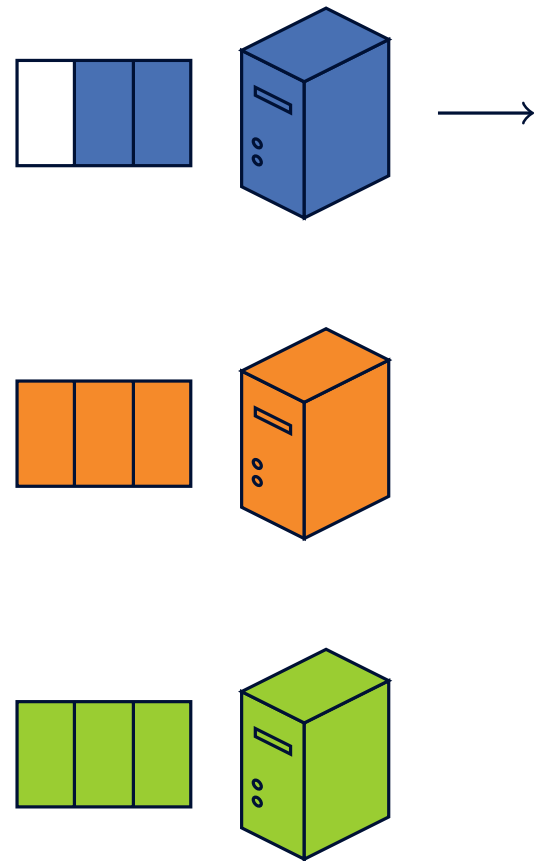
Algorithm

Dispatcher

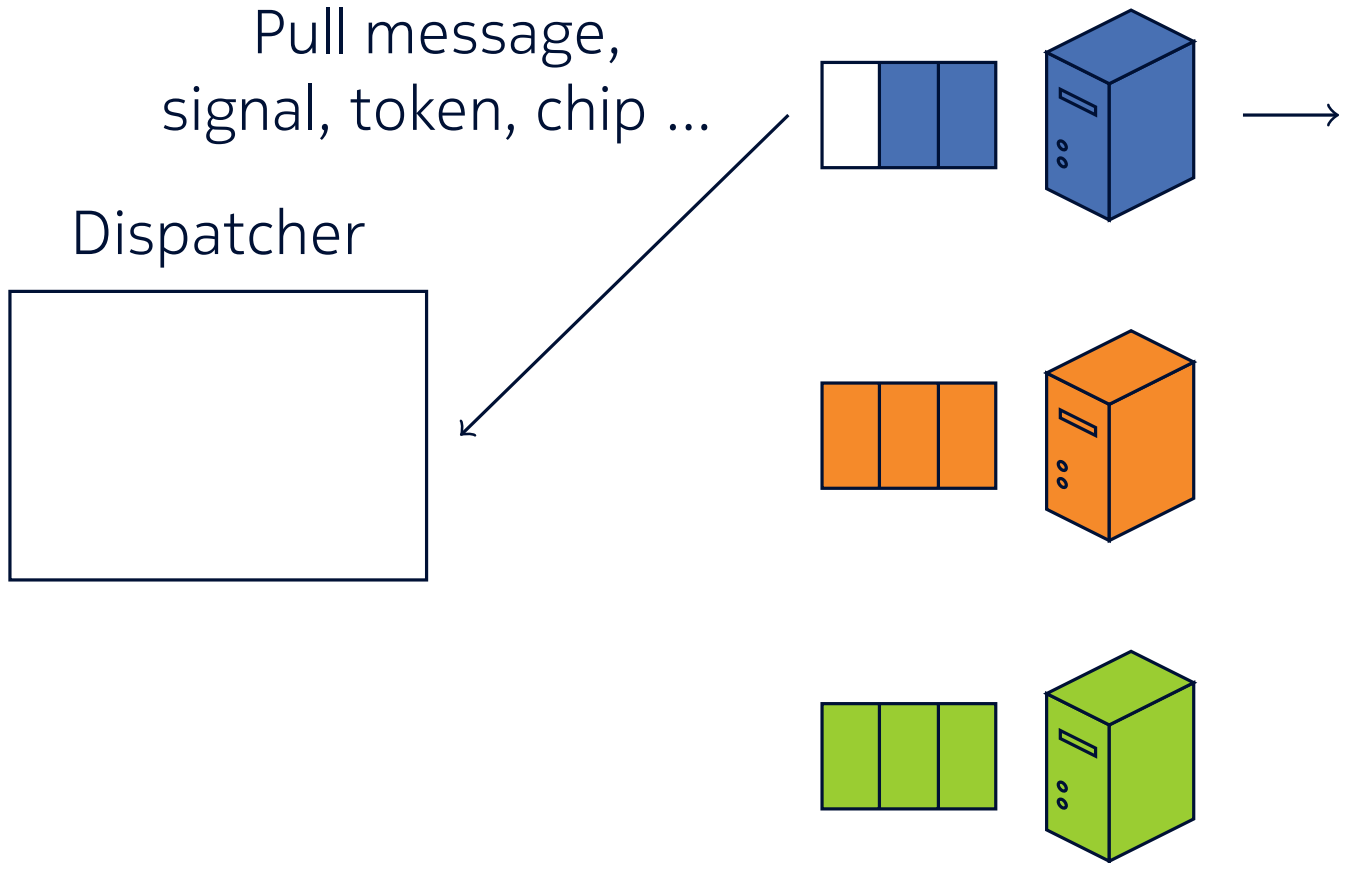


Algorithm

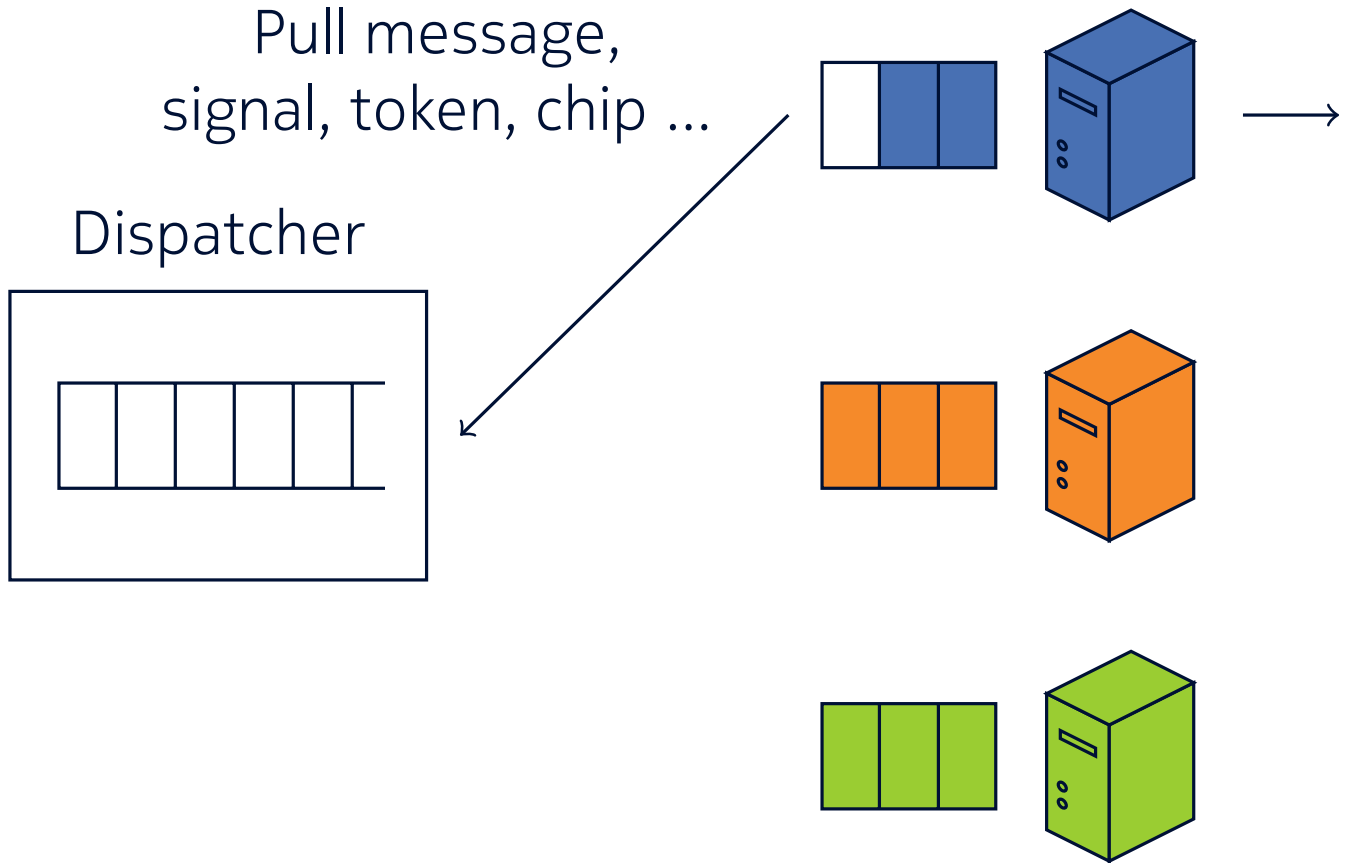
Dispatcher



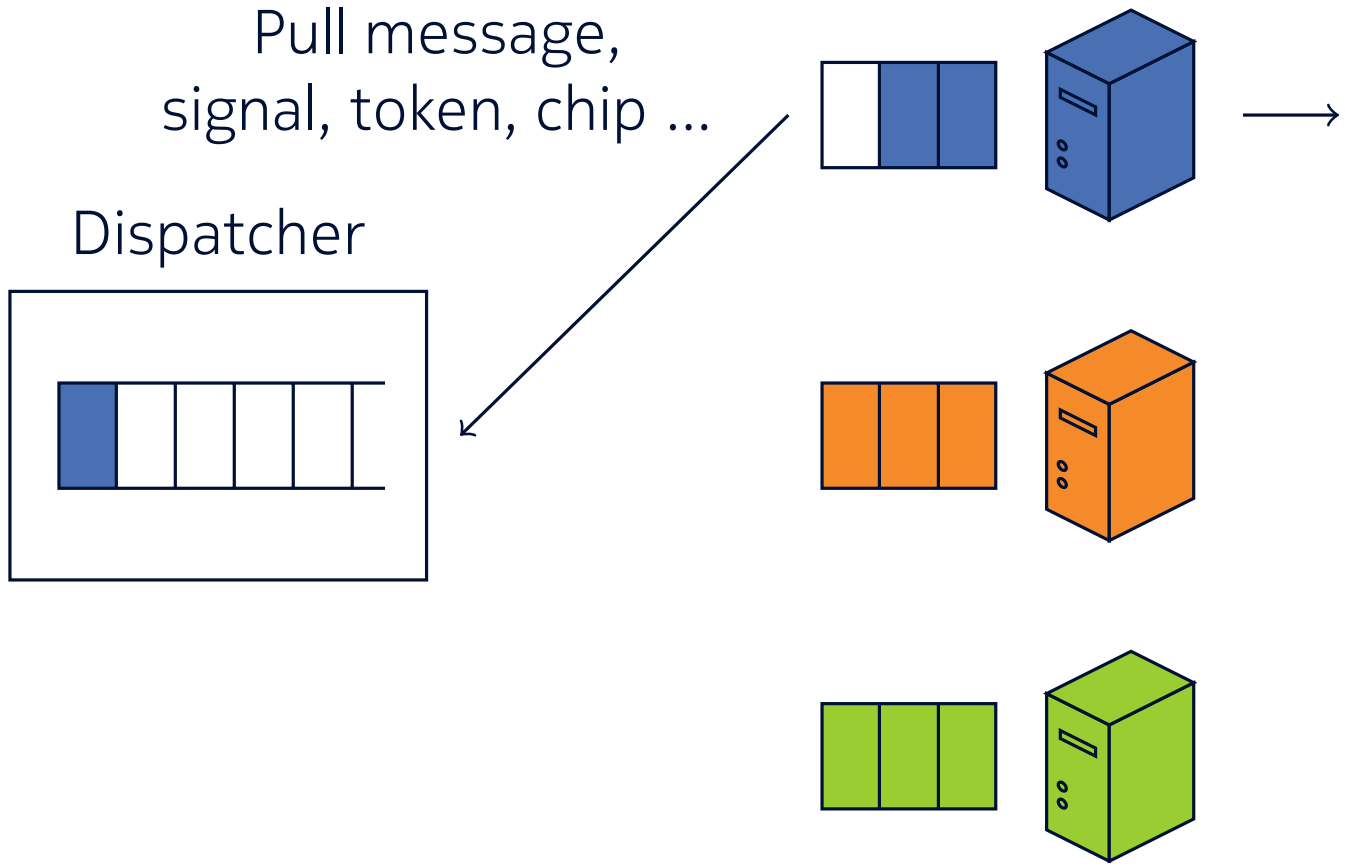
Algorithm



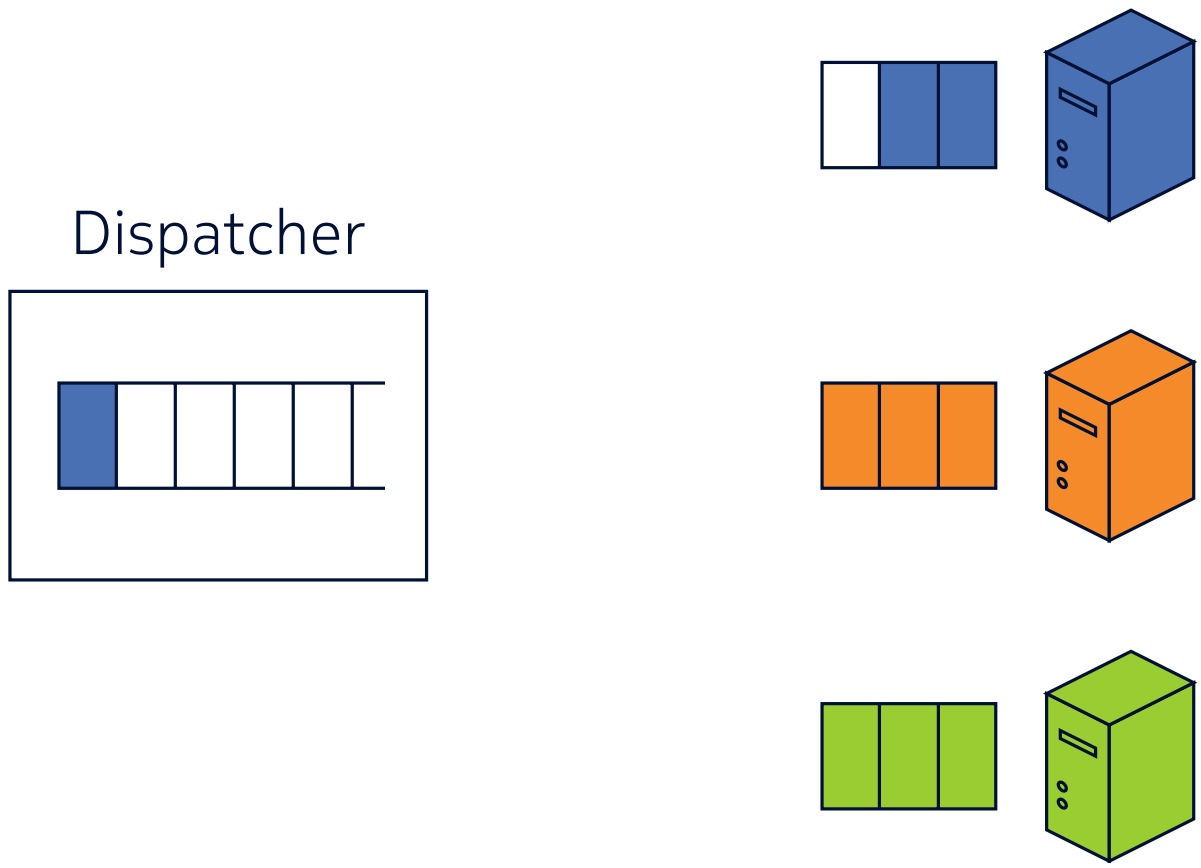
Algorithm



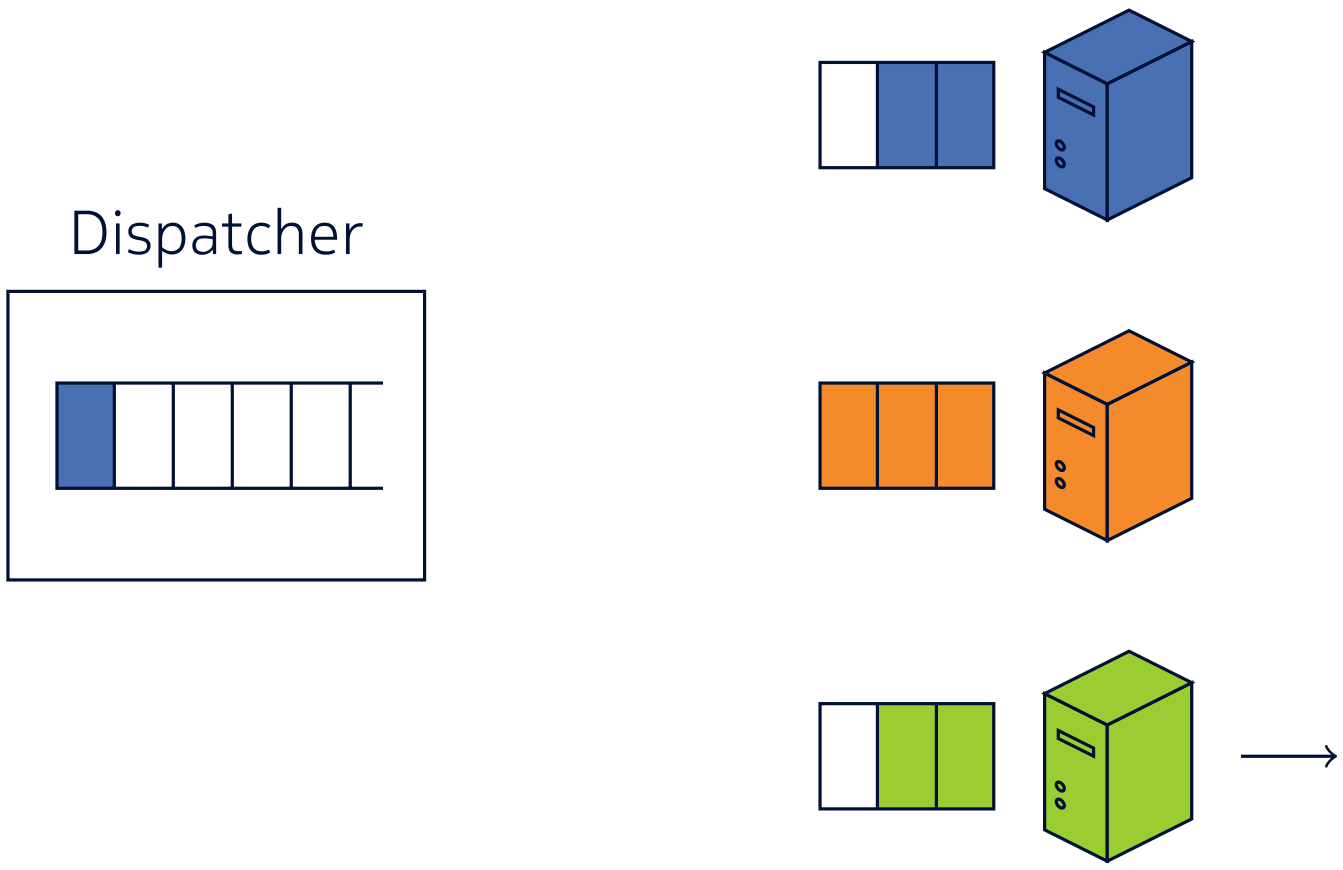
Algorithm



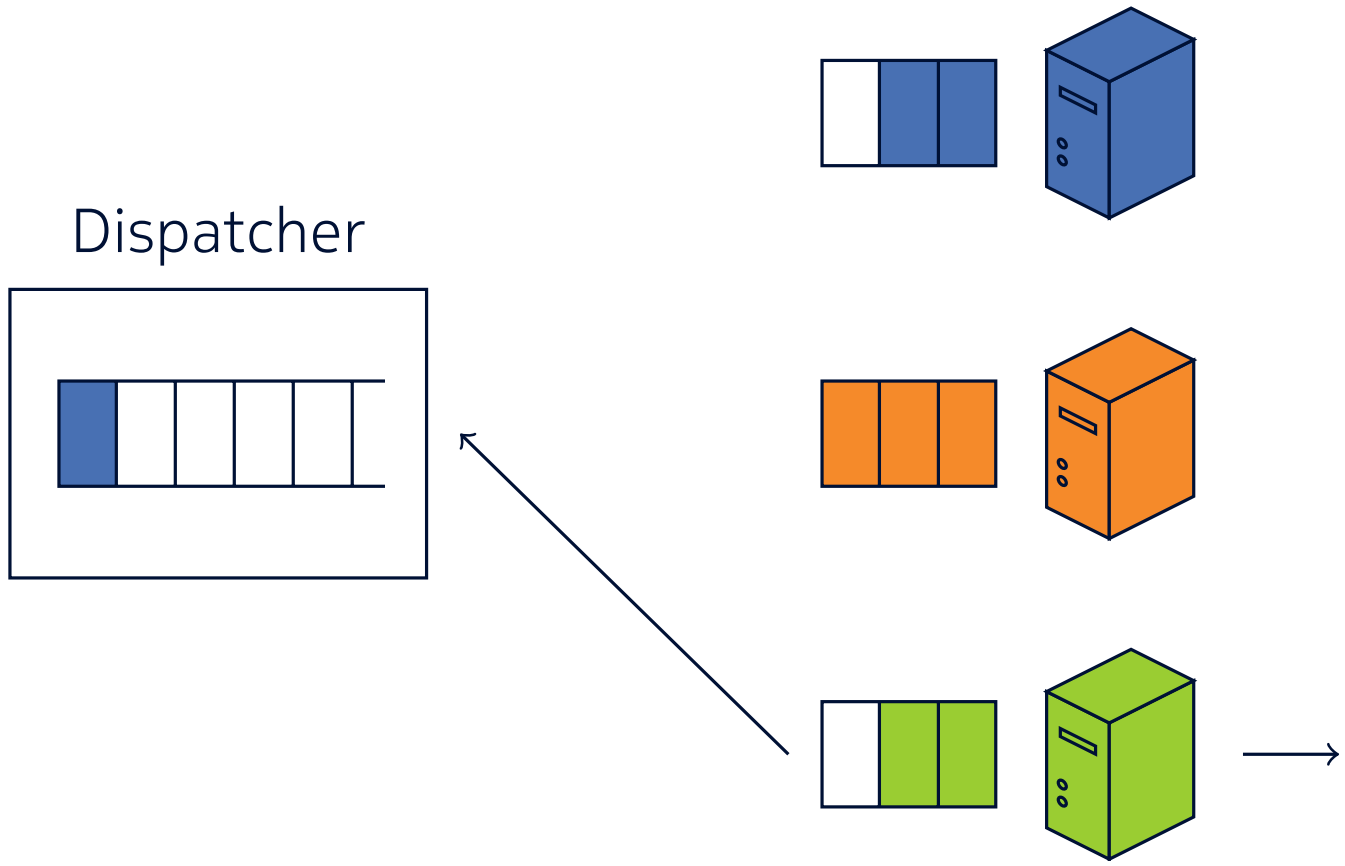
Algorithm



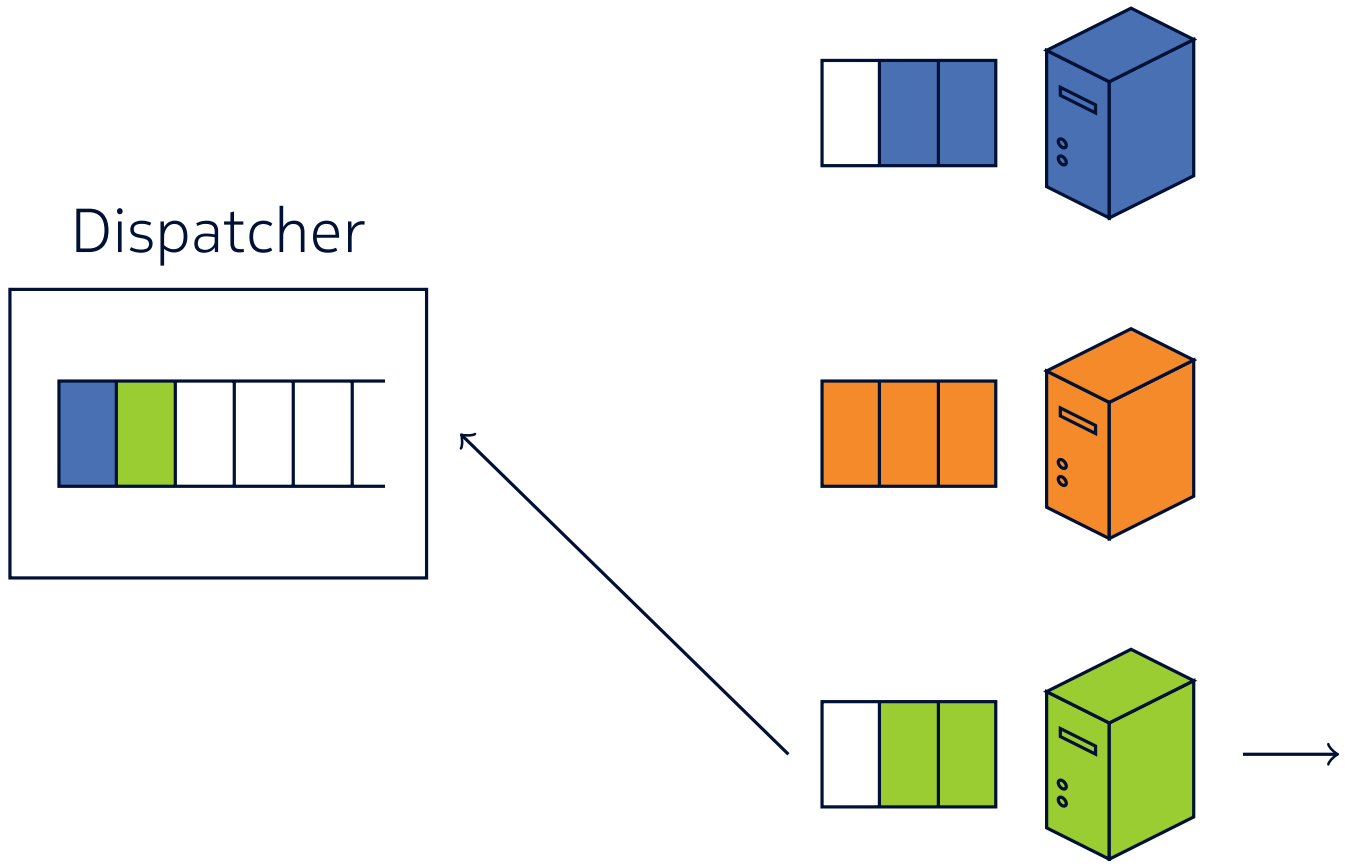
Algorithm



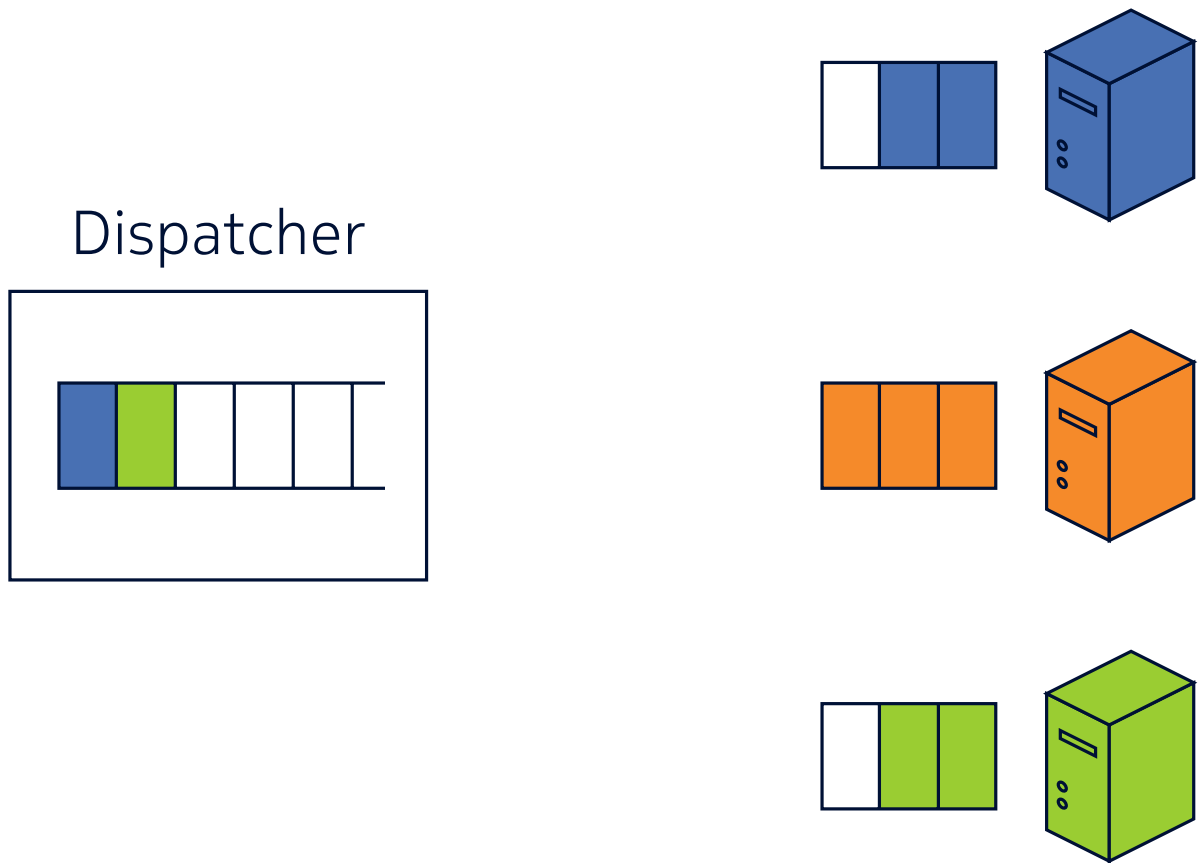
Algorithm



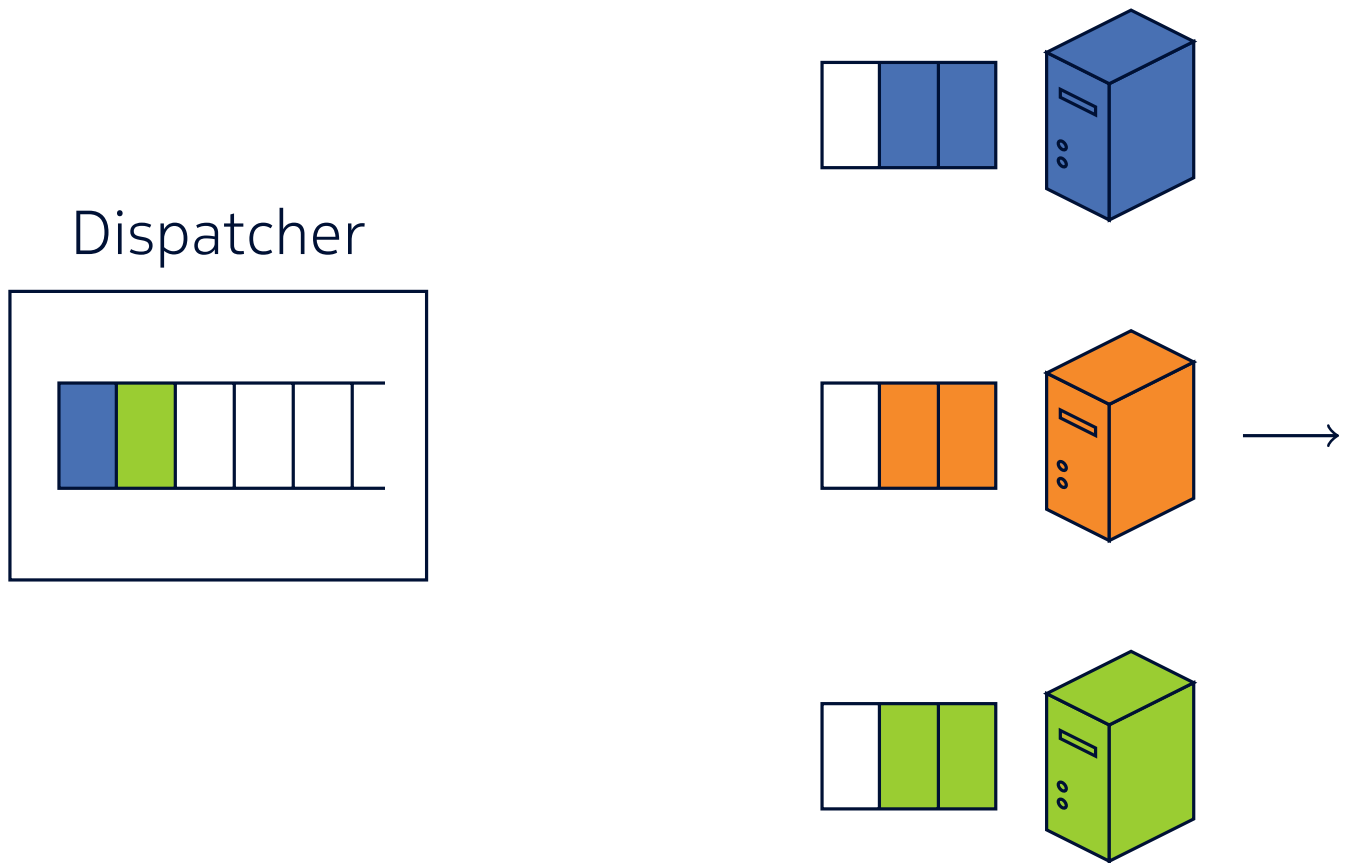
Algorithm



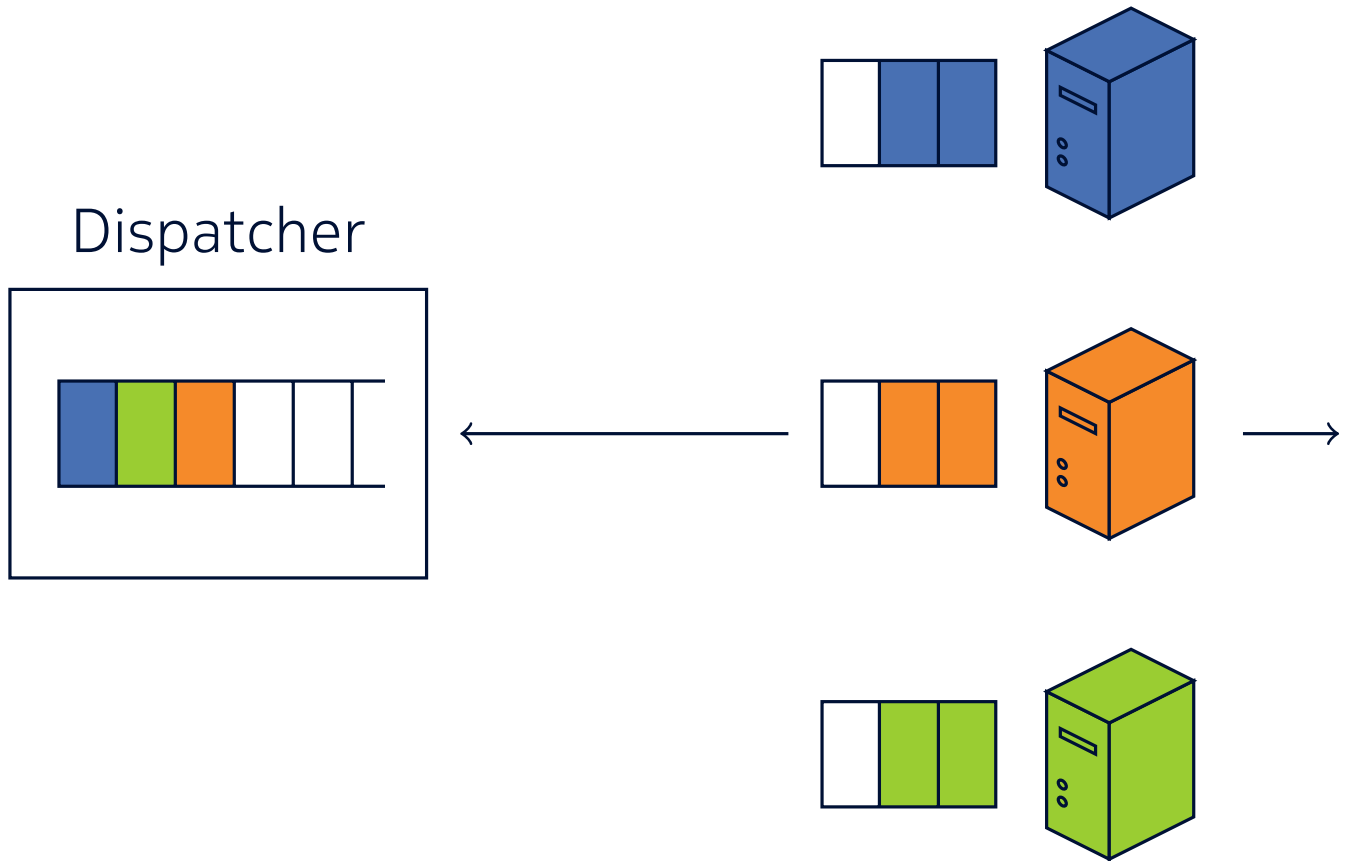
Algorithm



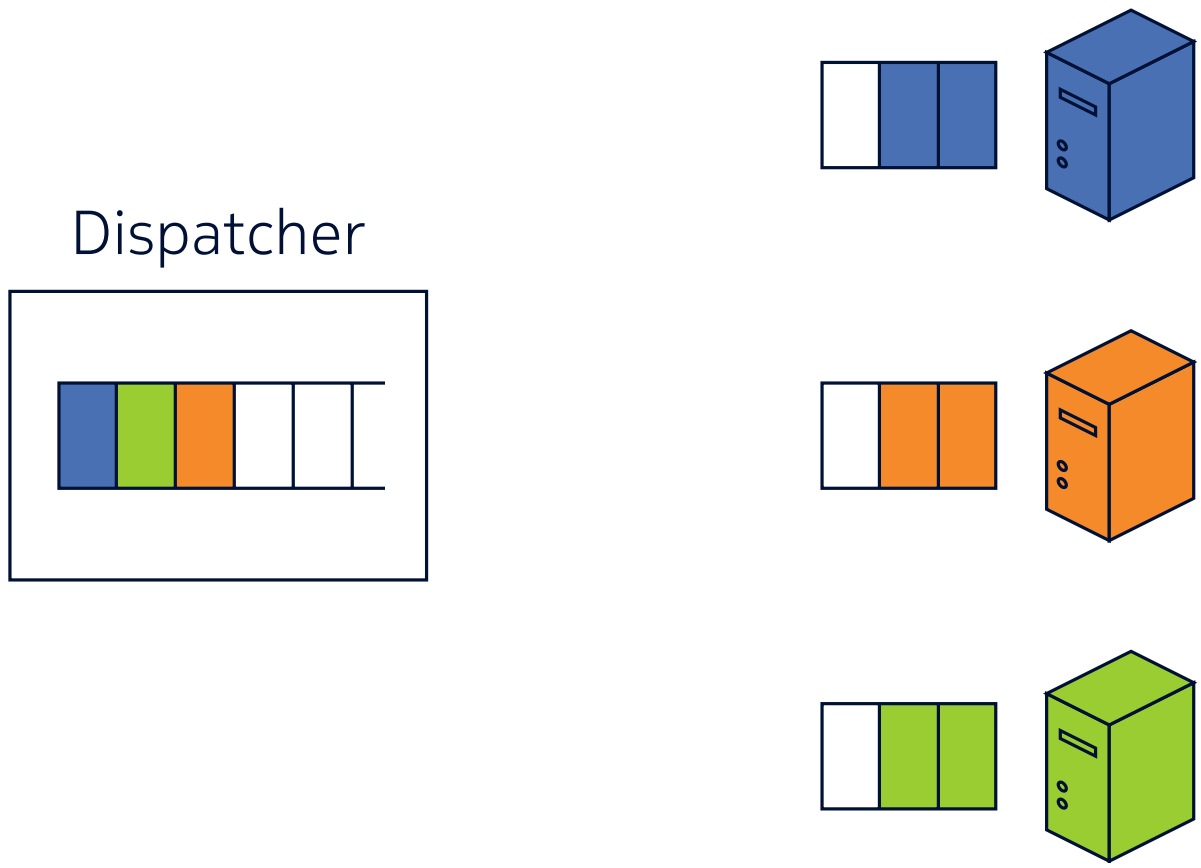
Algorithm



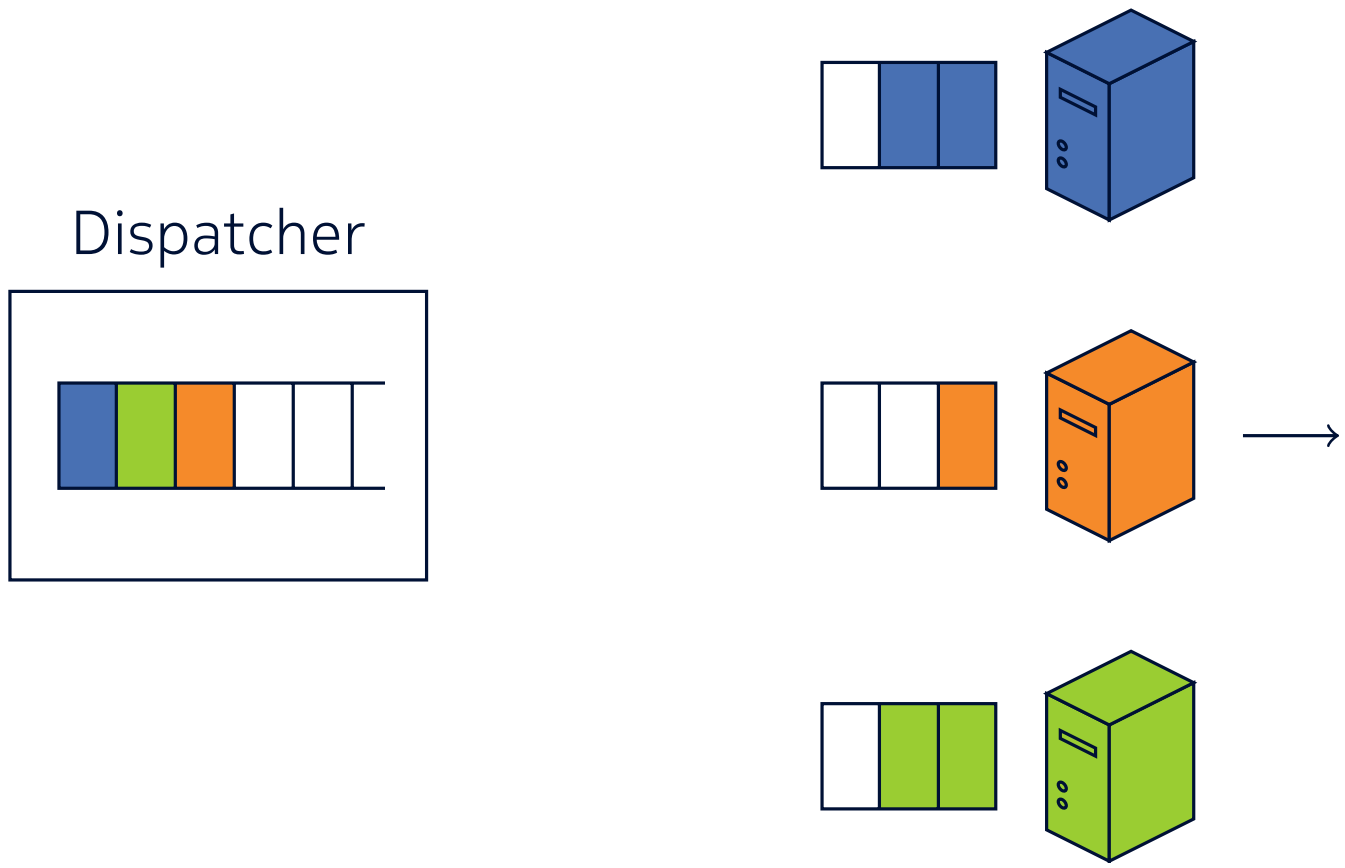
Algorithm



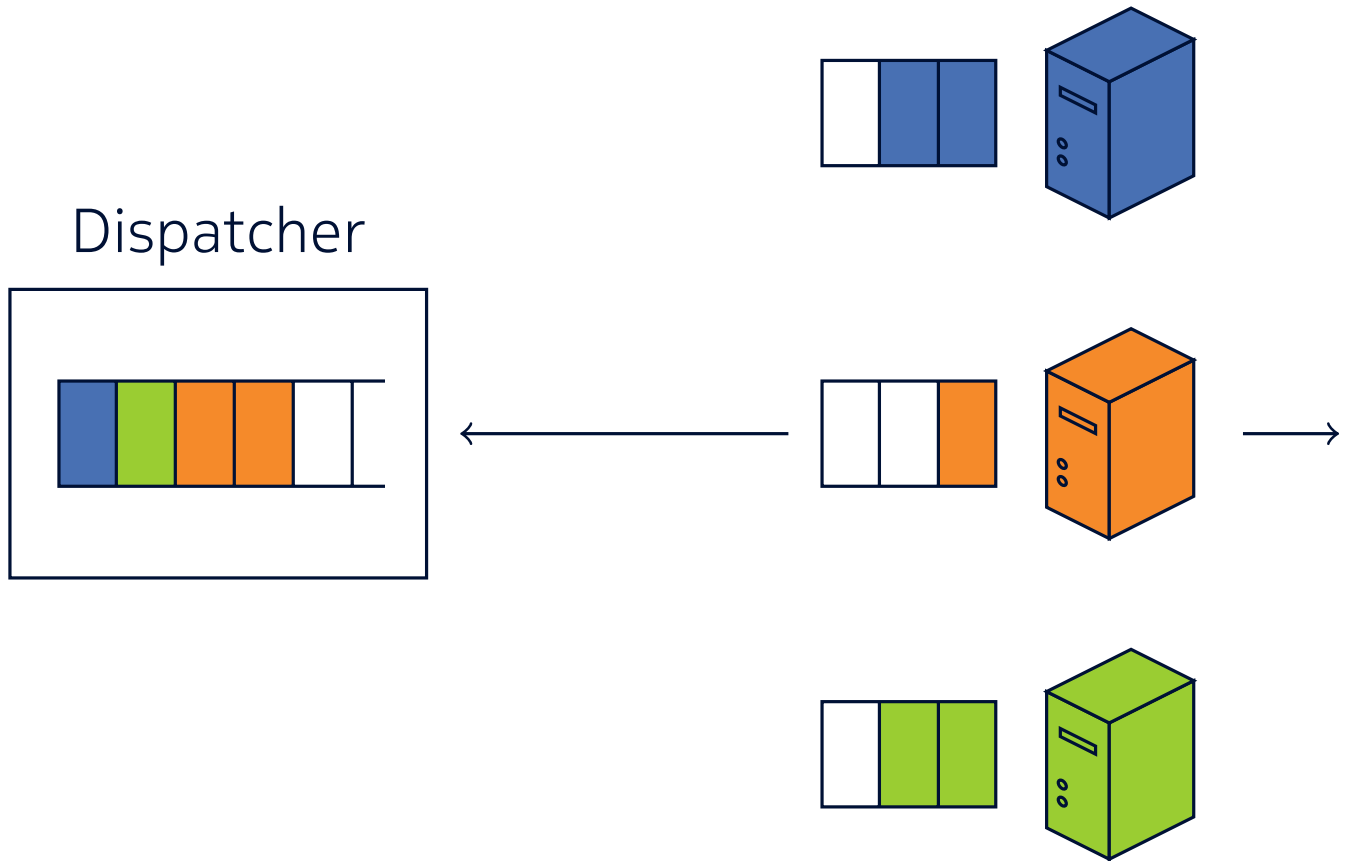
Algorithm



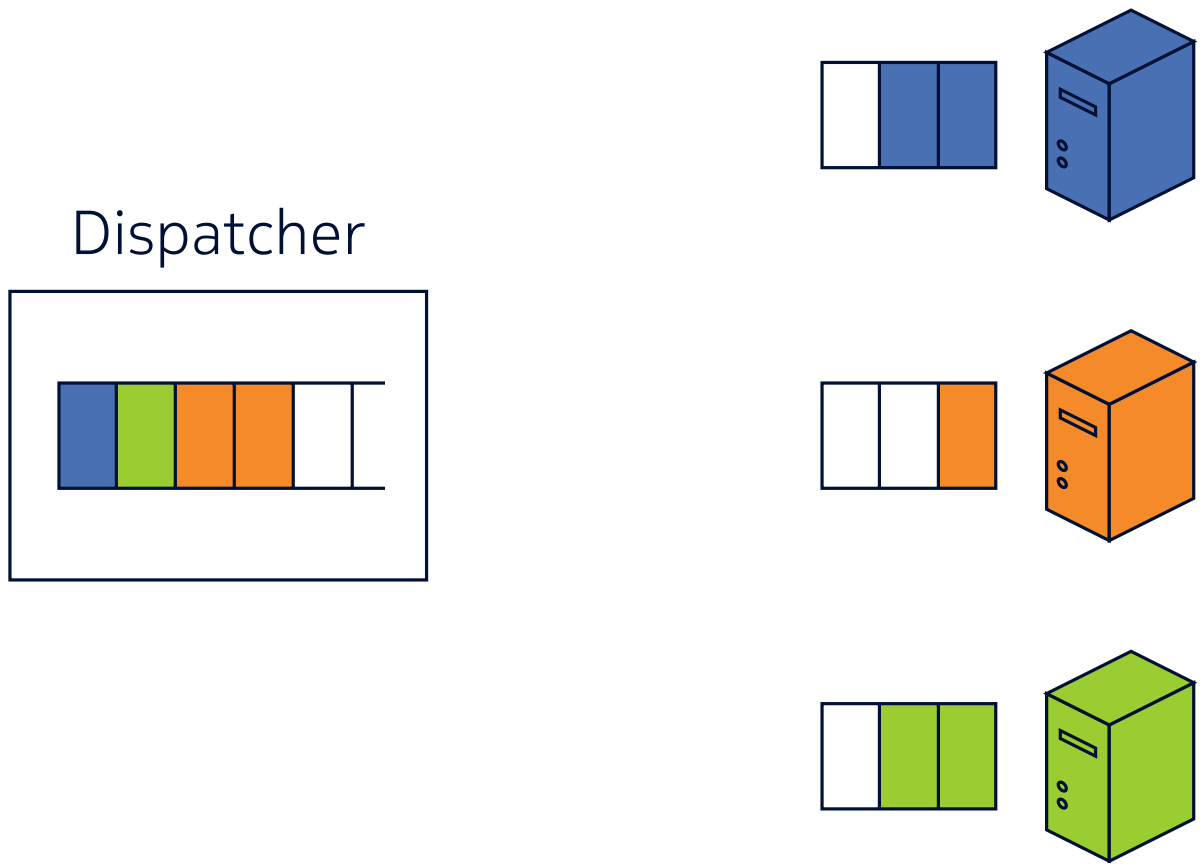
Algorithm



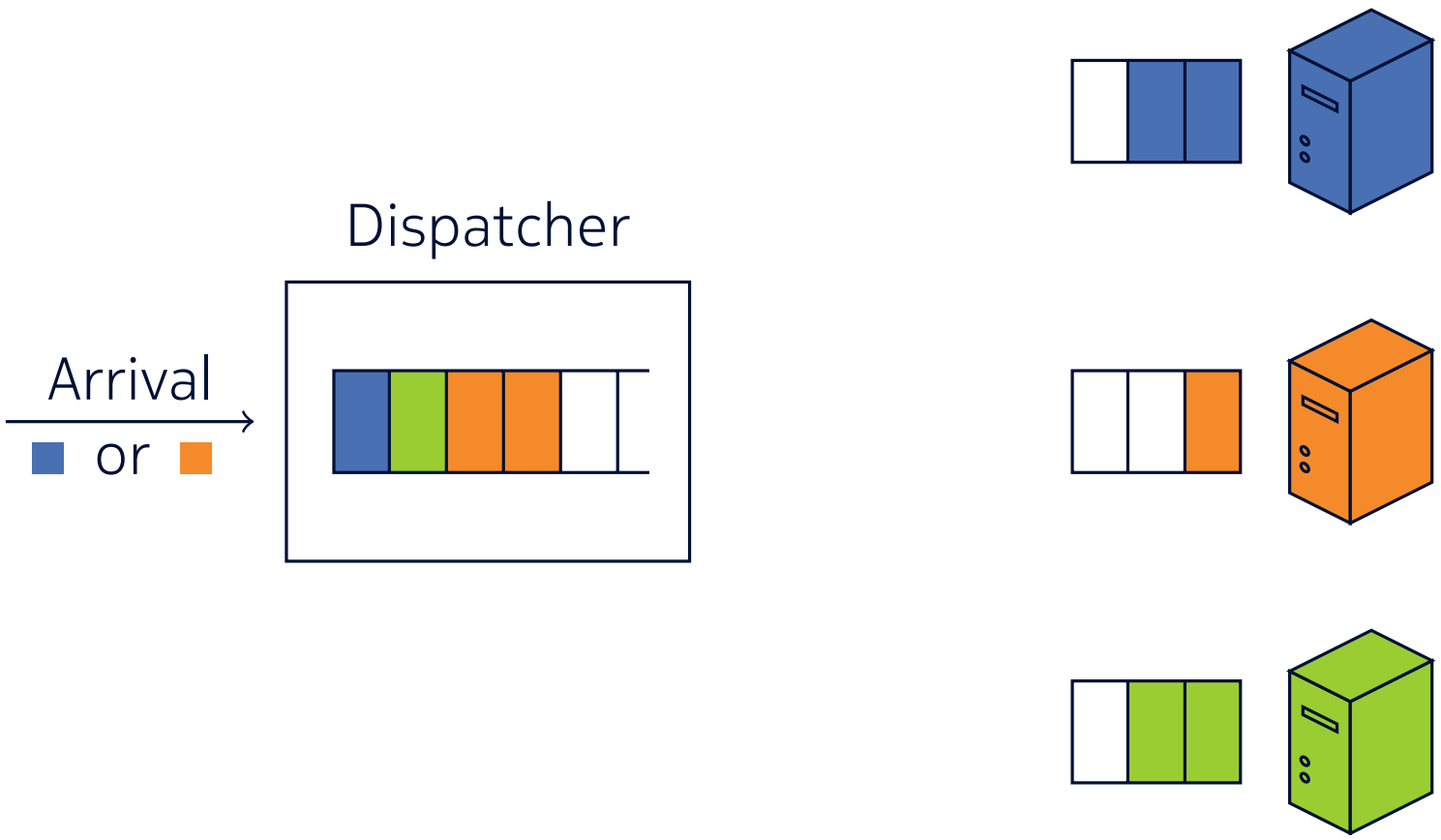
Algorithm



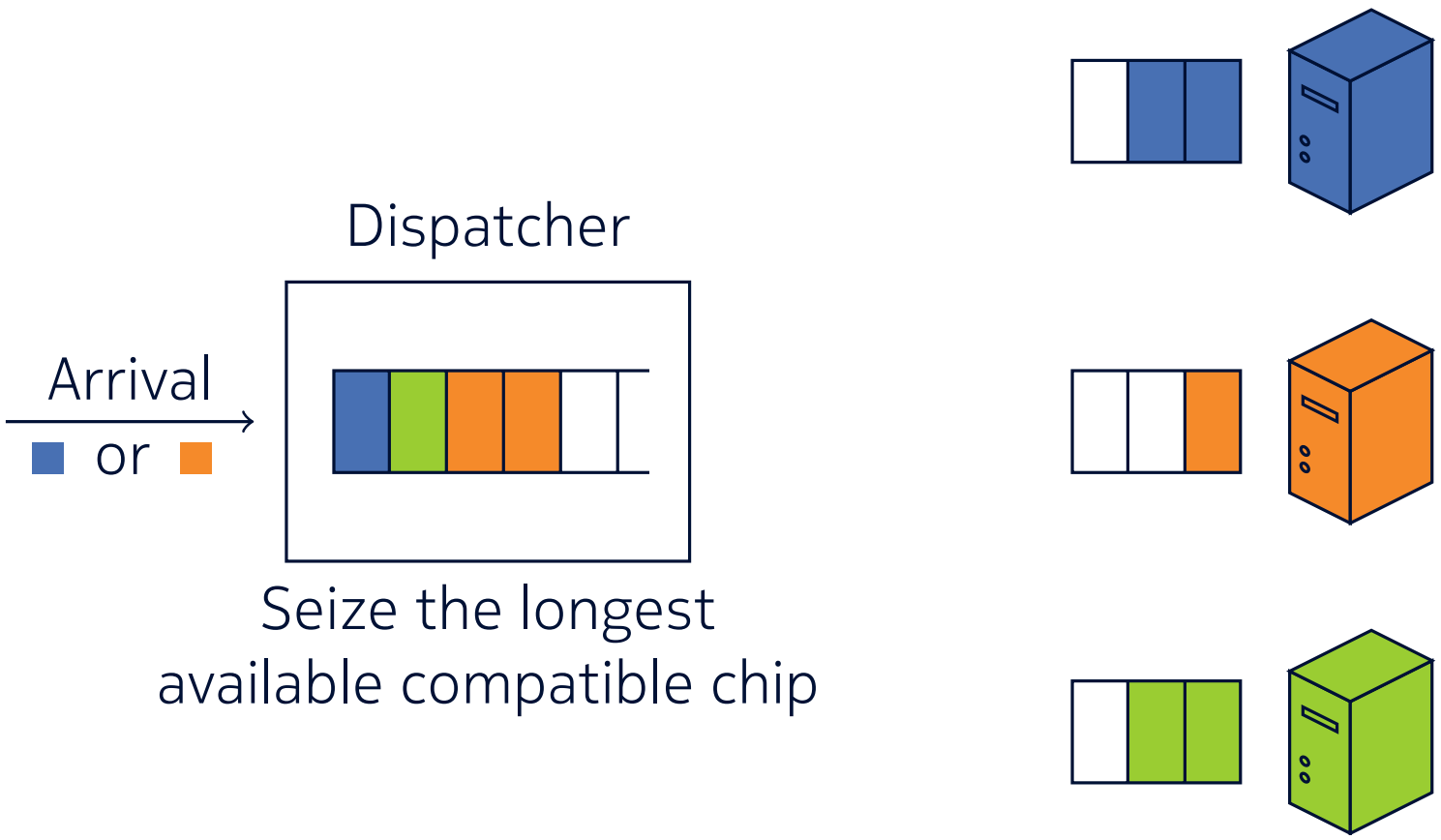
Algorithm



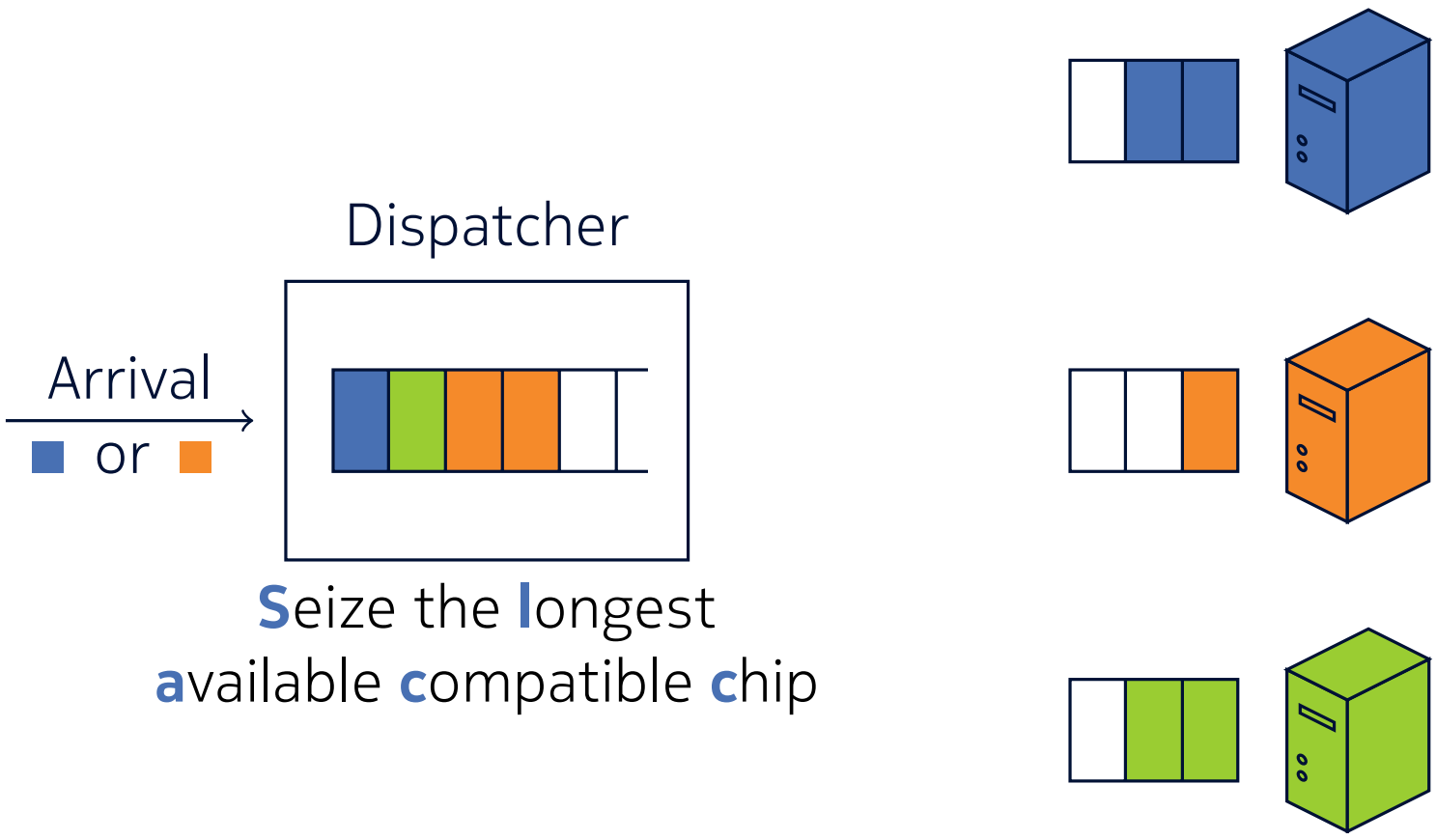
Algorithm



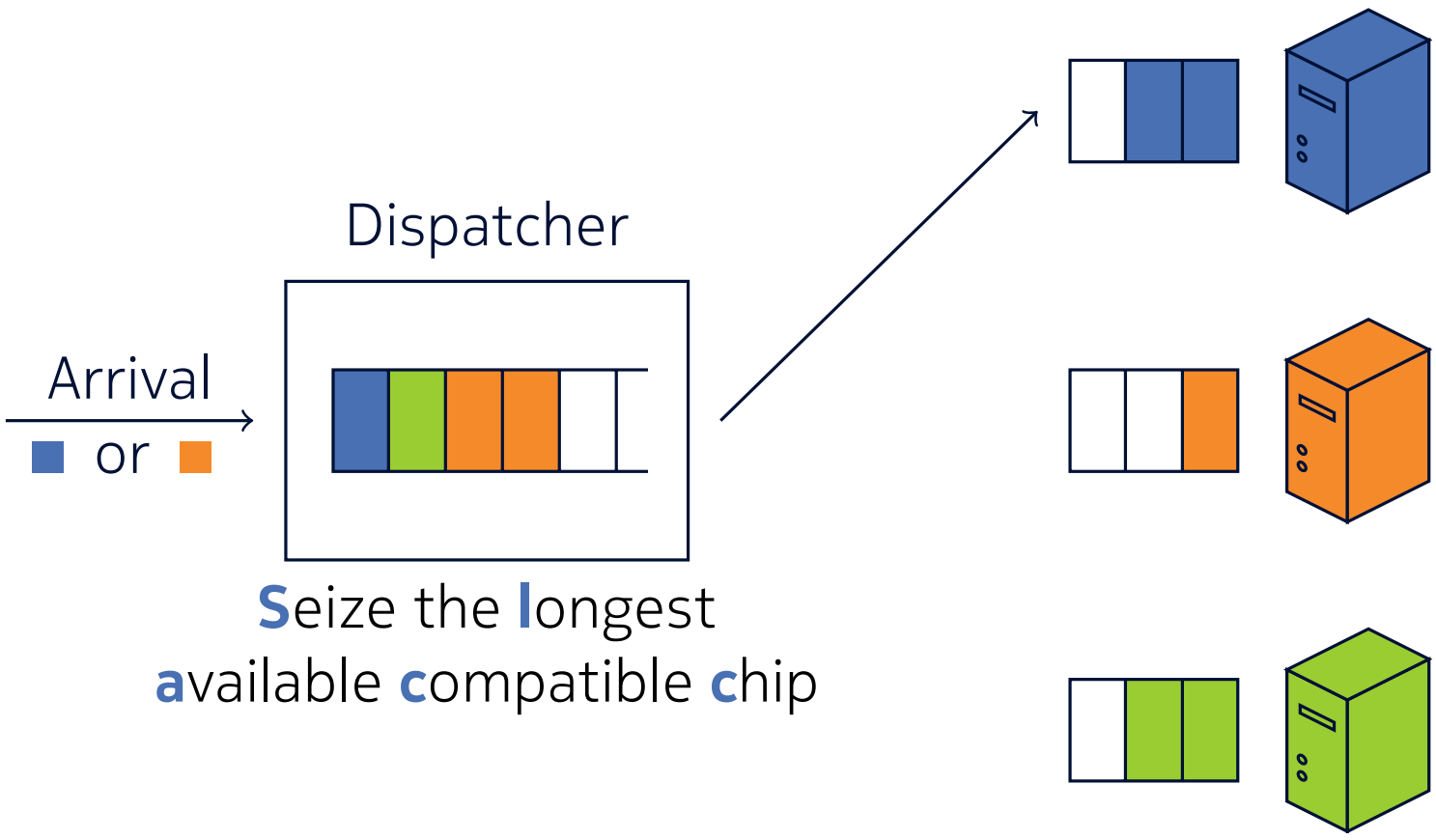
Algorithm



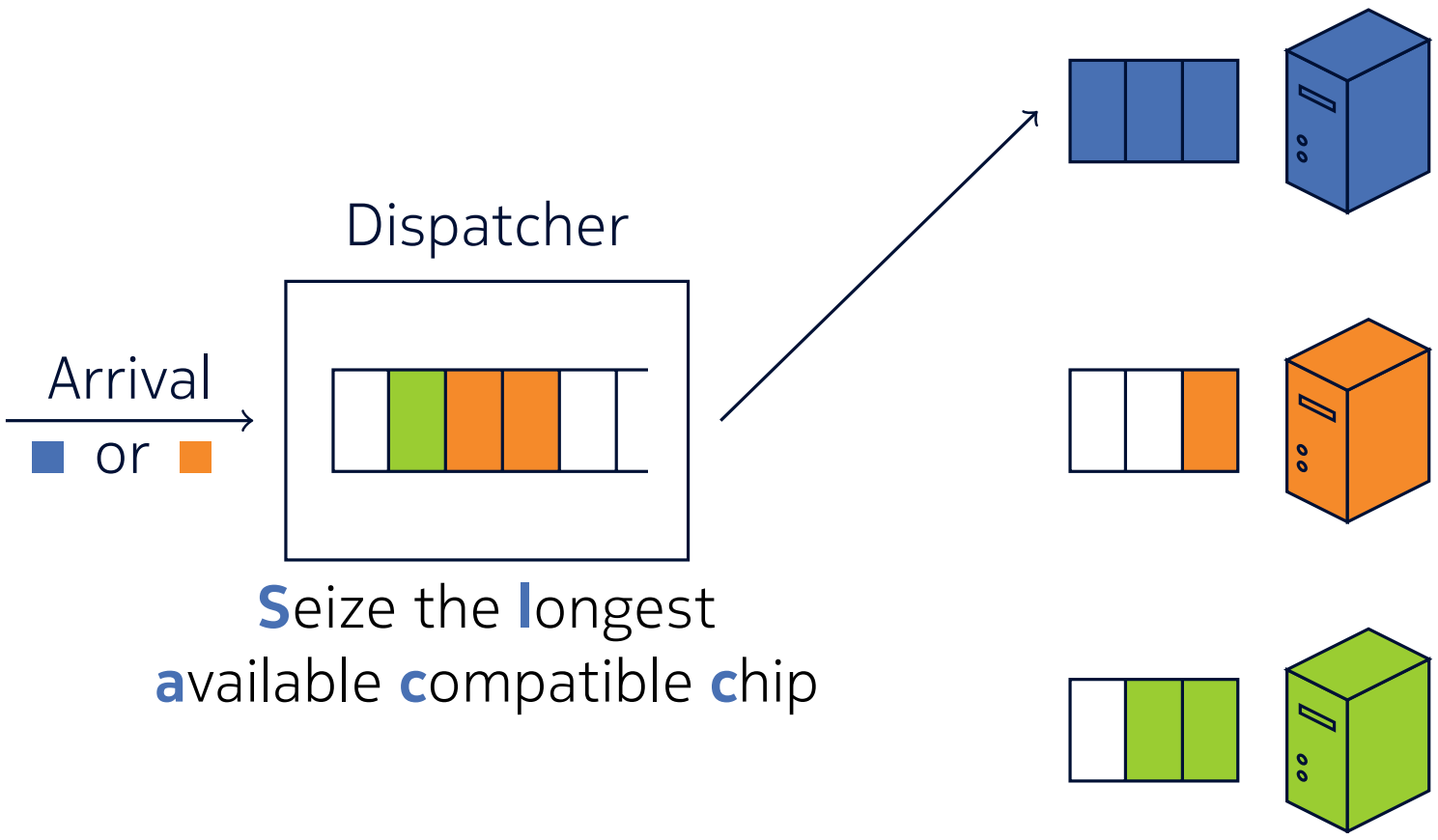
Algorithm



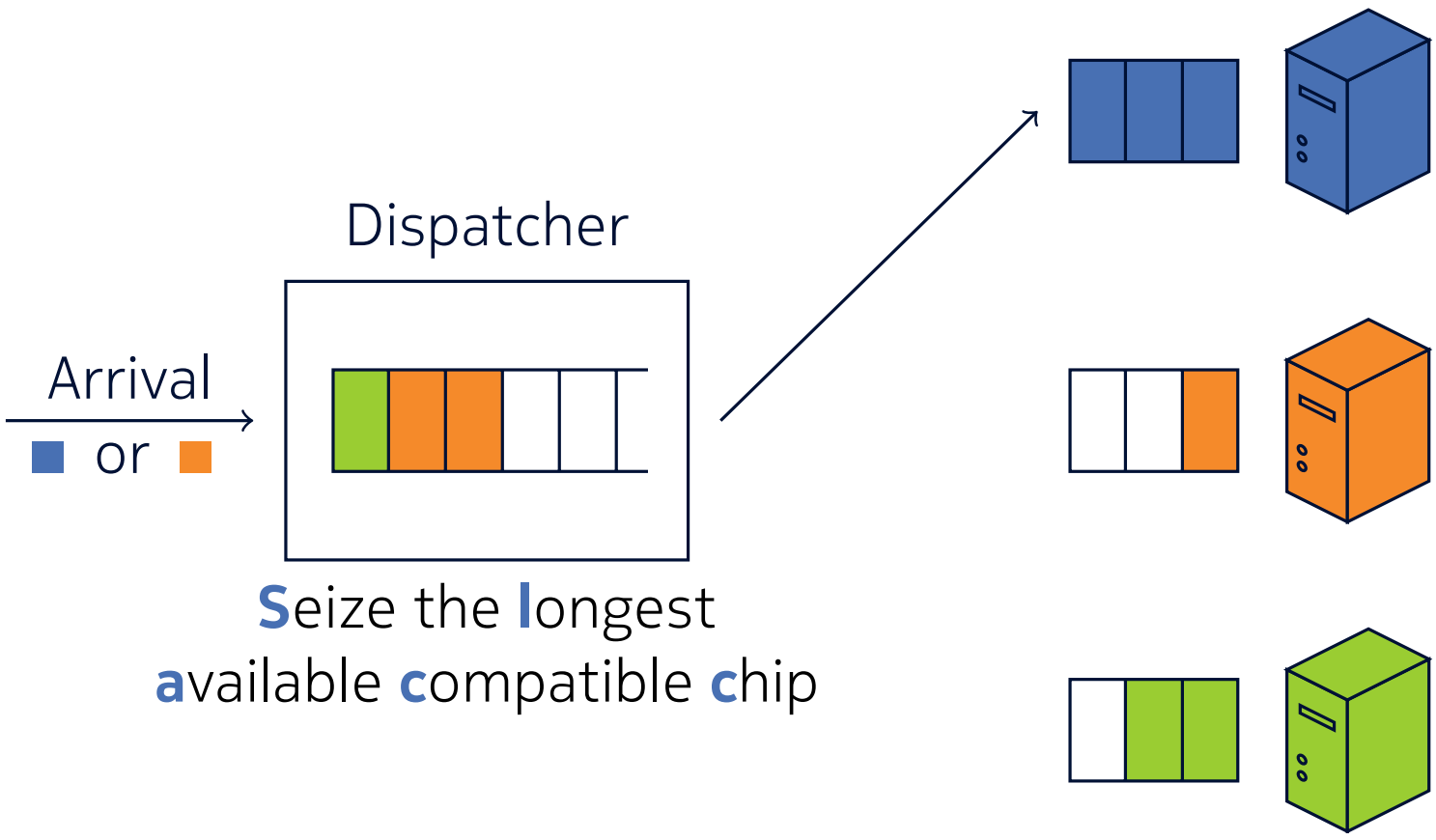
Algorithm



Algorithm

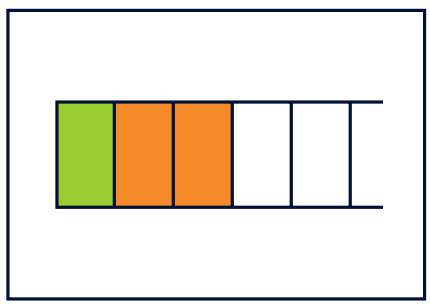


Algorithm

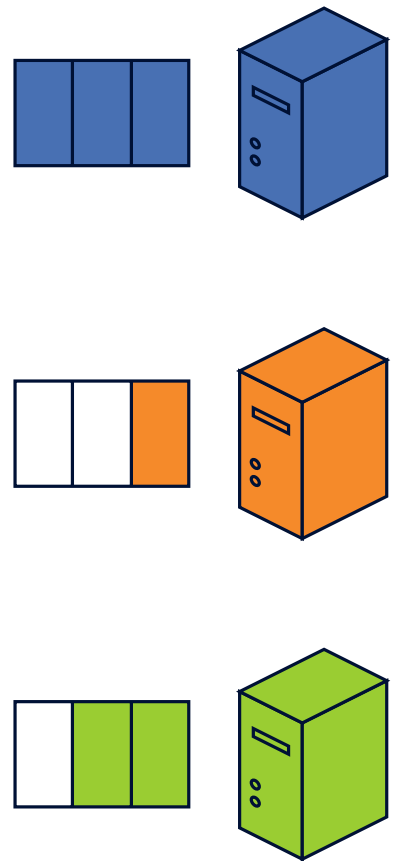


Algorithm

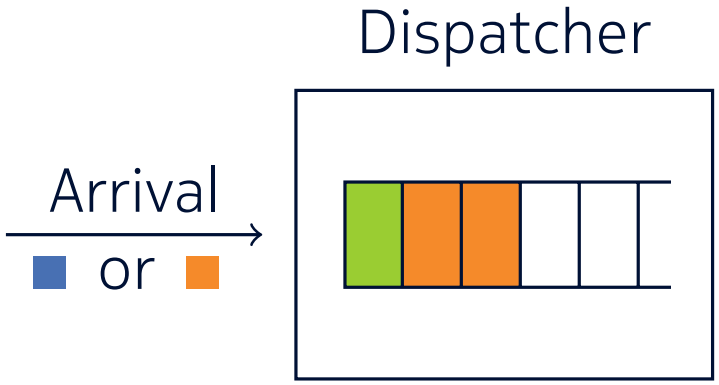
Dispatcher



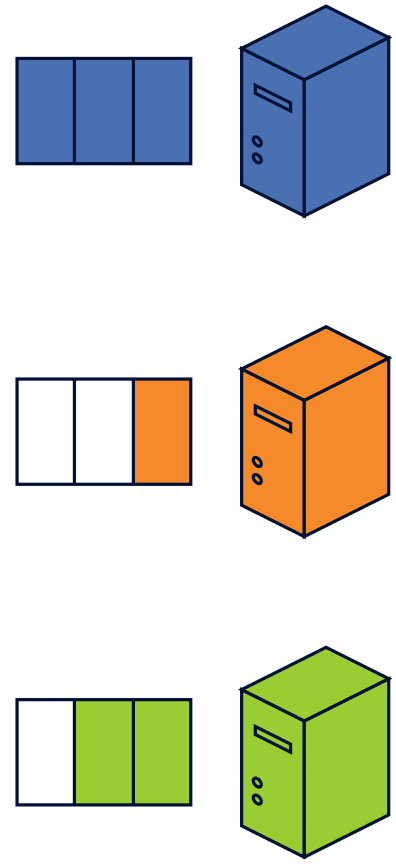
Seize the **l**ongest
available **c**ompatible **c**hip



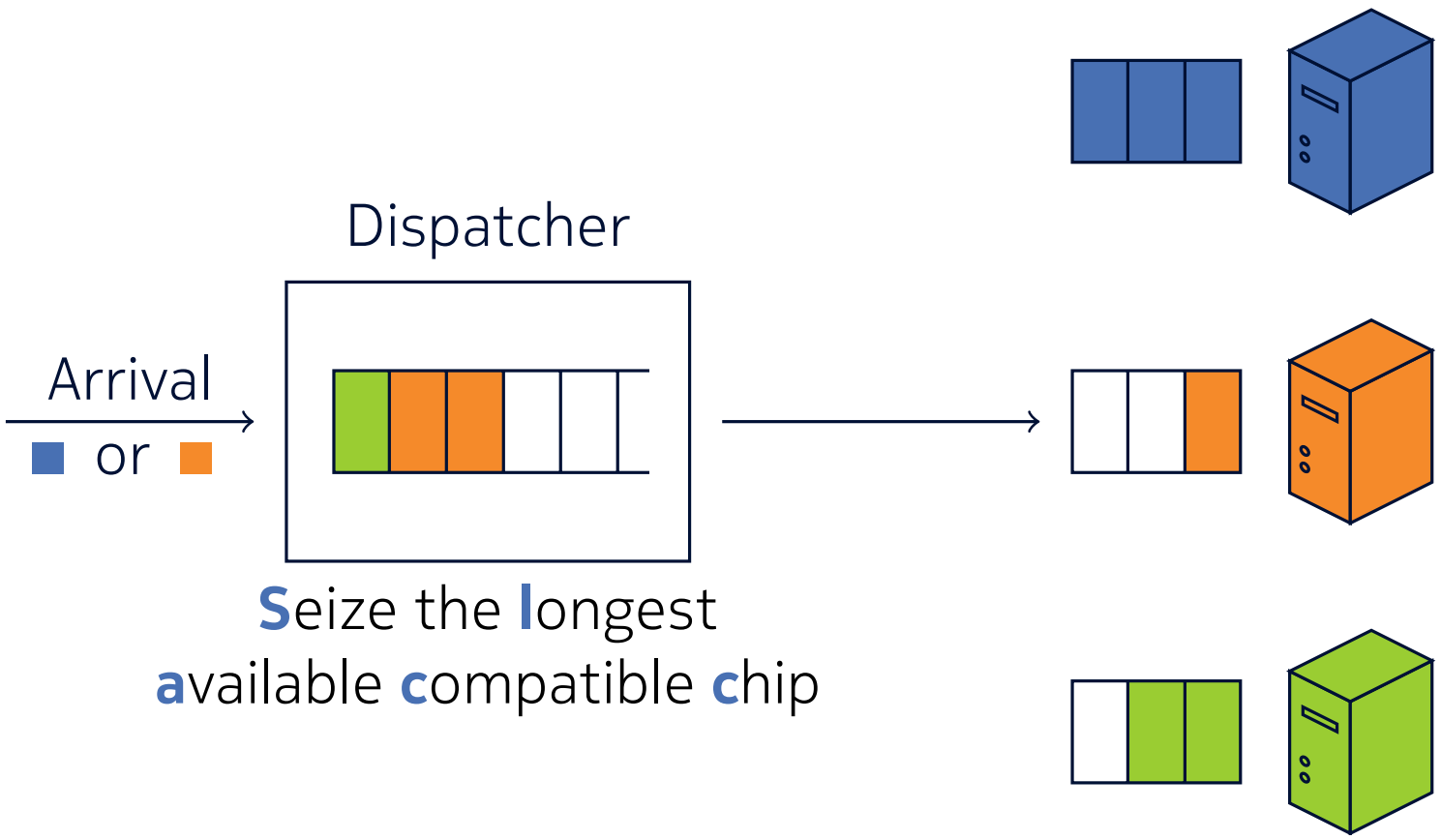
Algorithm



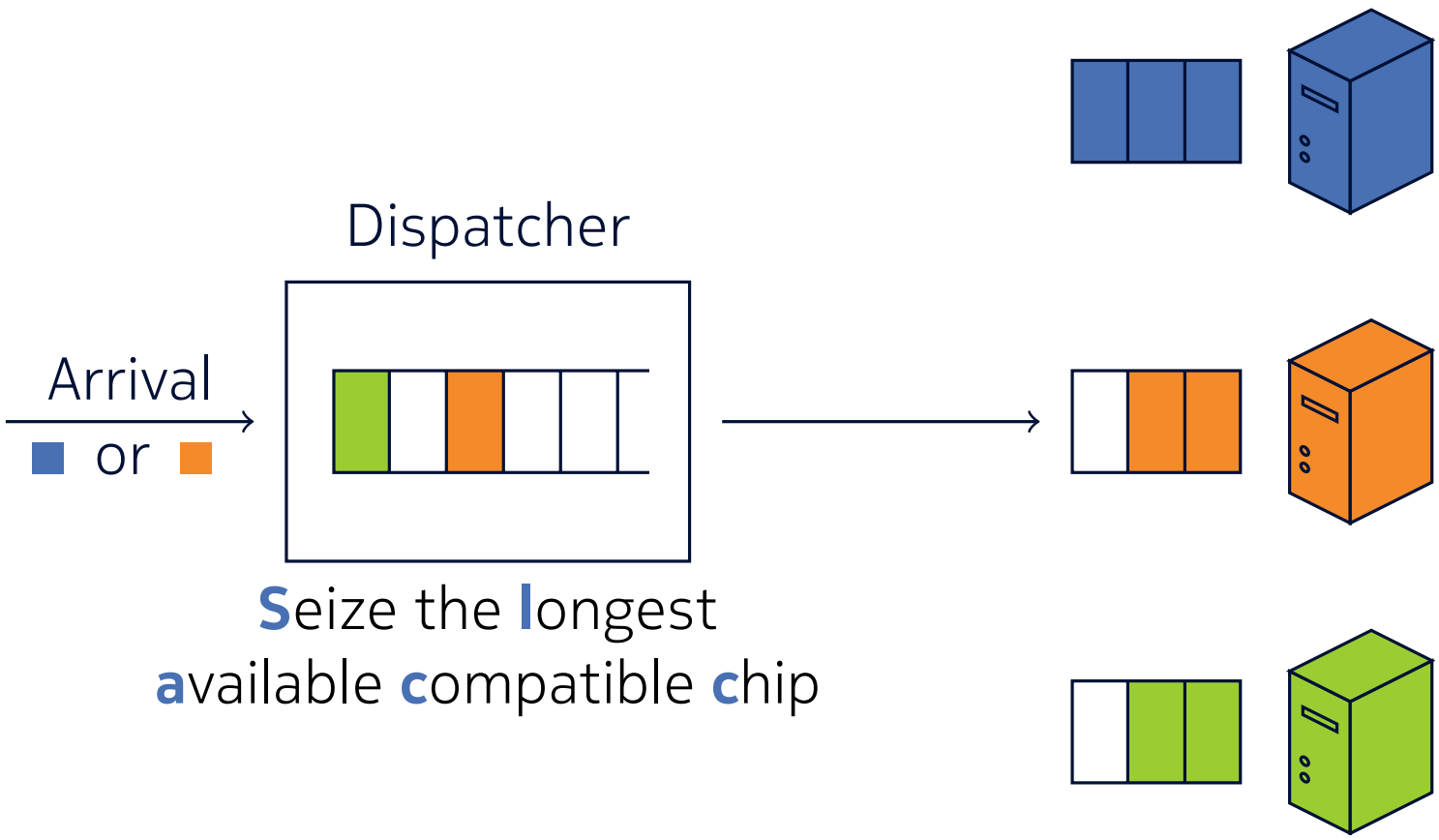
Seize the **l**ongest
available **c**ompatible **c**hip



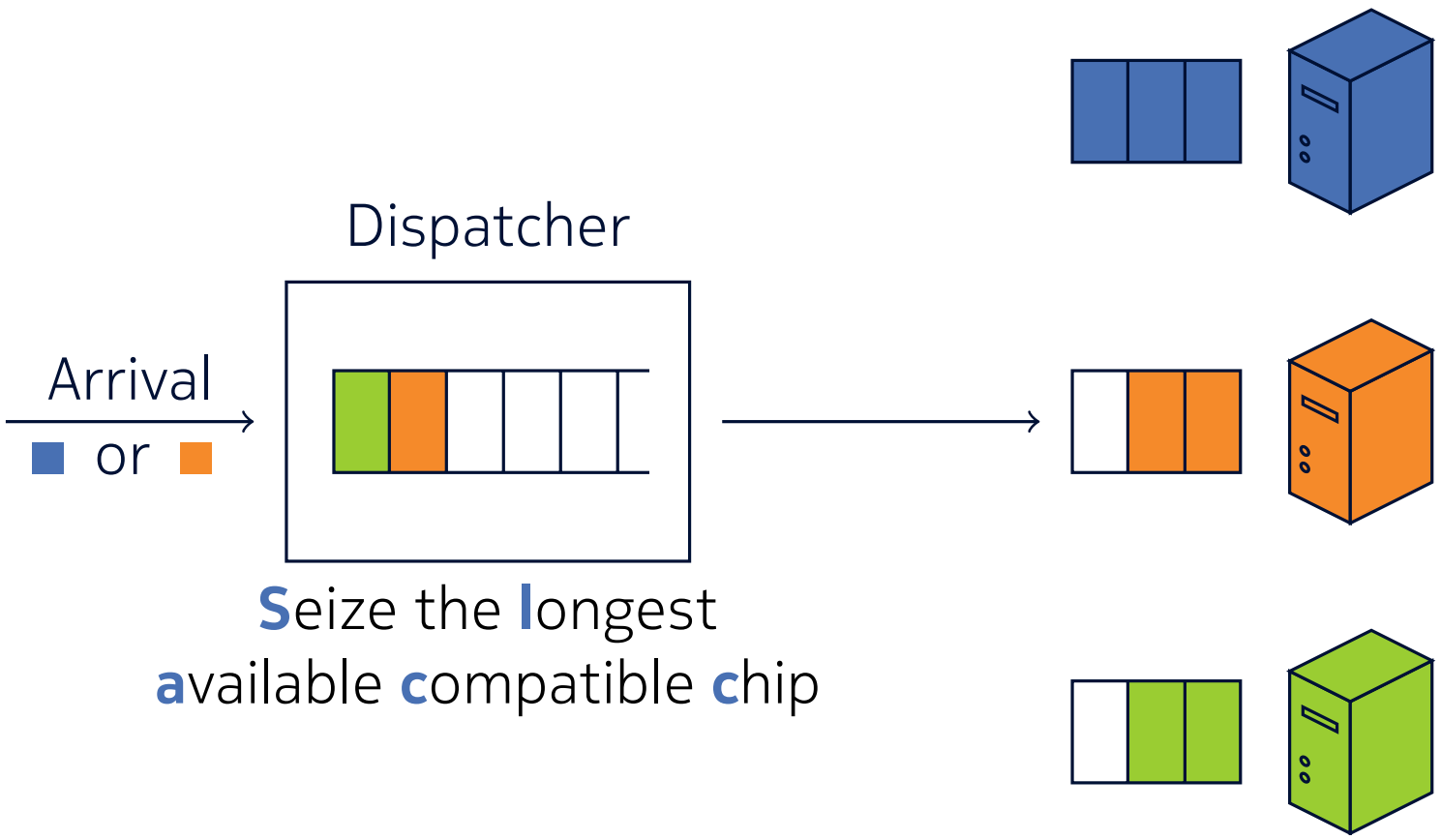
Algorithm



Algorithm

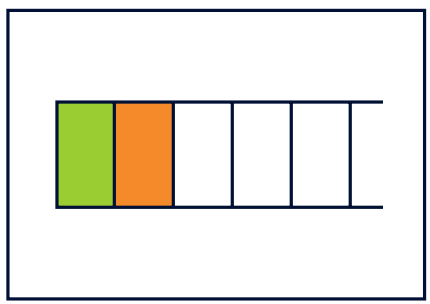


Algorithm

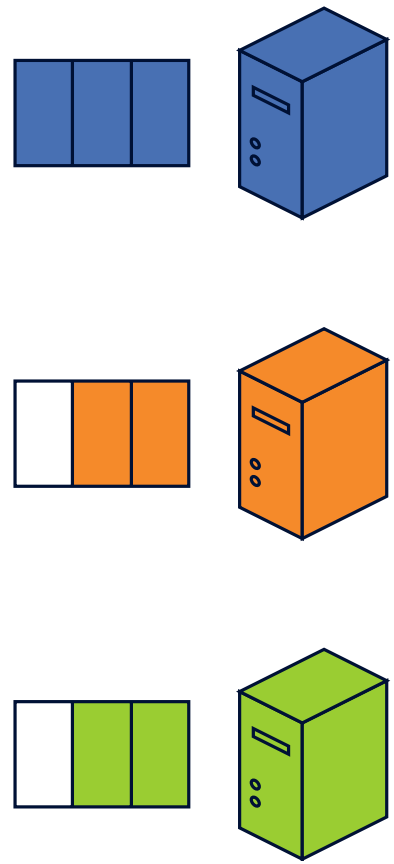


Algorithm

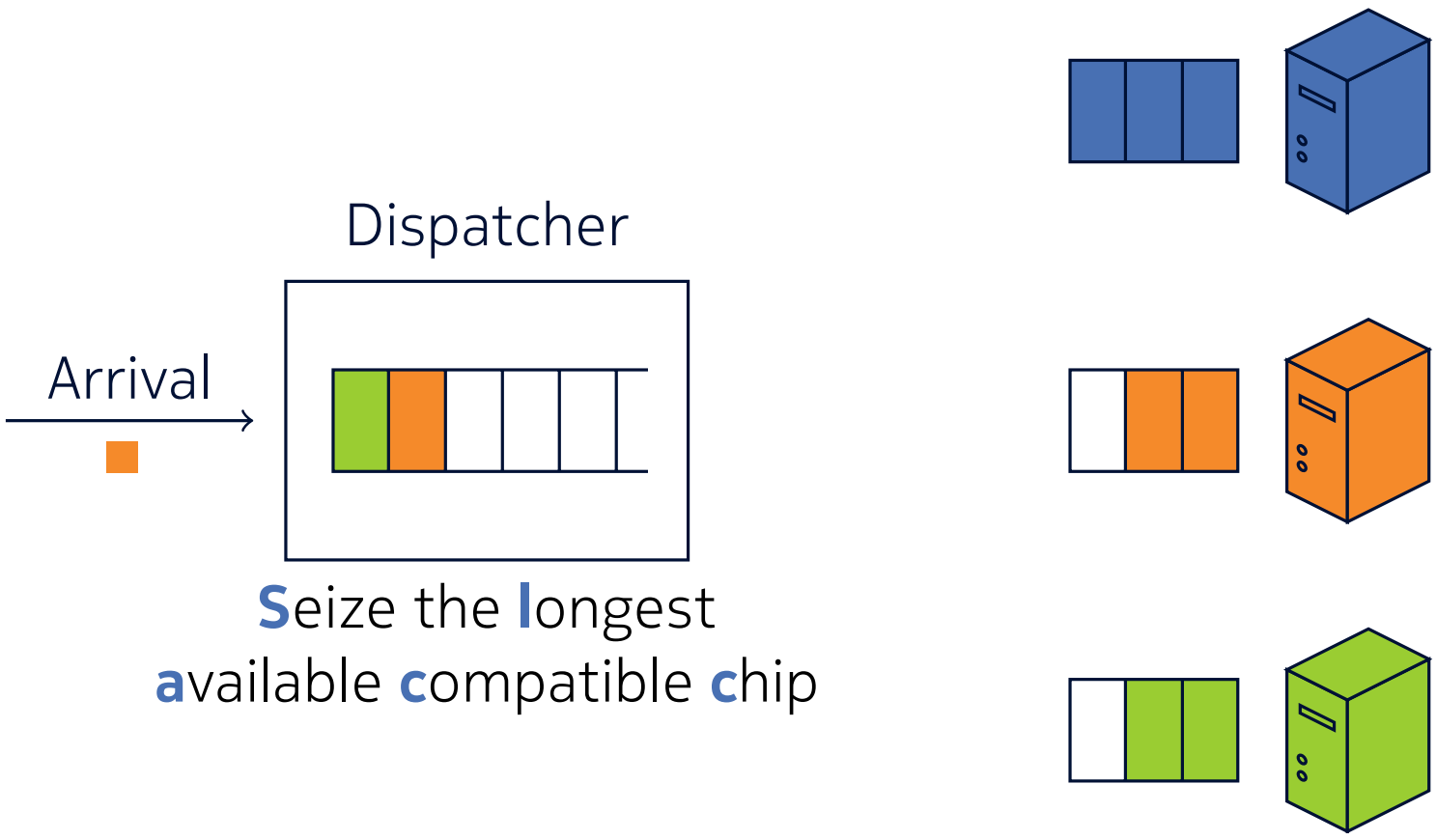
Dispatcher



Seize the **l**ongest
available **c**ompatible **c**hip

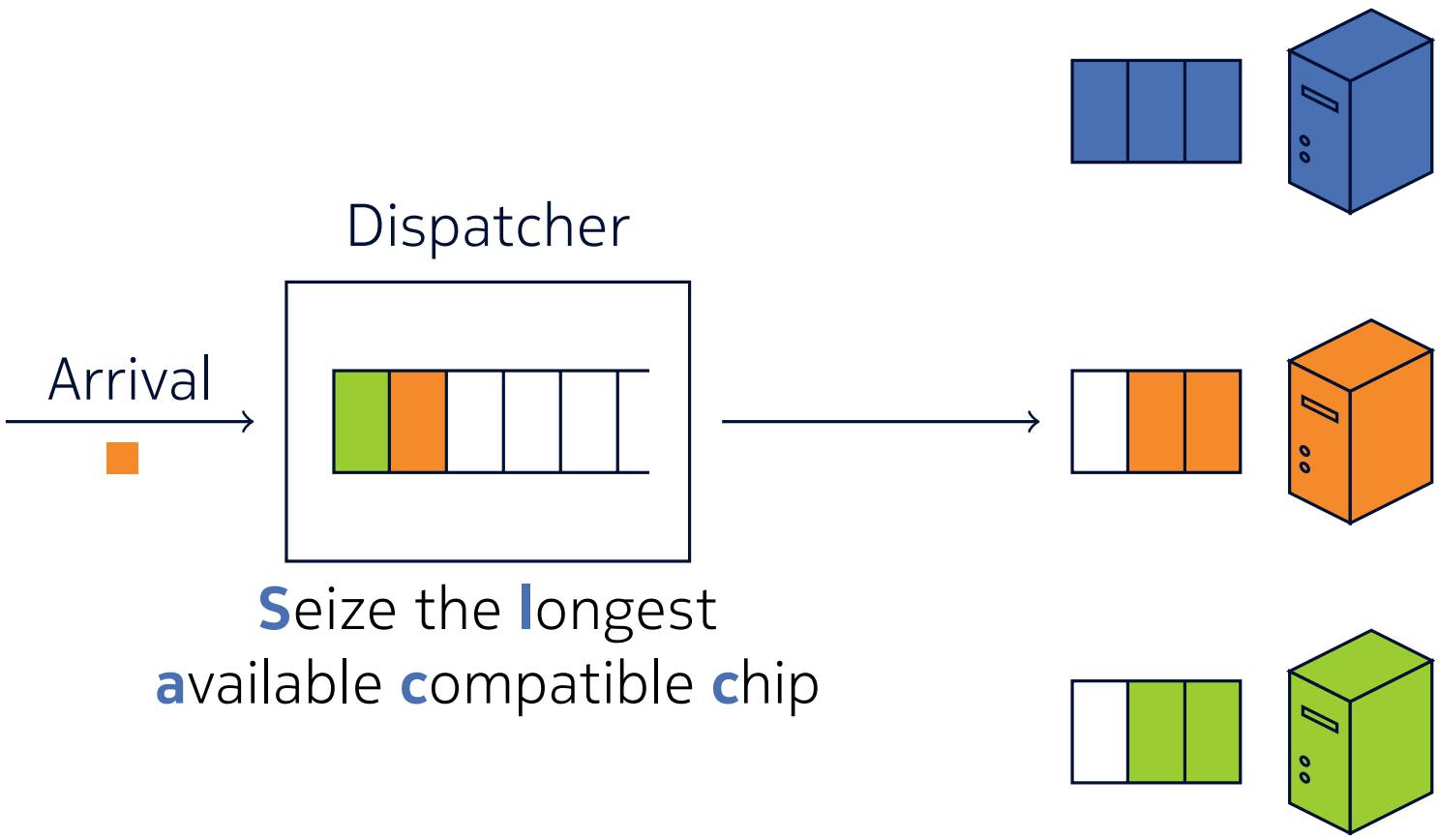


Algorithm

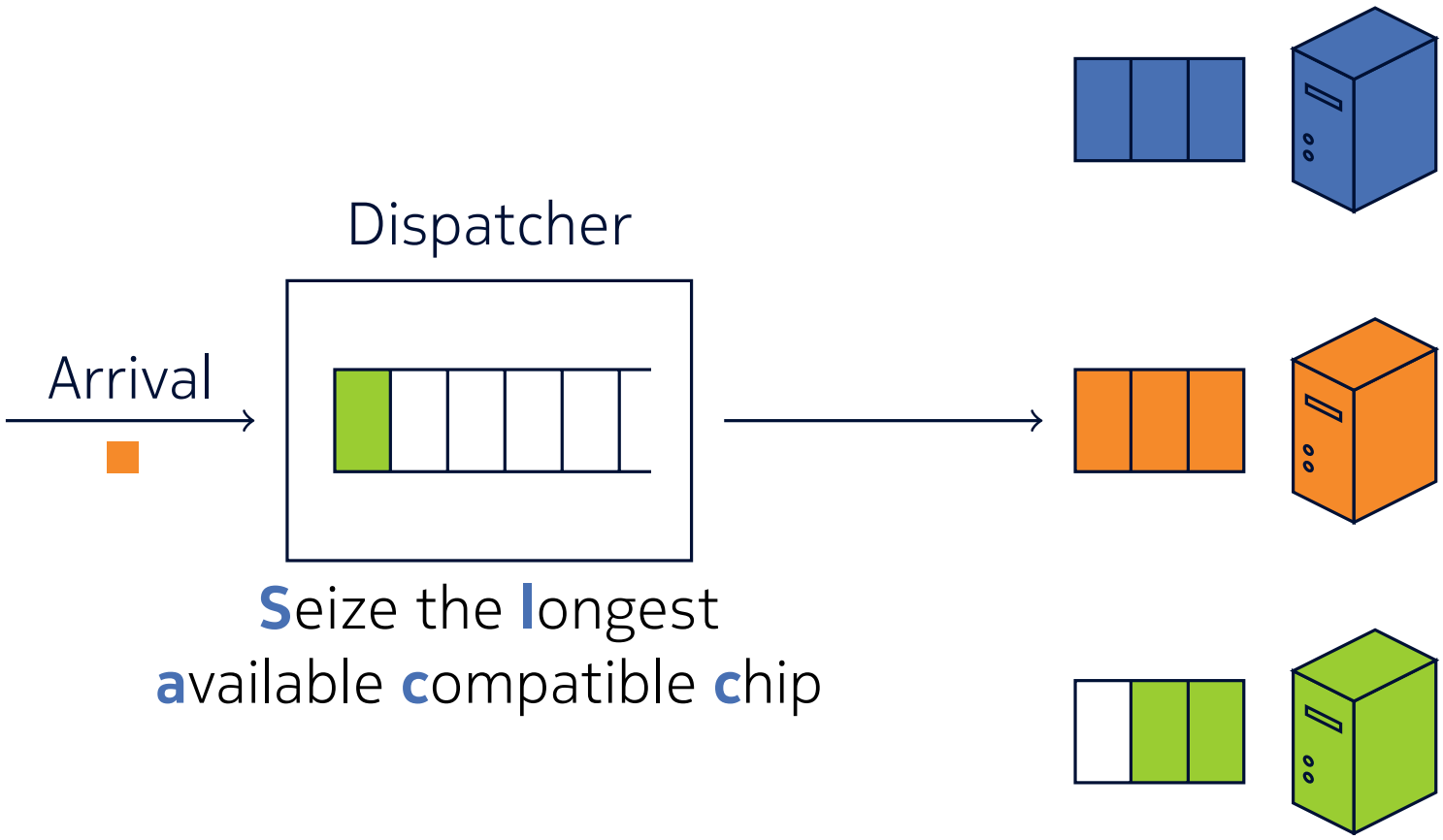


Seize the **l**ongest
available **c**ompatible **c**hip

Algorithm

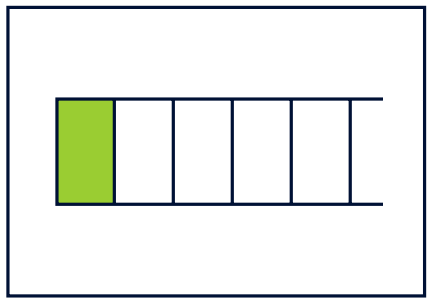


Algorithm

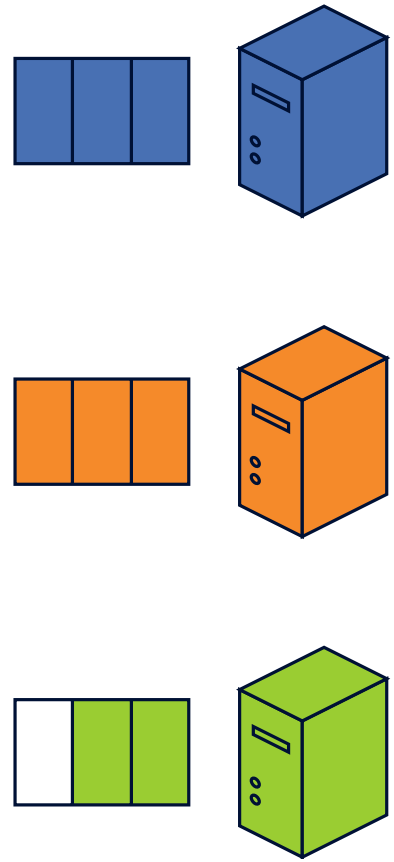


Algorithm

Dispatcher



Seize the **l**ongest
available **c**ompatible **c**hip



Intuition

Release order of the tokens ≃ Relative load of the servers

A token that has been available for a long time is likely to identify a server that is less loaded than others

Intuition

Release order of the tokens ≃ Relative load of the servers

A token that has been available for a long time is likely to identify a server that is less loaded than others

→ Better to seize the longest available compatible token

Intuition

Release order of the tokens ≃ Relative load of the servers

A token that has been available for a long time is likely to identify a server that is less loaded than others

→ Better to seize the longest available compatible token

Number of tokens ≃ Freshness of the information

Intuition

Release order of the tokens ≃ Relative load of the servers

A token that has been available for a long time is likely to identify a server that is less loaded than others

→ Better to seize the longest available compatible token

Number of tokens ≃ Freshness of the information

→ When the number of tokens increases:

- ☺ Higher tolerance to momentary mistakes
- ☹ The information at the head of the queue is “less fresh”

Insensitivity to the job size distribution

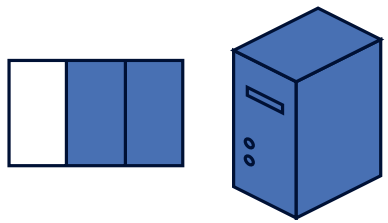
Under a Poisson arrival process

Insensitivity to the job size distribution

Under a Poisson arrival process

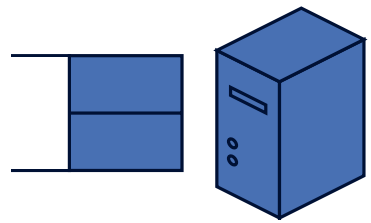
A single server:

First-come, first-served



Sensitive

Processor-sharing



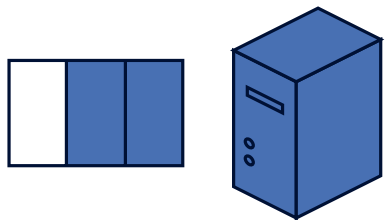
Insensitive

Insensitivity to the job size distribution

Under a Poisson arrival process

A single server:

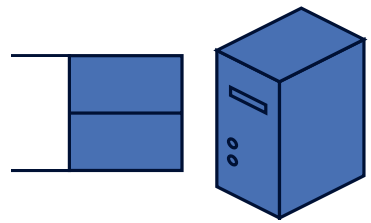
First-come, first-served



Sensitive

Round-robin

Processor-sharing

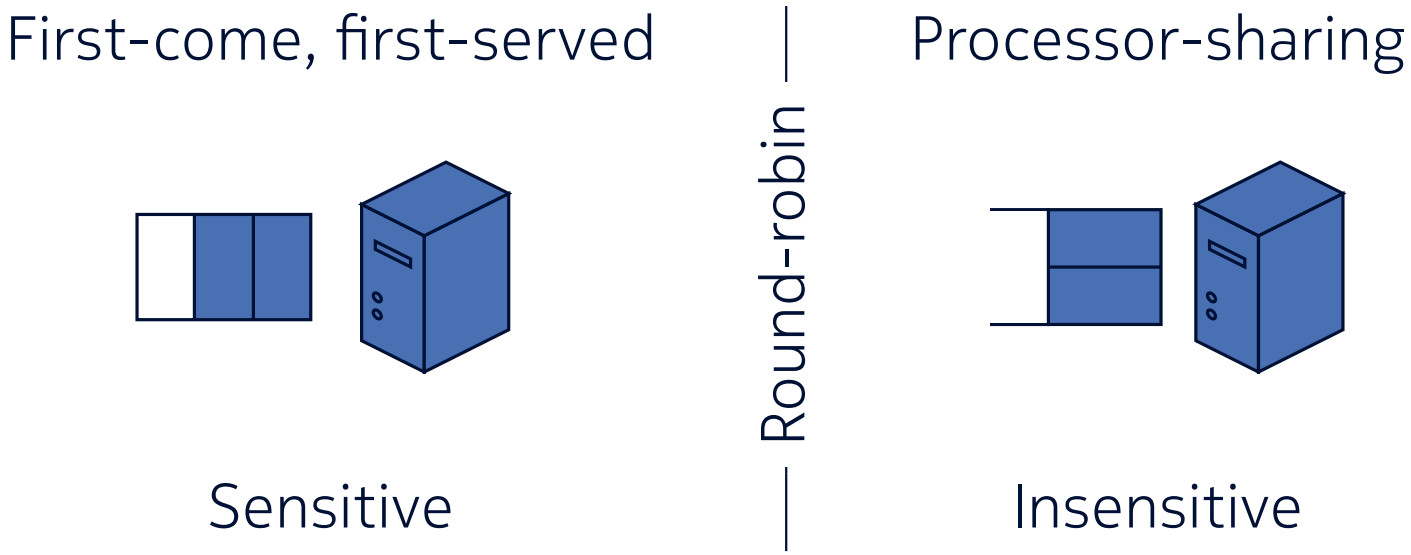


Insensitive

Insensitivity to the job size distribution

Under a Poisson arrival process

A single server:



Many servers: Our queueing model shows that the performance is insensitive if each server applies processor-sharing policy.

Outline

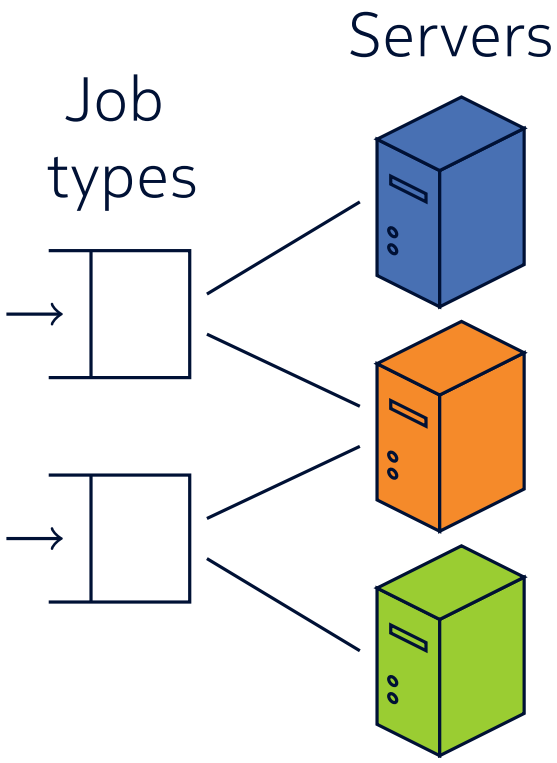
Algorithm

Queueing analysis

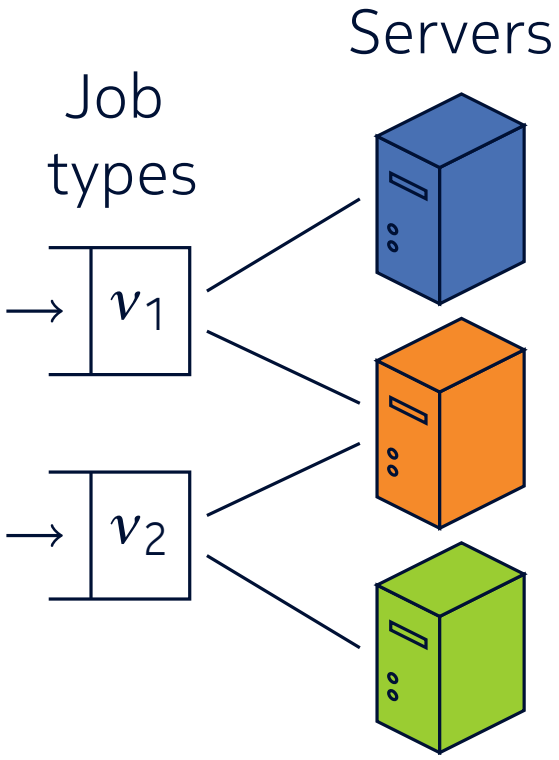
Numerical results

Extensions

Server pool model



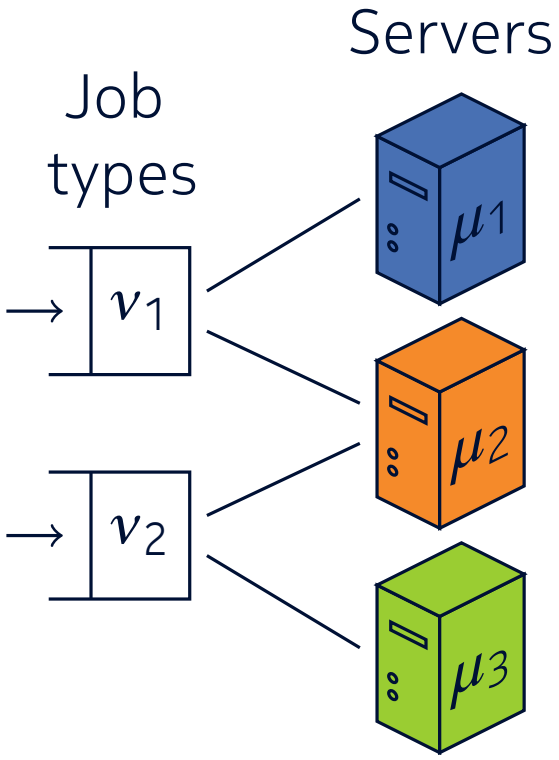
Server pool model



Arrivals:

- Type-k jobs arrive at rate ν_k
- Poisson arrival process

Server pool model



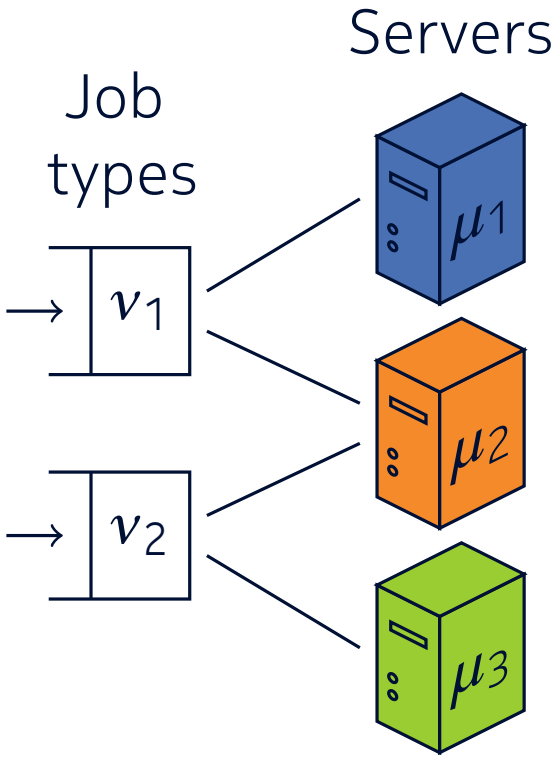
Arrivals:

- Type-k jobs arrive at rate ν_k
- Poisson arrival process

Service:

- Server s has capacity μ_s
- Exponentially distributed job sizes with unit mean

Server pool model



Arrivals:

- Type-k jobs arrive at rate ν_k
- Poisson arrival process

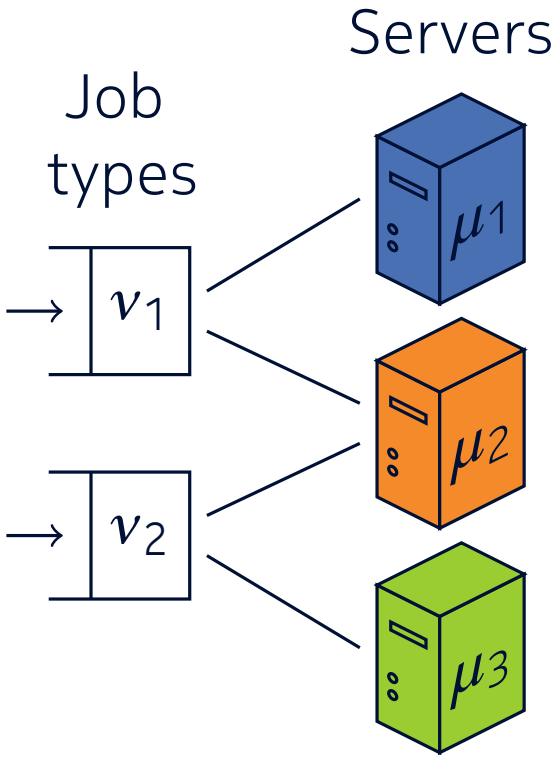
Service:

- Server s has capacity μ_s
- Exponentially distributed job sizes with unit mean

Admission limit:

- Each server has ℓ_s tokens

Server pool model



Arrivals:

- Type-k jobs arrive at rate ν_k
- Poisson arrival process

Service:

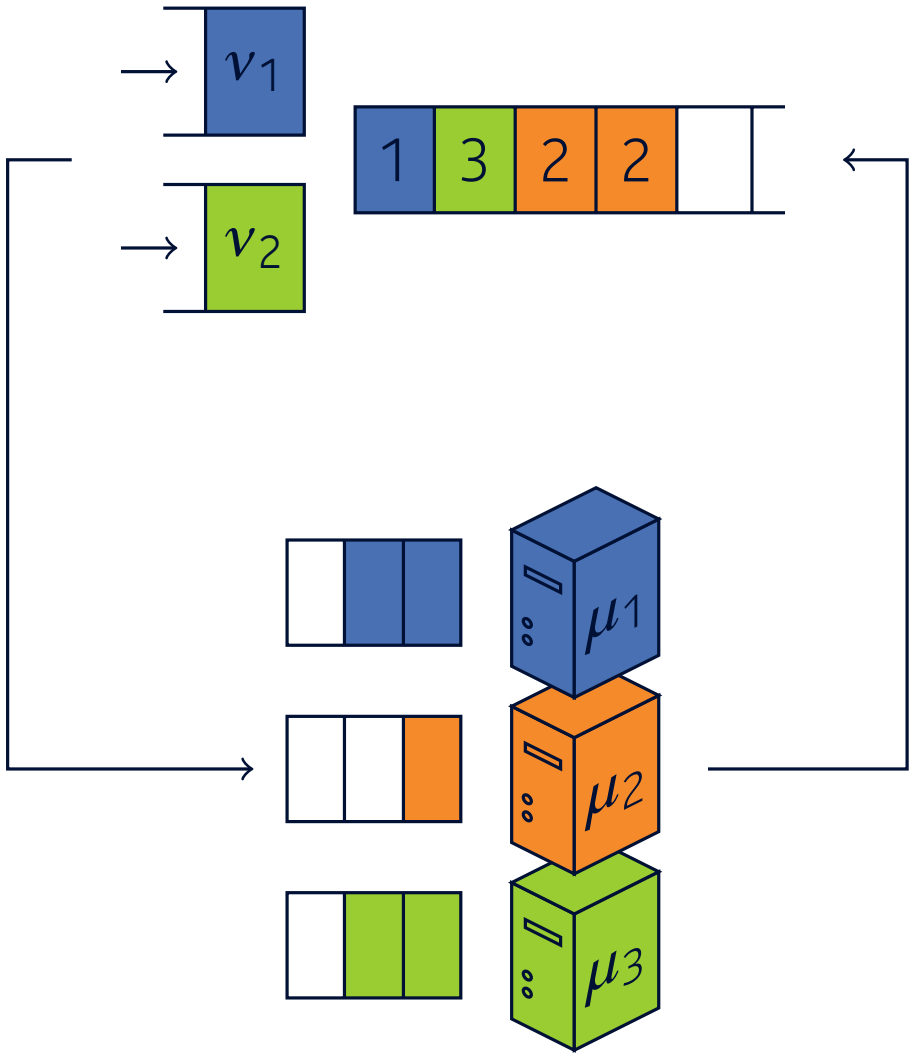
- Server s has capacity μ_s
- Exponentially distributed job sizes with unit mean

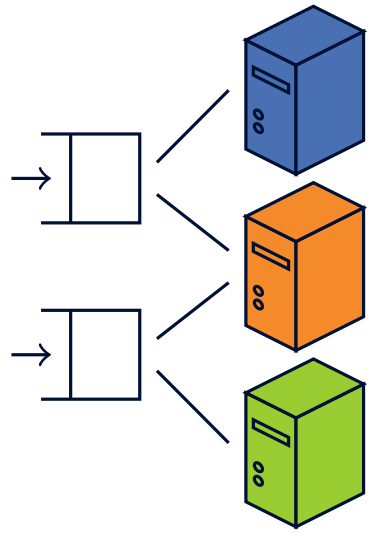
Admission limit:

- Each server has ℓ_s tokens

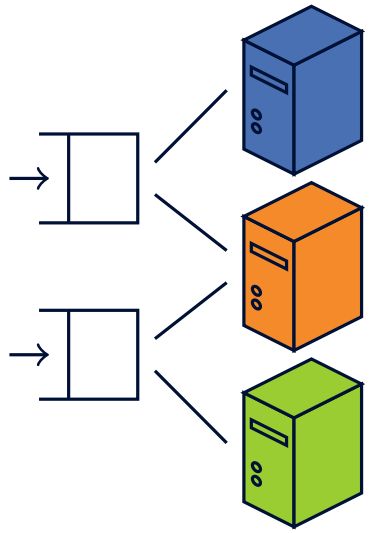
Remark: We only need to know the graph to analyze the algorithm

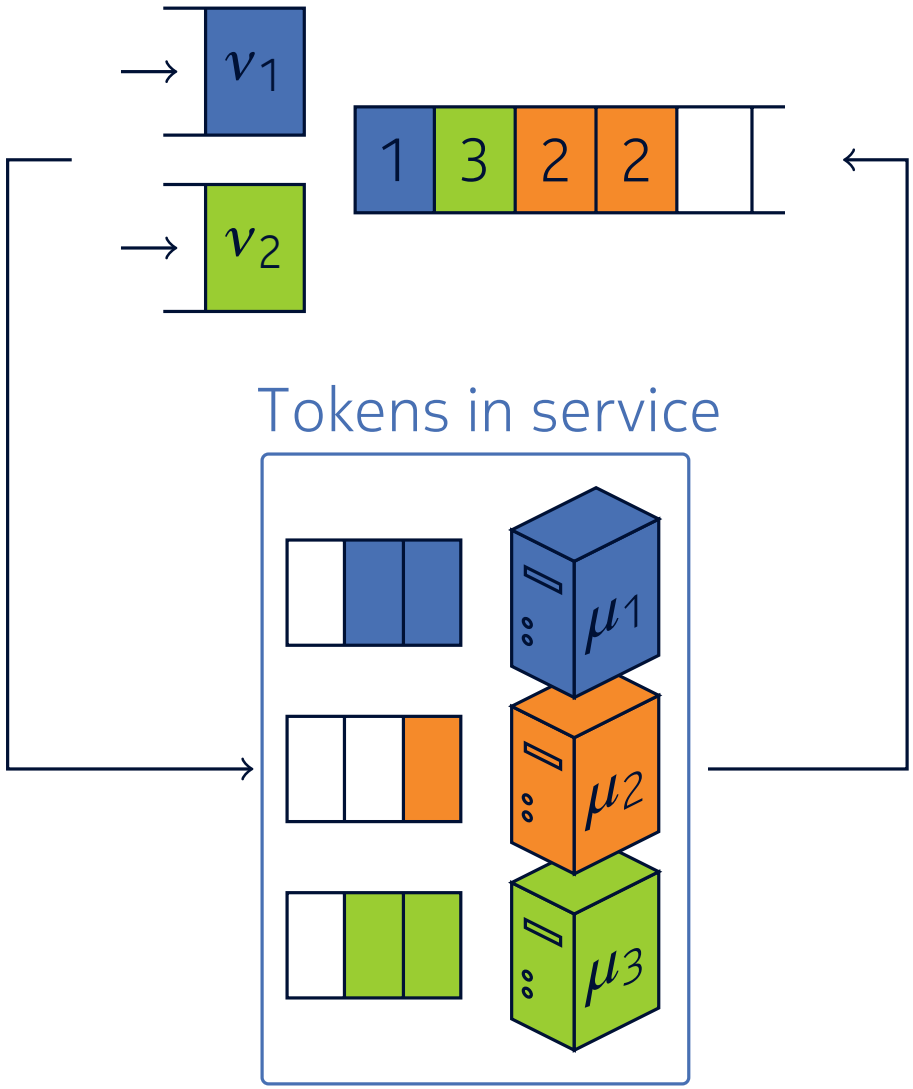
Queueing model
Describe the token state





Queueing model
Describe the token state

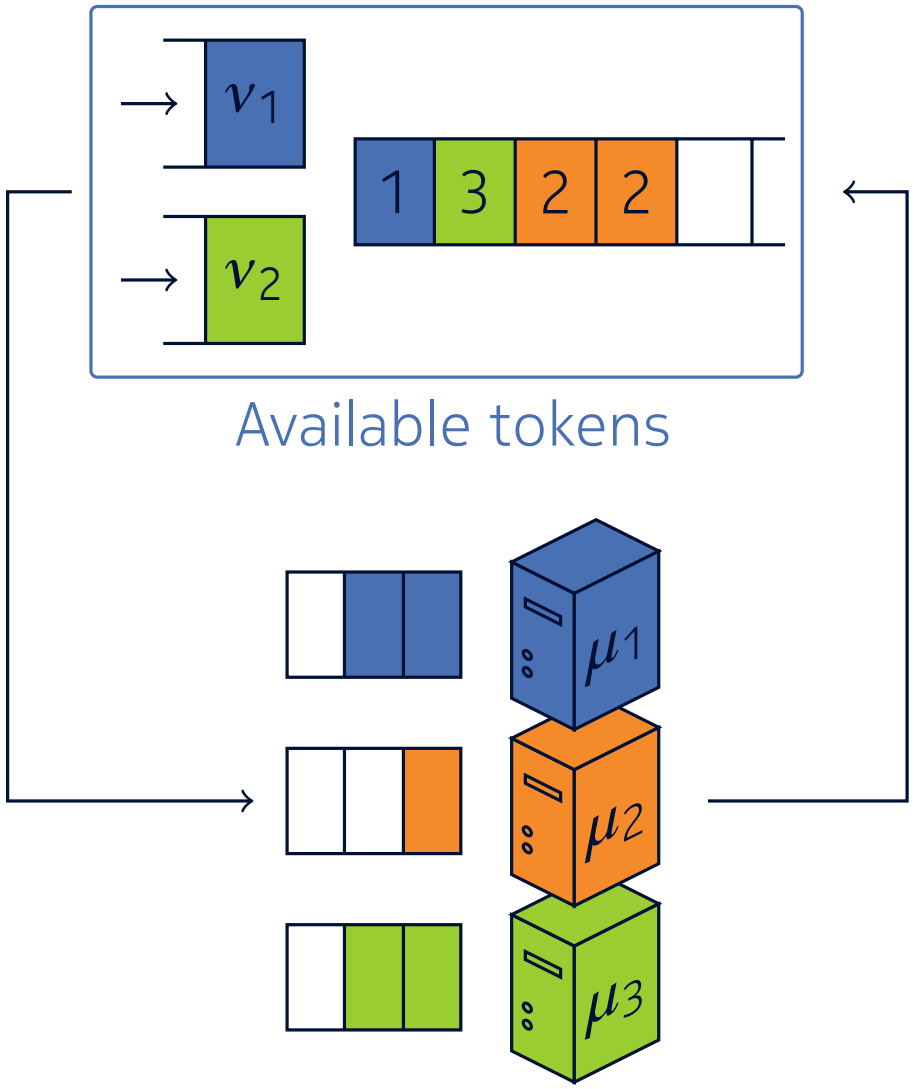


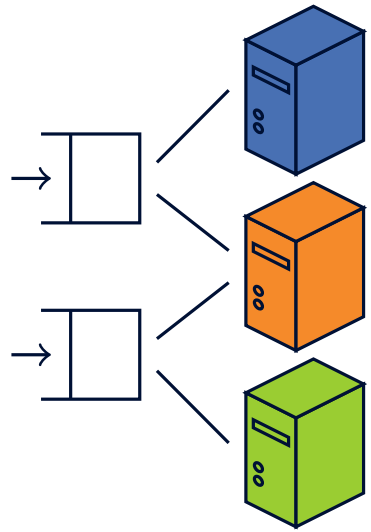


NOKIA Bell Labs

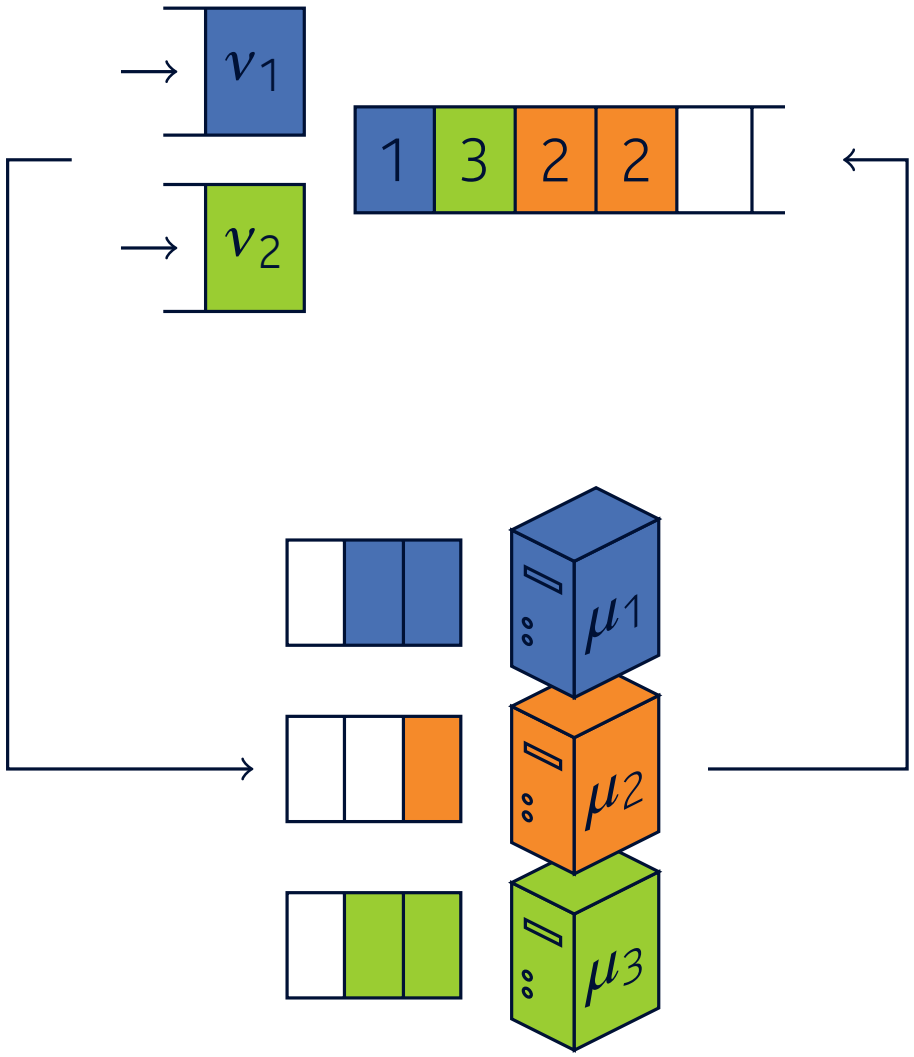
TELECOM ParisTech

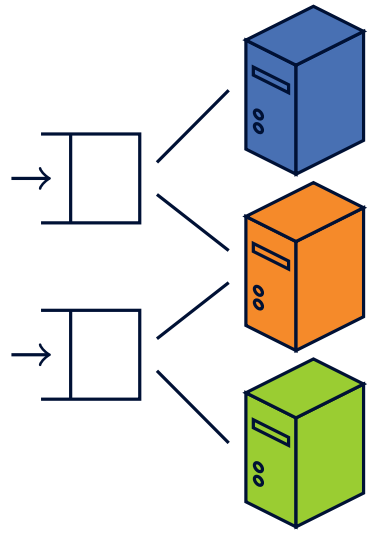
Queueing model
Describe the token state



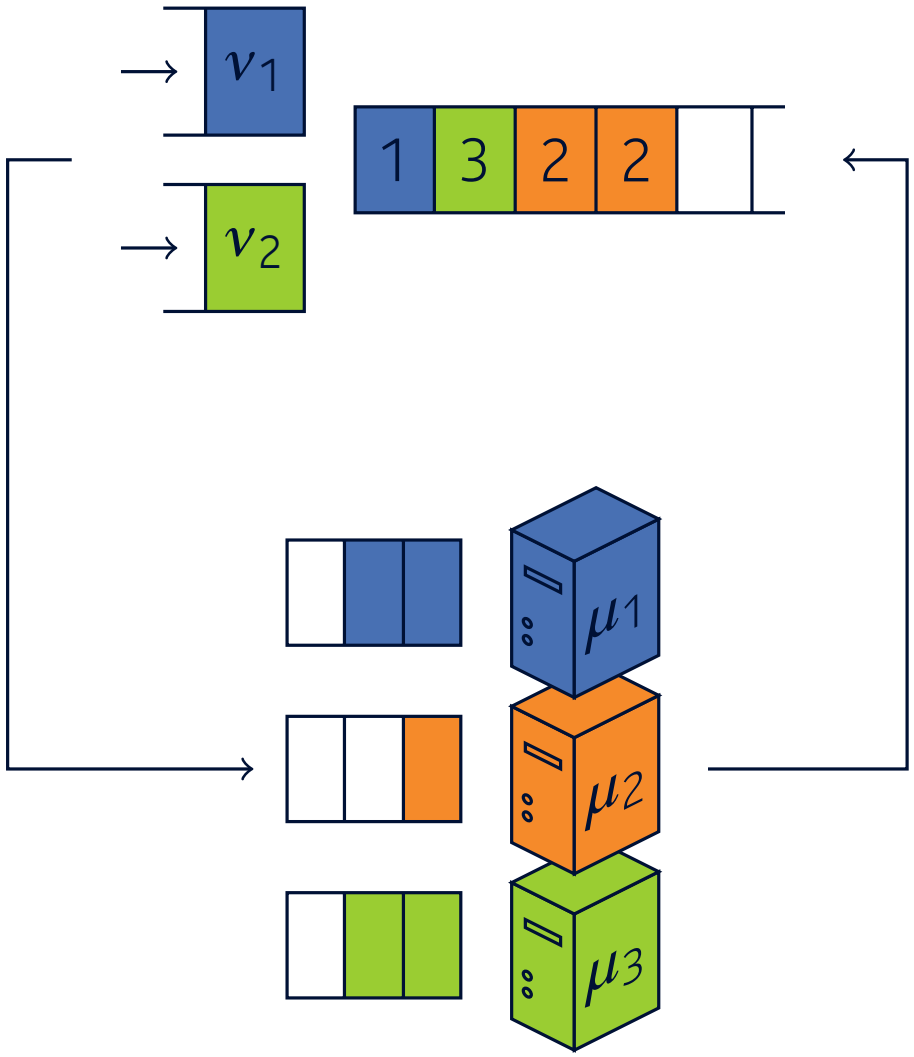


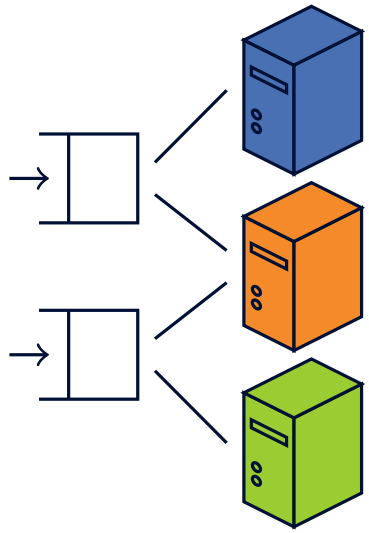
Queueing model
Describe the token state





Queueing model
Describe the token state

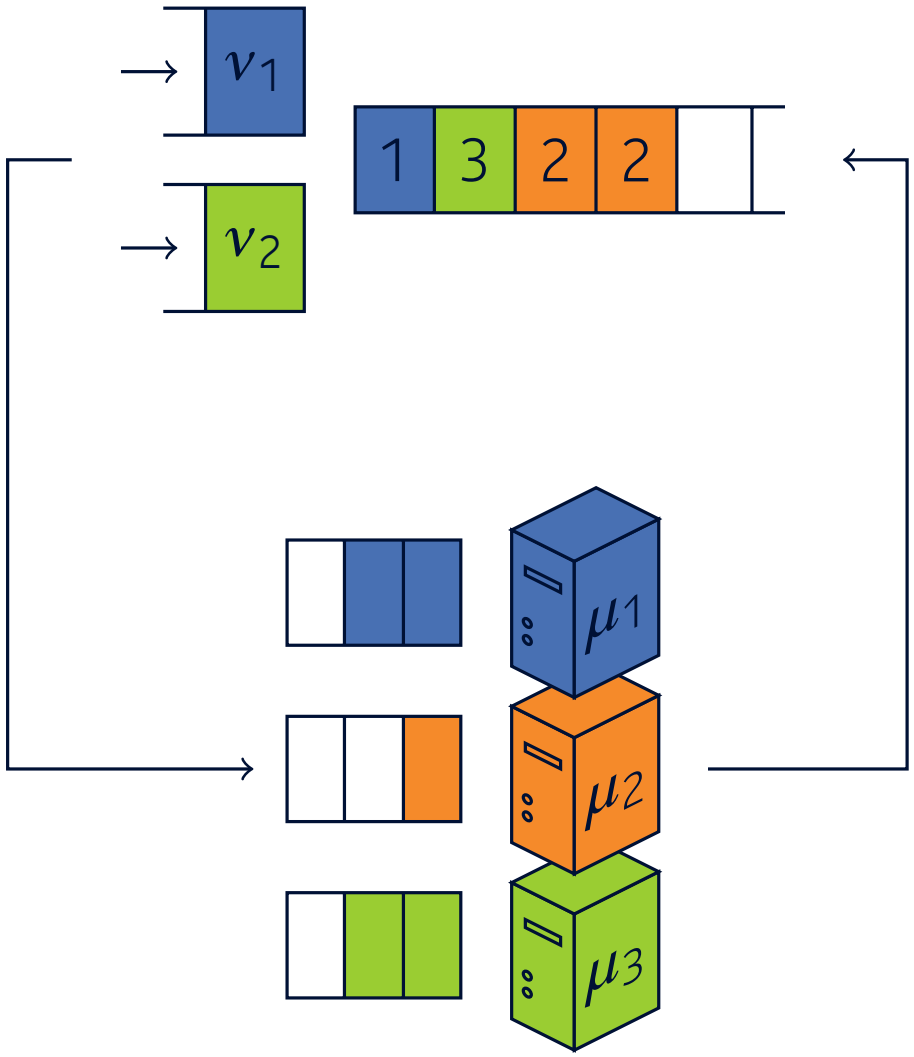


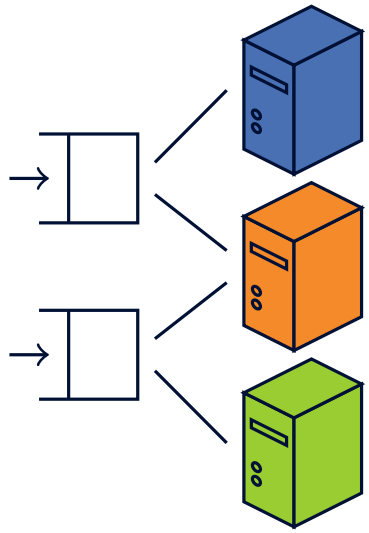


The job types play the part of the servers in the queue of available tokens

NOKIA Bell Labs TELECOM ParisTech

Queueing model
Describe the token state





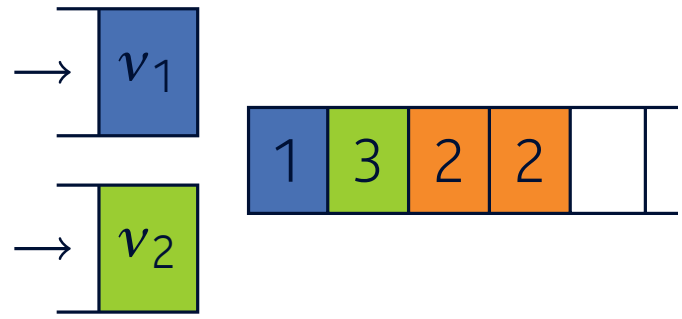
The job types play the part of the servers in the queue of available tokens

Network state:
The sequence of available tokens
t = (■, ■, ■, ■)

NOKIA Bell Labs TELECOM ParisTech

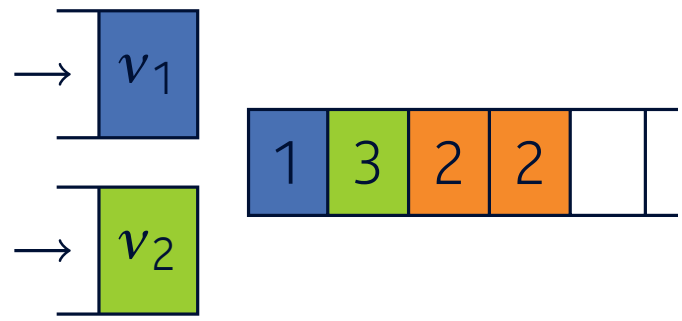
Order-independent (OI) queues

(Berezner, Kriel, and Krzesinski, 1995)



Order-independent (OI) queues

(Berezner, Kriel, and Krzesinski, 1995)

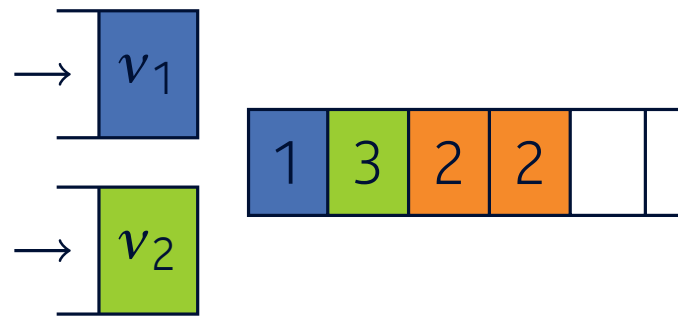


Definition:

- The total service rate only depends on the **number** of tokens of each server in the queue
- The service rate received by a token doesn't depend on the subsequent tokens

Order-independent (OI) queues

(Berezner, Kriel, and Krzesinski, 1995)



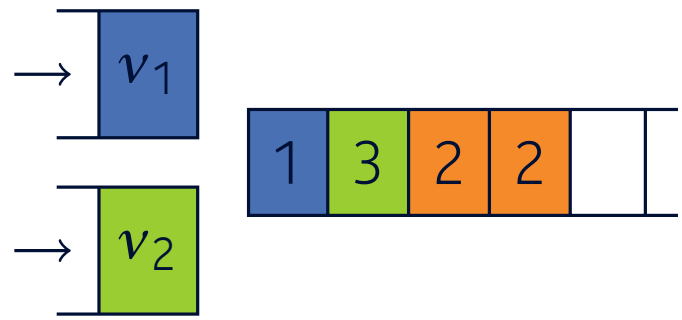
Definition:

- The total service rate only depends on the **number** of tokens of each server in the queue
- The service rate received by a token doesn't depend on the subsequent tokens

All the queues we consider are order-independent!

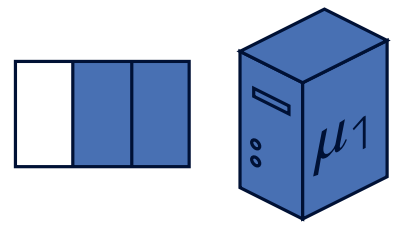
Order-independent (OI) queues

(Berezner, Kriel, and Krzesinski, 1995)



Definition:

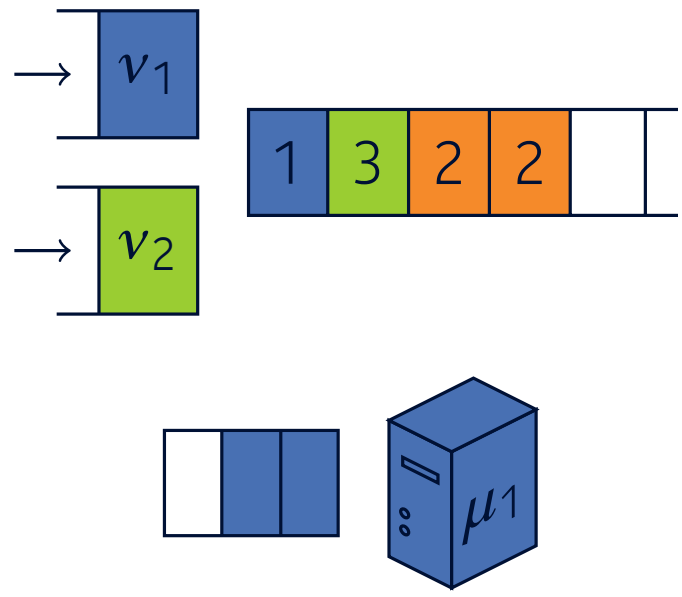
- The total service rate only depends on the **number** of tokens of each server in the queue
- The service rate received by a token doesn't depend on the subsequent tokens



All the queues we consider are order-independent!

Order-independent (OI) queues

(Berezner, Kriel, and Krzesinski, 1995)



Definition:

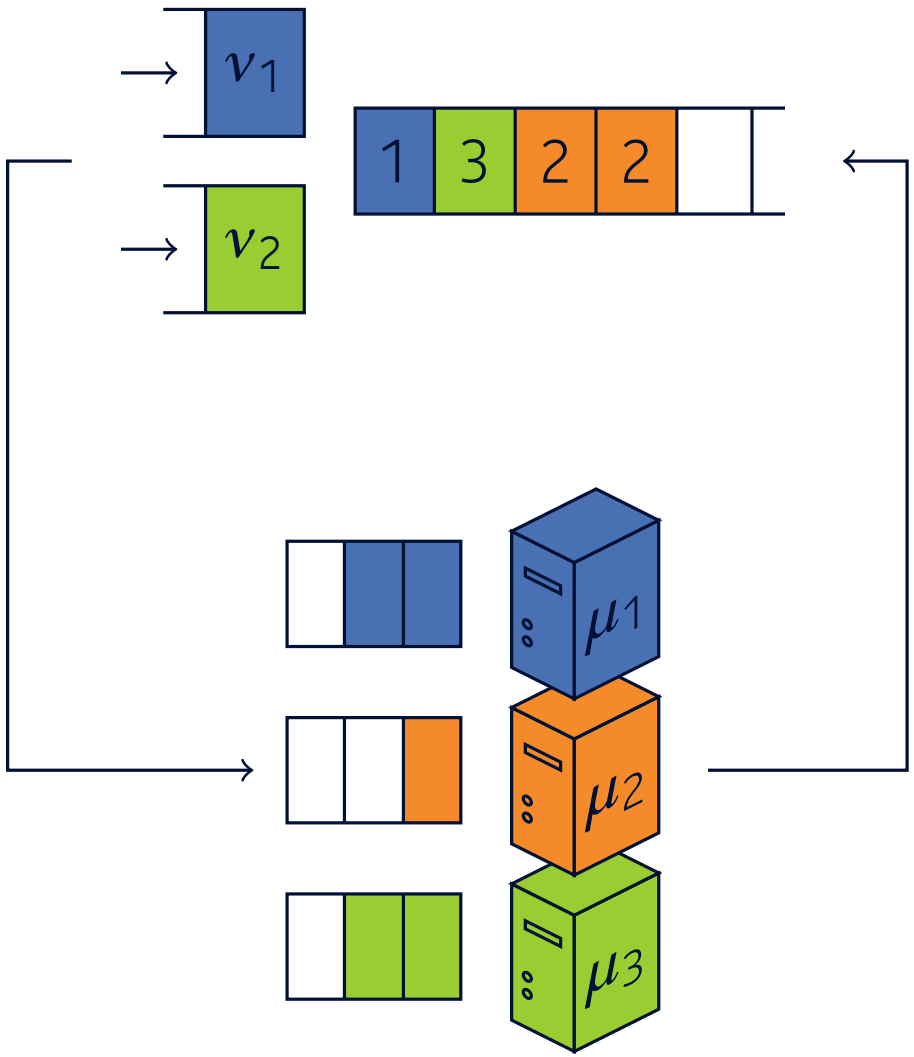
- The total service rate only depends on the **number** of tokens of each server in the queue
- The service rate received by a token doesn't depend on the subsequent tokens

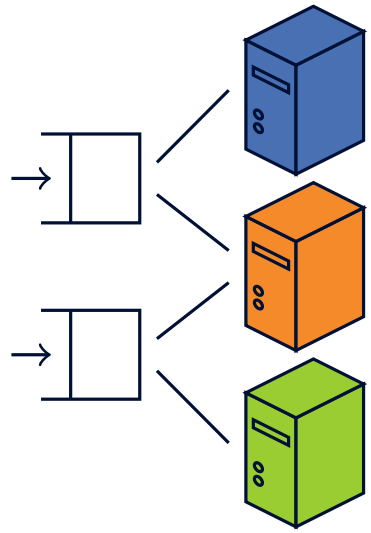
All the queues we consider are order-independent!

Quasi-reversibility:

- A network of OI queues has a product-form
- The stationary distribution of the network is unchanged by the addition of Markov routing

State aggregation

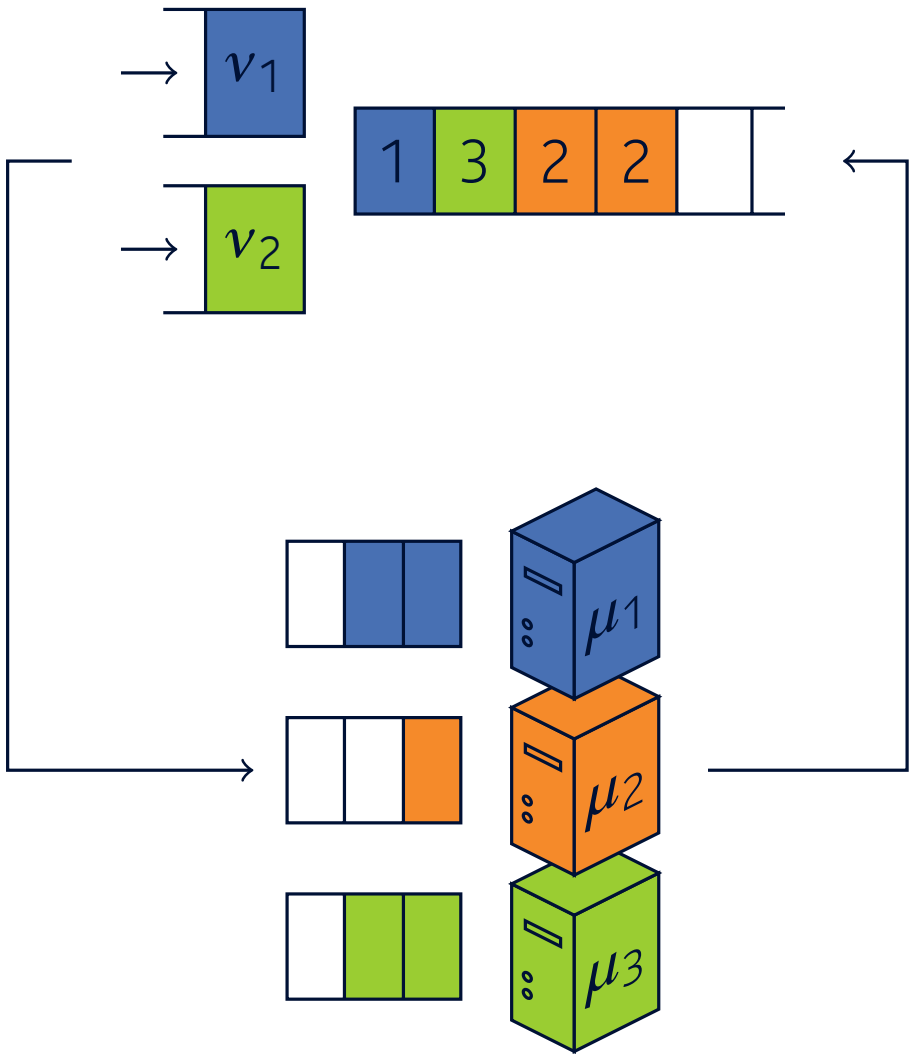


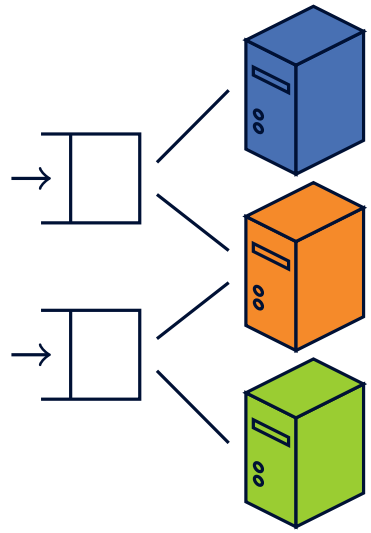


Detailed state:

t = (■, ■, ■, ■)

State aggregation





Detailed state:

t = (■, ■, ■, ■)

Aggregate state:

The number of available tokens

$$y = \begin{pmatrix} 1\blacksquare \\ 2\blacksquare \\ 1\blacksquare \end{pmatrix}$$

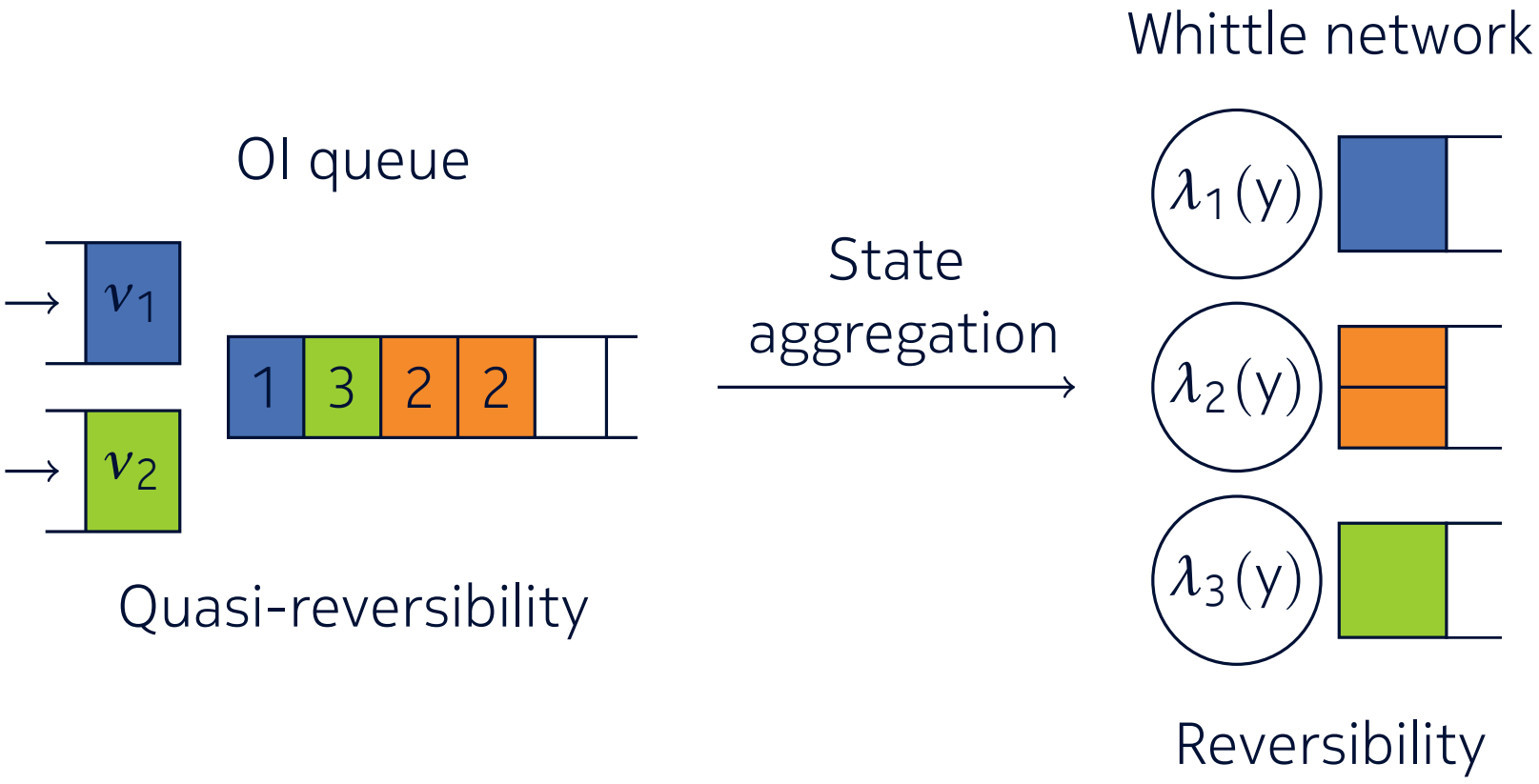
NOKIA Bell Labs TELECOM ParisTech

State aggregation

Equivalent Whittle network:
The stationary distribution of the aggregate state is that of a Whittle network (Whittle, 1985)

State aggregation

Equivalent Whittle network:
The stationary distribution of the aggregate state is that of a Whittle network (Whittle, 1985)



State aggregation

Equivalent Whittle network:
The stationary distribution of the aggregate state is that of a Whittle network (Whittle, 1985)

State aggregation

Equivalent Whittle network:
The stationary distribution of the aggregate state is that of a Whittle network (Whittle, 1985)

Average load balancing:
The average arrival rates to the servers, ignoring the order of tokens at the dispatcher, are as defined by the insensitive load balancing of (Bonald et al., 2004)

State aggregation

Equivalent Whittle network:
The stationary distribution of the aggregate state is that of a Whittle network (Whittle, 1985)

Average load balancing:
The average arrival rates to the servers, ignoring the order of tokens at the dispatcher, are as defined by the insensitive load balancing of (Bonald et al., 2004)

Performance metrics: Explicit formulas
$O(\ell_1 \times \ell_2 \times \ldots \times \ell_S)$ terms to compute

Outline

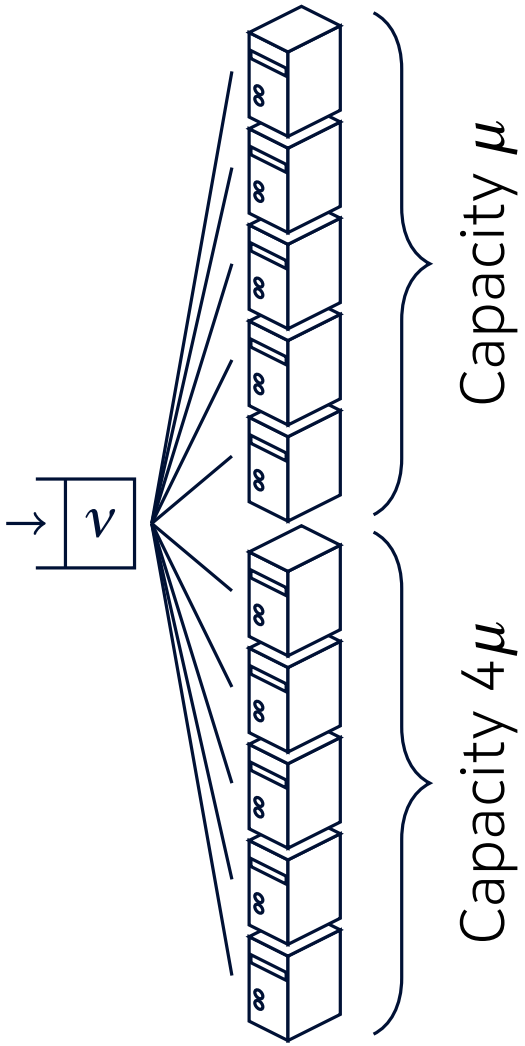
Algorithm

Queueing analysis

Numerical results

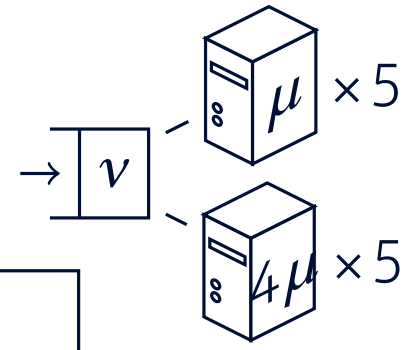
Extensions

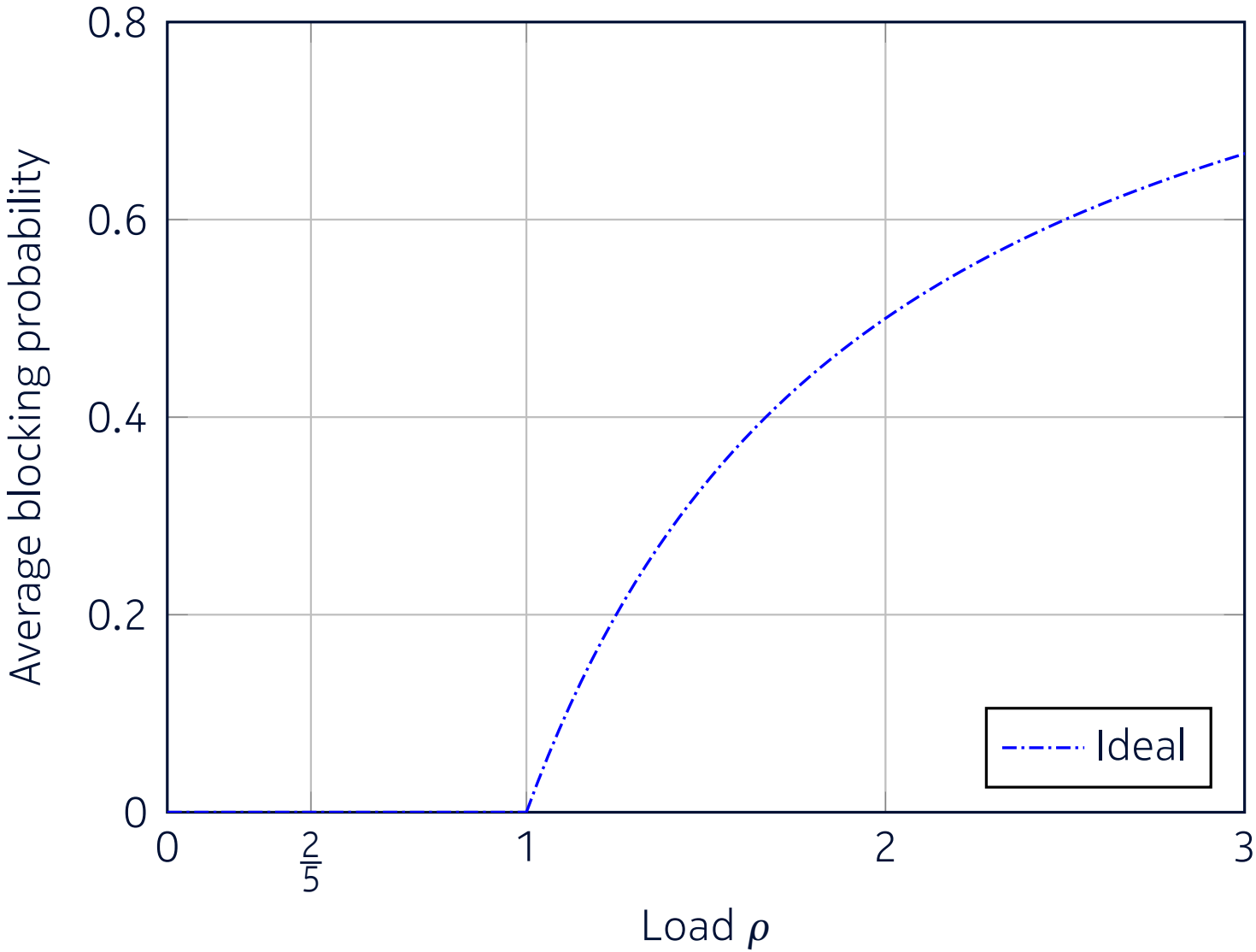
Servers with different service rates



- No compatibility constraints:
  All jobs can be assigned to all servers
- Poisson arrival process with rate ν
- Half of the servers have capacity μ,
  the other have capacity 4μ
- Each server has 6 tokens
- Load $\rho = \frac{\nu}{25\mu}$

Servers with different service rates

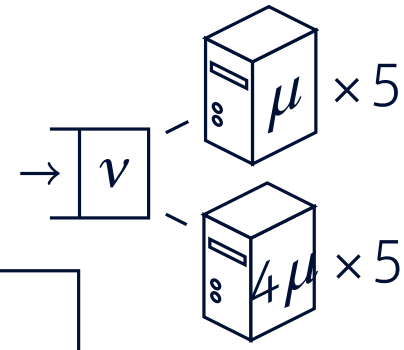


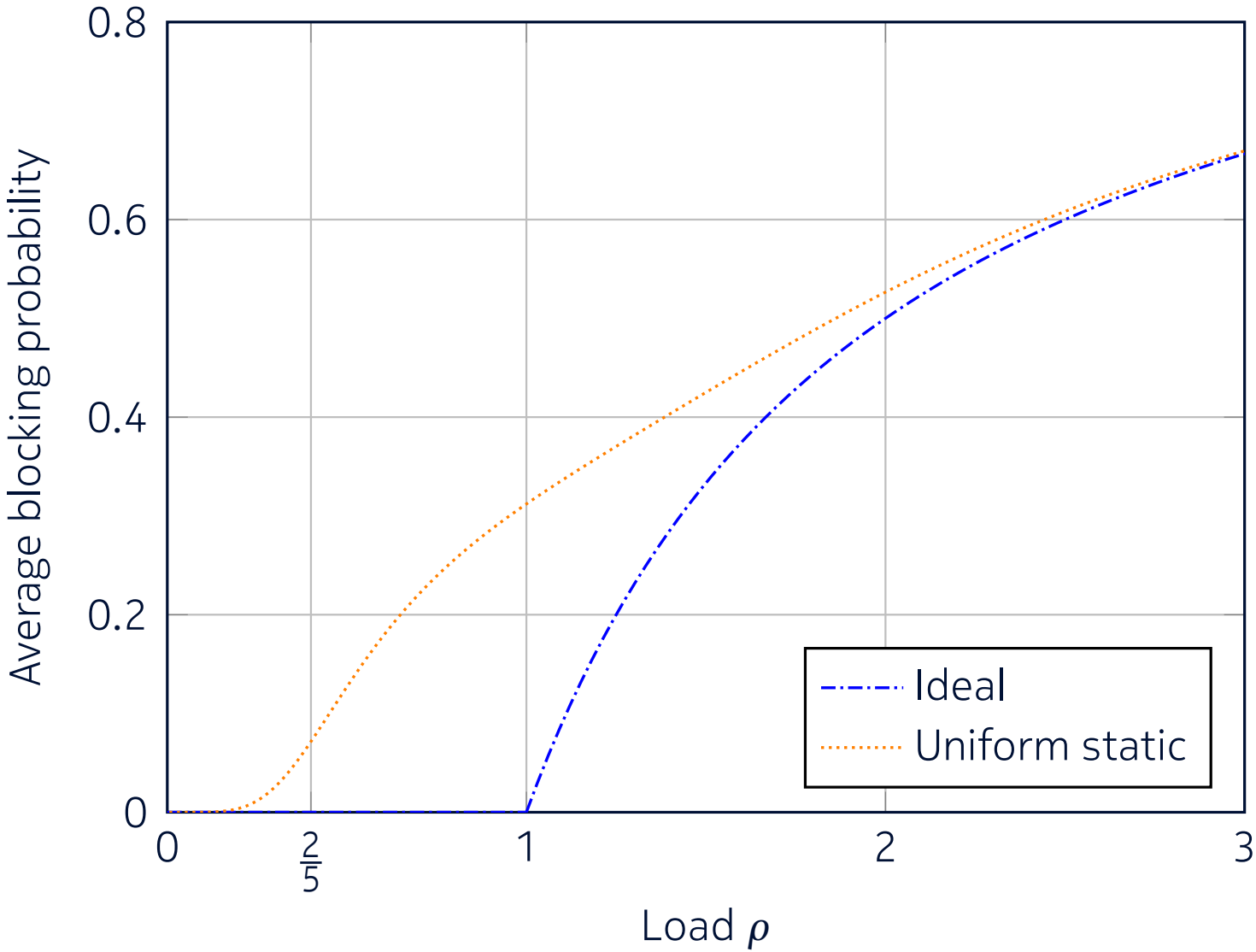


NOKIA Bell Labs

TELECOM ParisTech

Servers with different service rates

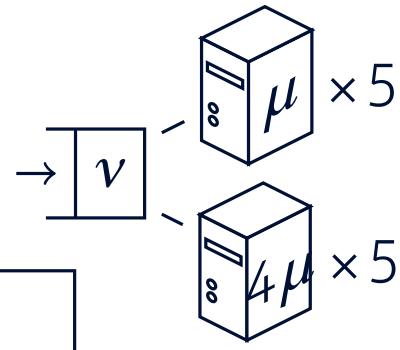


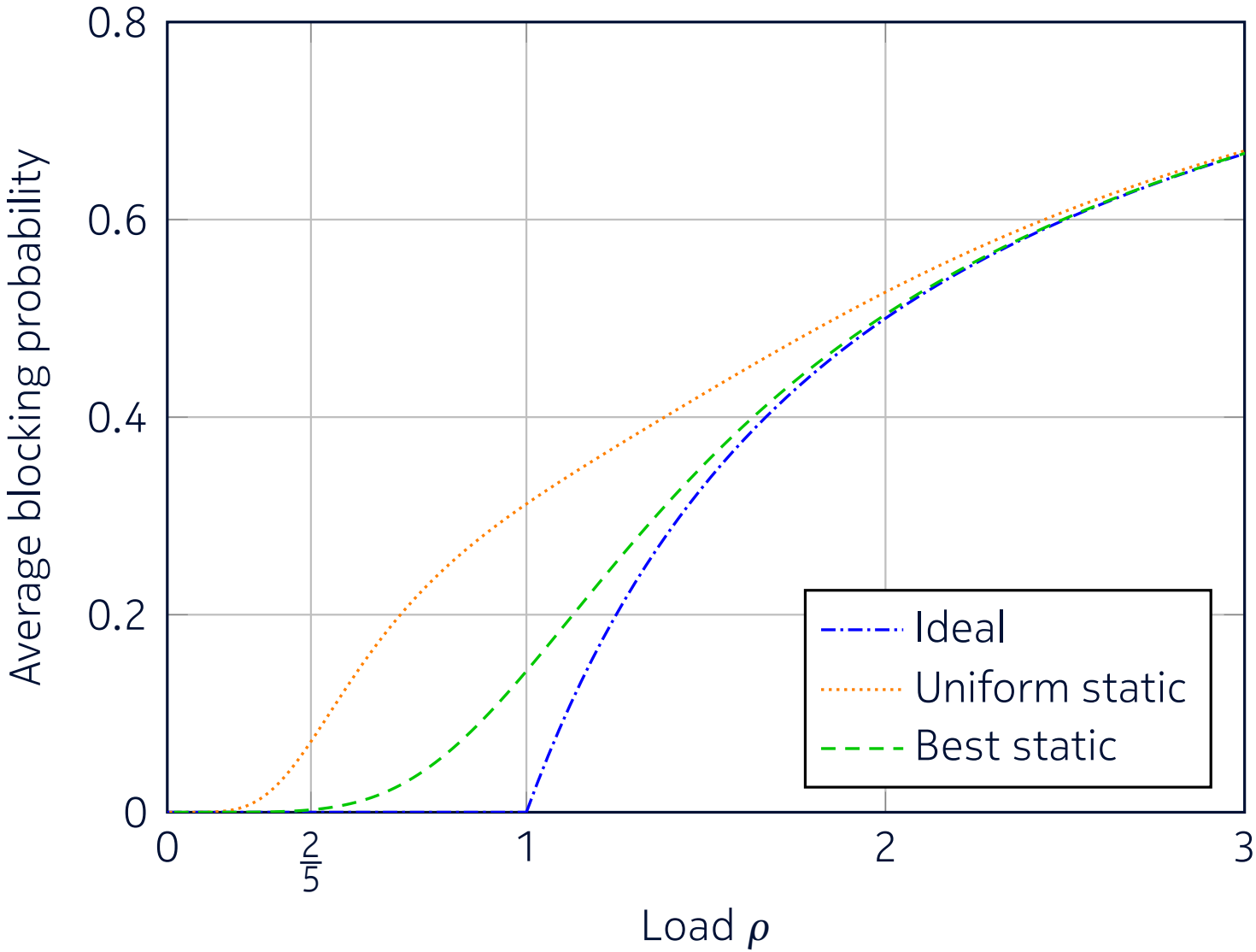


NOKIA Bell Labs

TELECOM ParisTech

Servers with different service rates

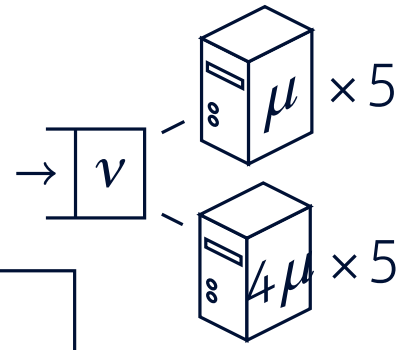


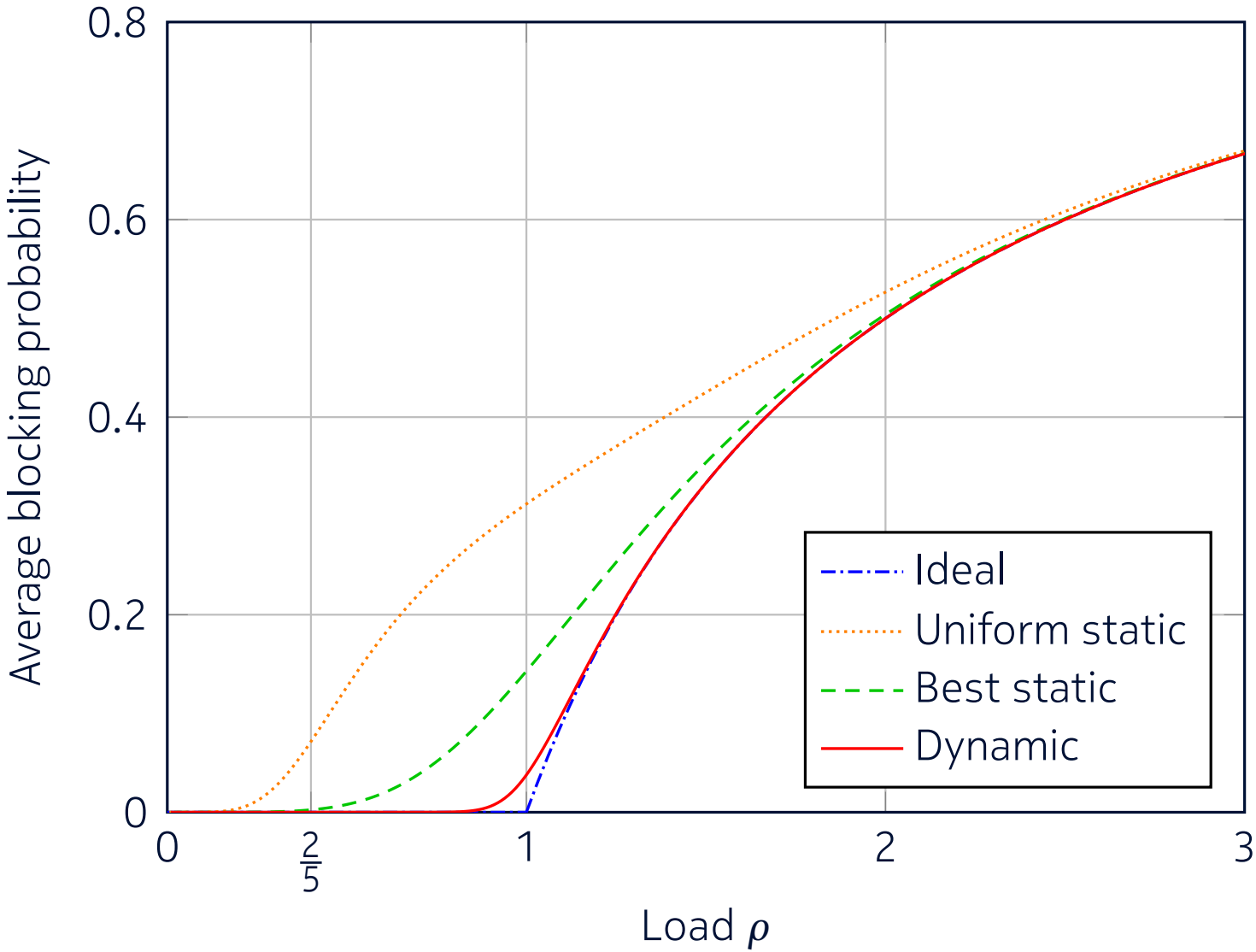


NOKIA Bell Labs

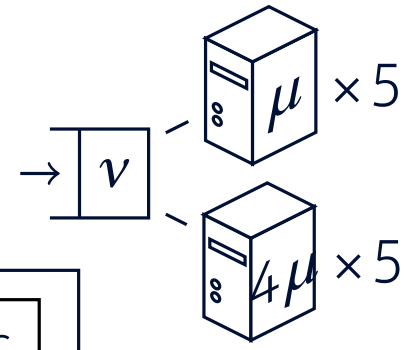
TELECOM ParisTech

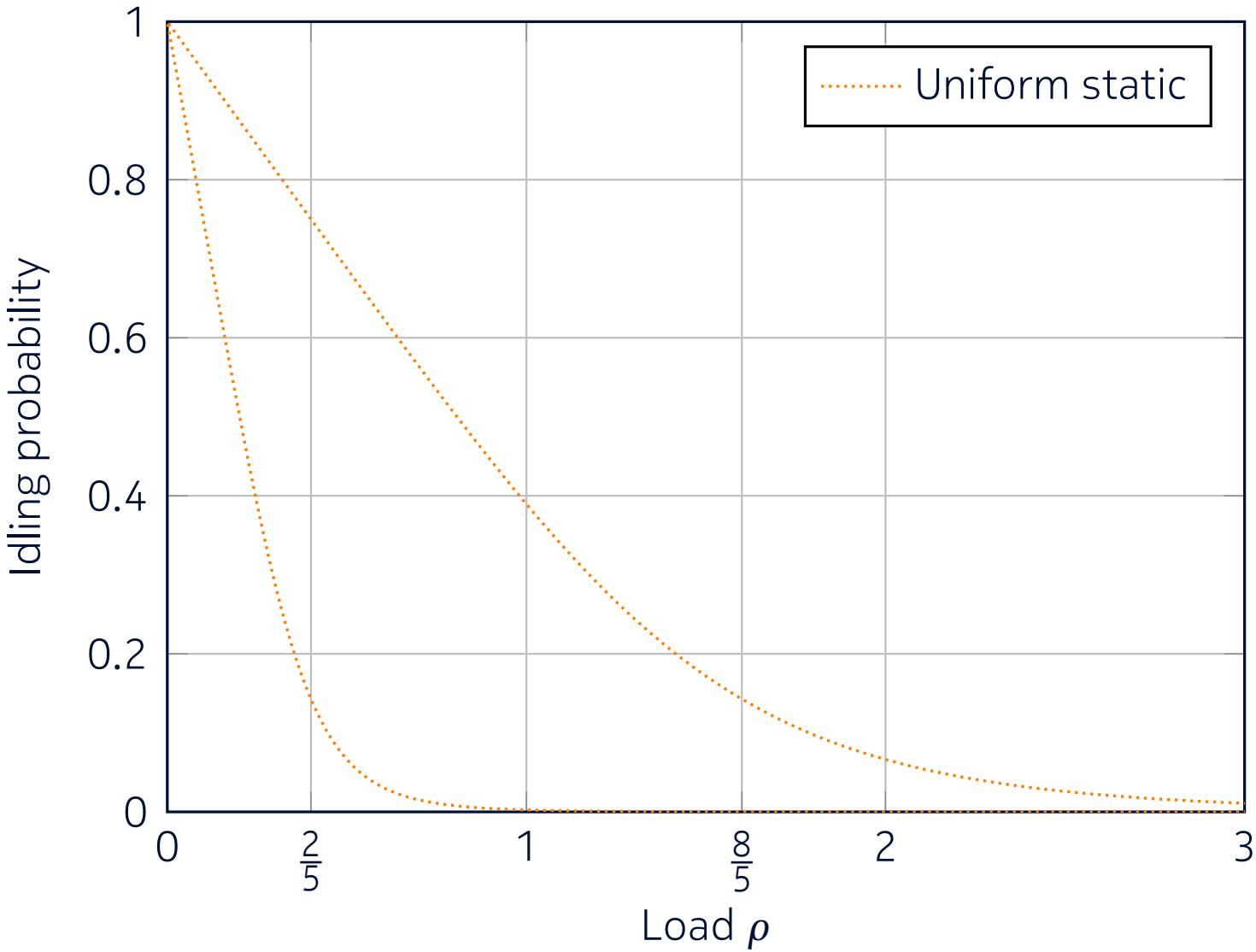
Servers with different service rates





Servers with different service rates

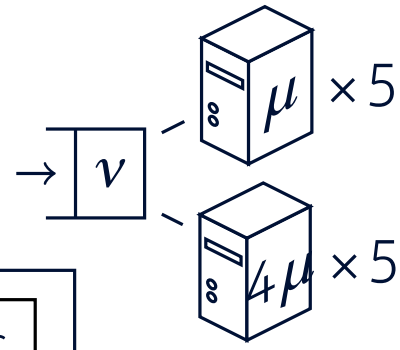


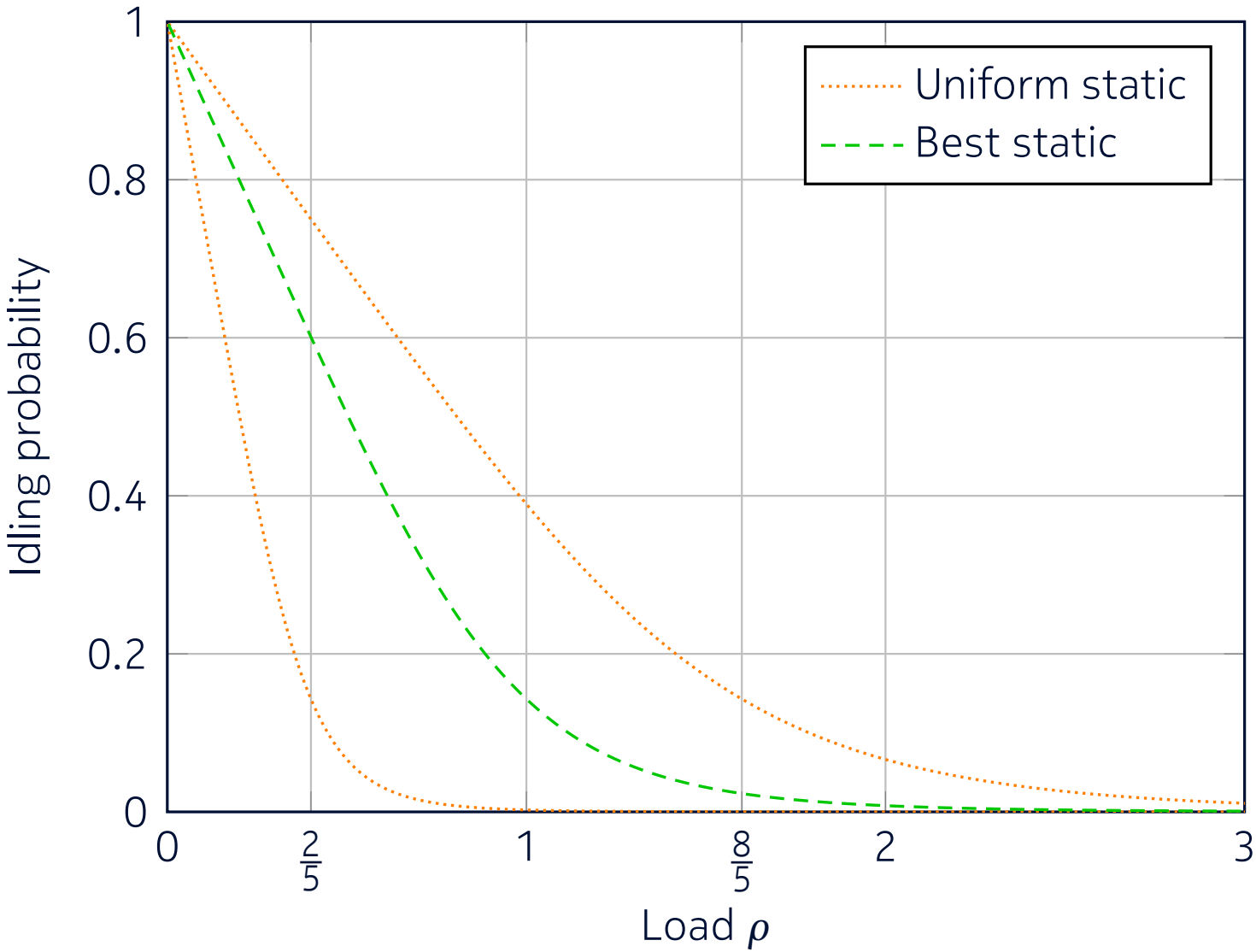


NOKIA Bell Labs

TELECOM ParisTech

Servers with different service rates

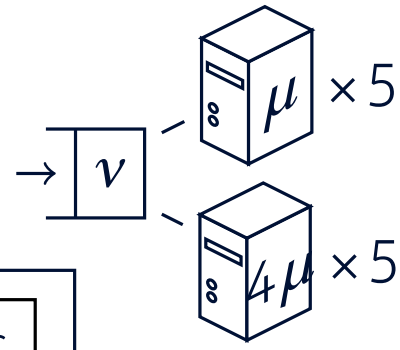


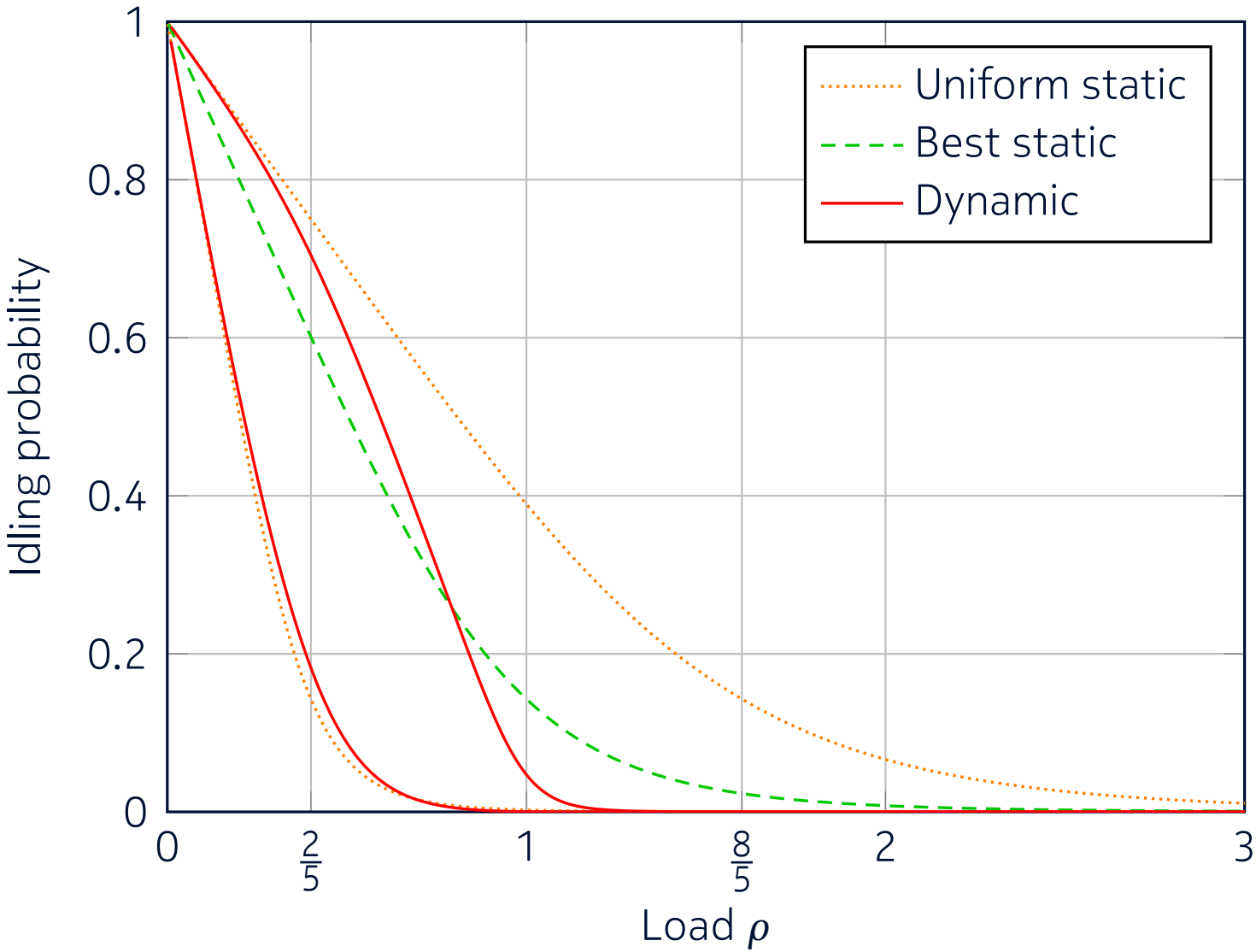


NOKIA Bell Labs

TELECOM ParisTech

Servers with different service rates

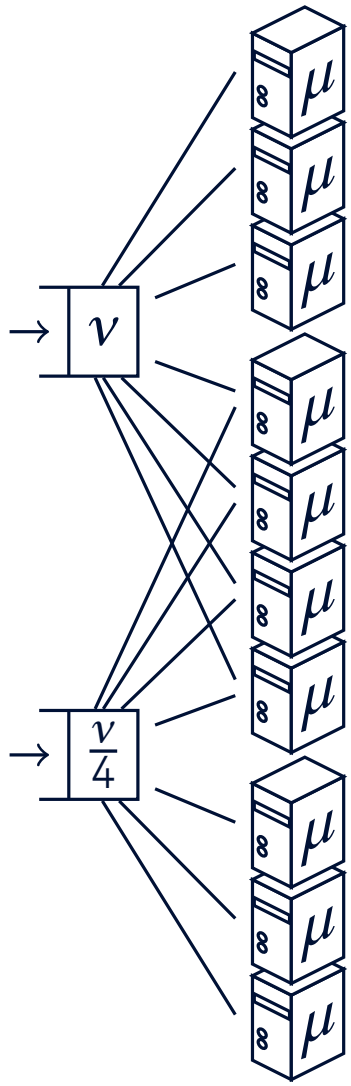




NOKIA Bell Labs

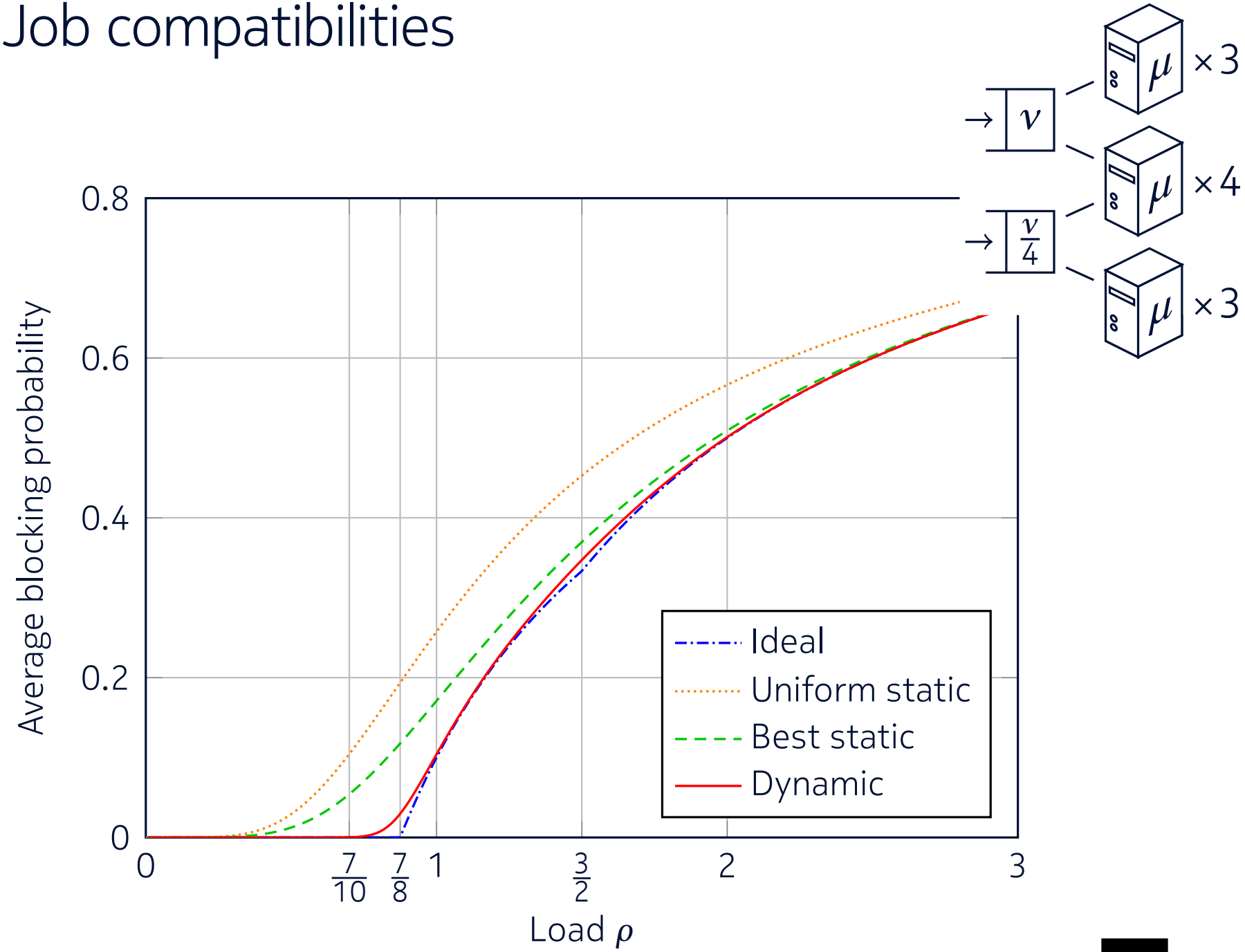
TELECOM ParisTech

Job compatibilities



- Two job types
- Poisson arrival processes with rates ν and $\frac{\nu}{4}$
- All servers have capacity μ
- Each server has 6 tokens
- Load $\rho = \frac{\nu}{8\mu}$

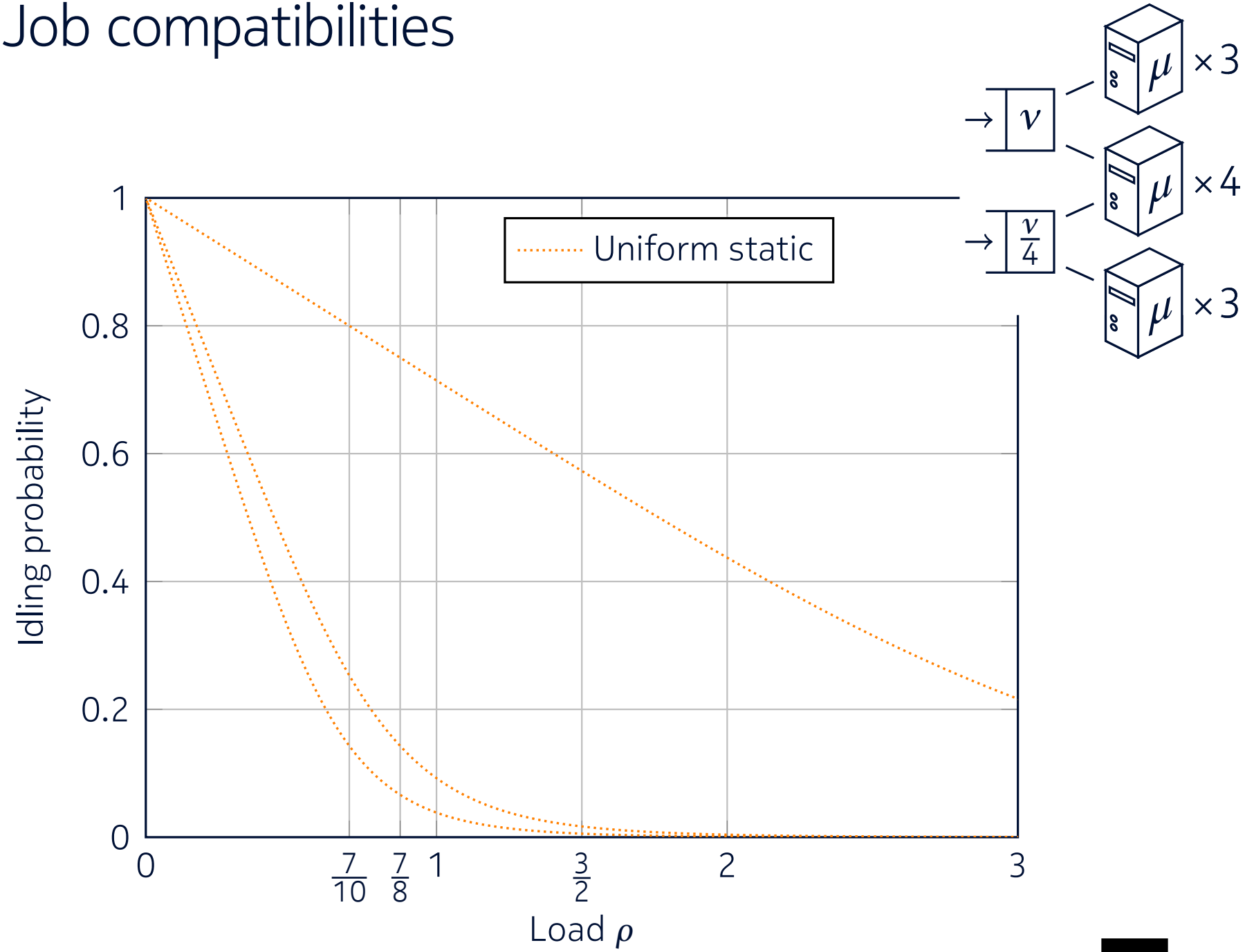
Job compatibilities



NOKIA Bell Labs

TELECOM ParisTech

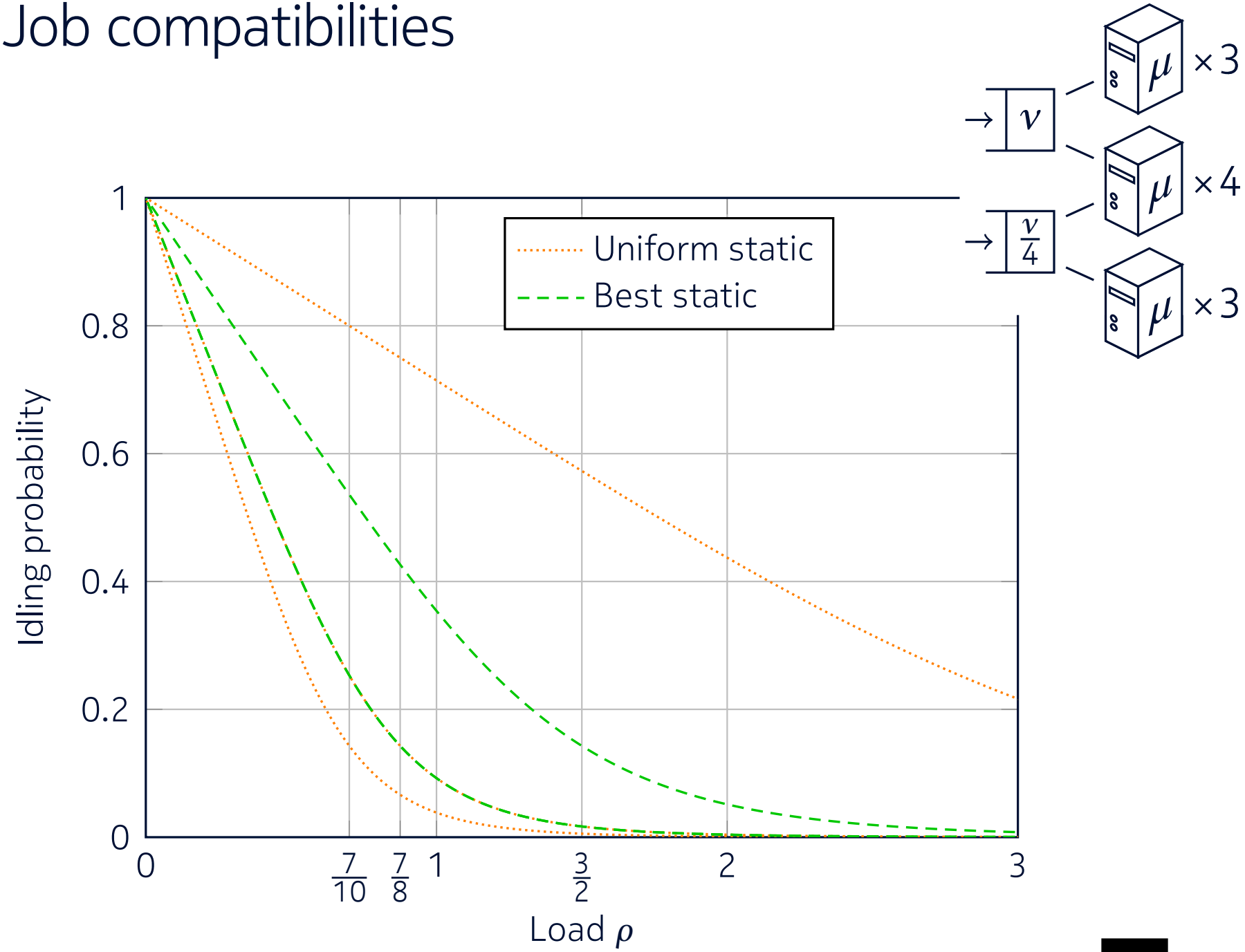
Job compatibilities



NOKIA Bell Labs

TELECOM ParisTech

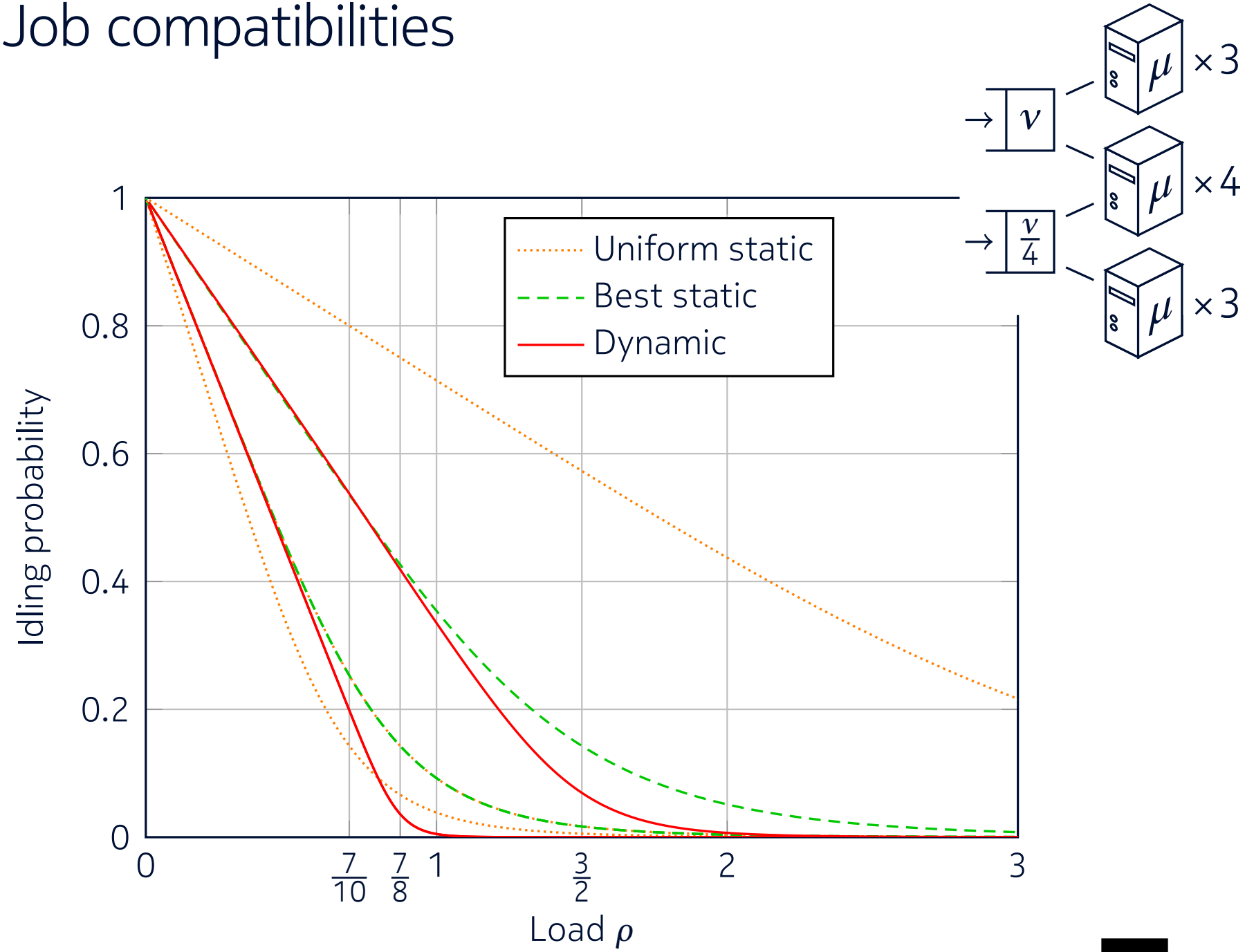
Job compatibilities



NOKIA Bell Labs

TELECOM ParisTech

Job compatibilities



NOKIA Bell Labs

TELECOM ParisTech

Outline

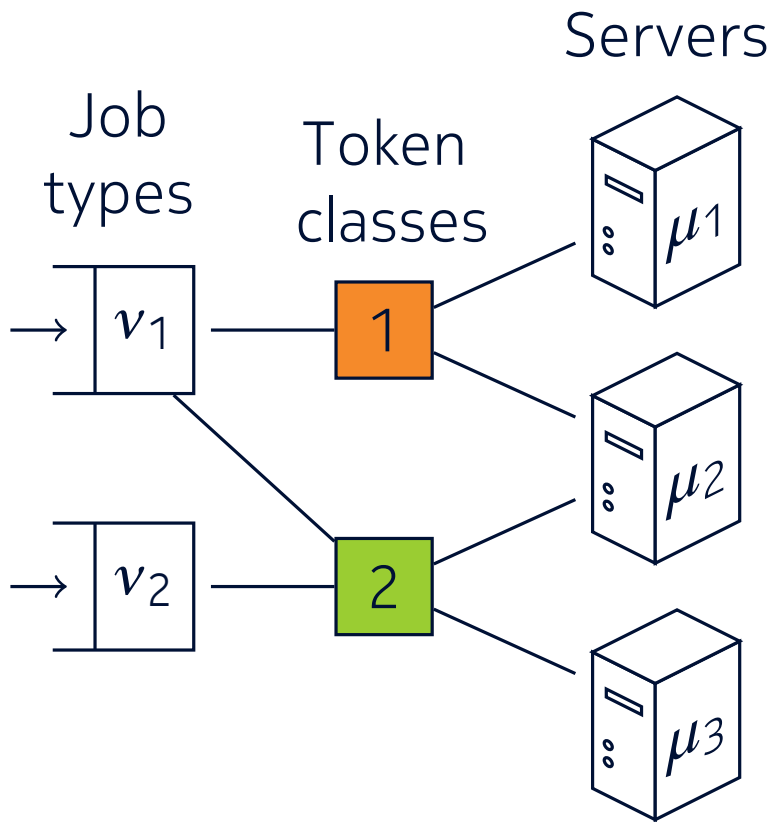
Algorithm

Queueing analysis

Numerical results

Extensions

Distributed processing



Each token identifies a **set of servers**

Non-blocking extensions
Future works

Instead of rejecting an incoming job when there is no available compatible token, we could …

Non-blocking extensions
Future works

Instead of rejecting an incoming job when there is no available compatible token, we could ...

Choose a compatible server uniformly at random

→ Join-Idle-Queue (Lu et al., 2011)

Non-blocking extensions
Future works

Instead of rejecting an incoming job when there is no available compatible token, we could ...

Choose a compatible server uniformly at random

→ Join-Idle-Queue (Lu et al., 2011)

Wait for the release of a compatible token

→ FCFS and assign to the longest idle server (ALIS) (Adan and Weiss, 2012)

Conclusion

Our contributions

- Insensitive and adaptative load balancing algorithm
- Queueing analysis based on order-independent queues under a Poisson arrival process
- Relate several existing works

Conclusion

Our contributions

- Insensitive and adaptative load balancing algorithm
- Queueing analysis based on order-independent queues under a Poisson arrival process
- Relate several existing works

Future works

- Derive simpler formulas for the performance prediction
- Evaluate the non-blocking versions of the algorithm

