

**The 25<sup>th</sup> International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems**

24<sup>th</sup> - 28<sup>th</sup> October 2022

Montreal, Canada (Virtual)

# **A Latency-Levelling Load Balancing Algorithm for Fog and Edge Computing**

Gabriele Proietti Mattia, Marco Magnani, Roberto Beraldi

Department of Computer, Control and Management Engineering “Antonio Ruberti”, Sapienza University of Rome, Italy

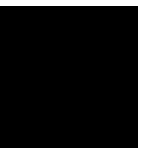
[proiettimattia@diag.uniroma1.it](mailto:proiettimattia@diag.uniroma1.it) · [gpm.name](http://gpm.name)



**SAPIENZA**  
UNIVERSITÀ DI ROMA

**DIAG**

Dipartimento di Ingegneria  
informatica, automatica e gestionale  
Antonio Ruberti



# Outline

- 1. Introduction**
- 2. Dynamic System Model**
- 3. Adaptive Heuristic**
- 4. Results**
- 5. Conclusions**

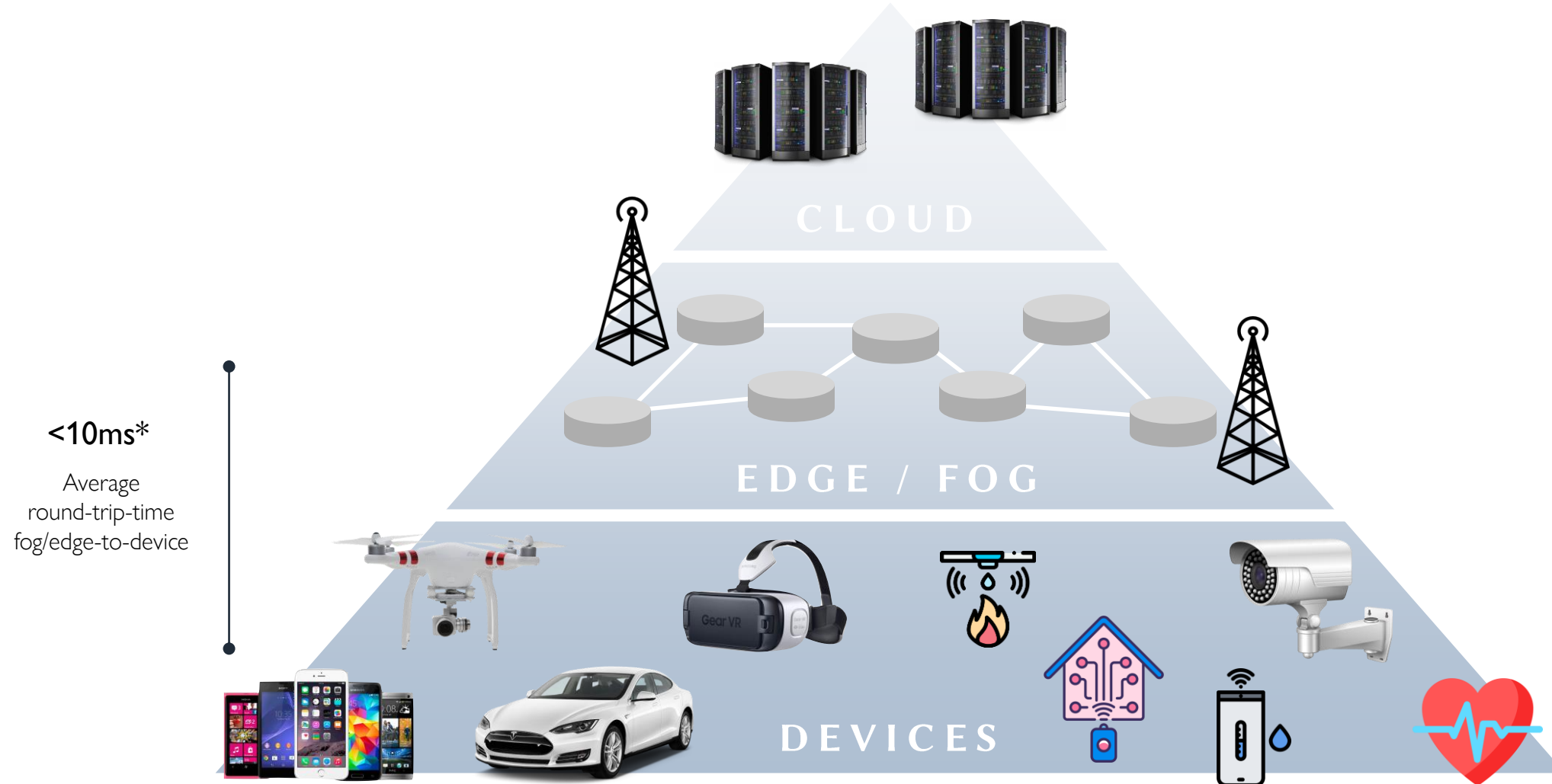
1

# Introduction

*The 25th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*

# Context

## Fog and Edge Computing



\*<https://geekflare.com/google-cloud-latency/>

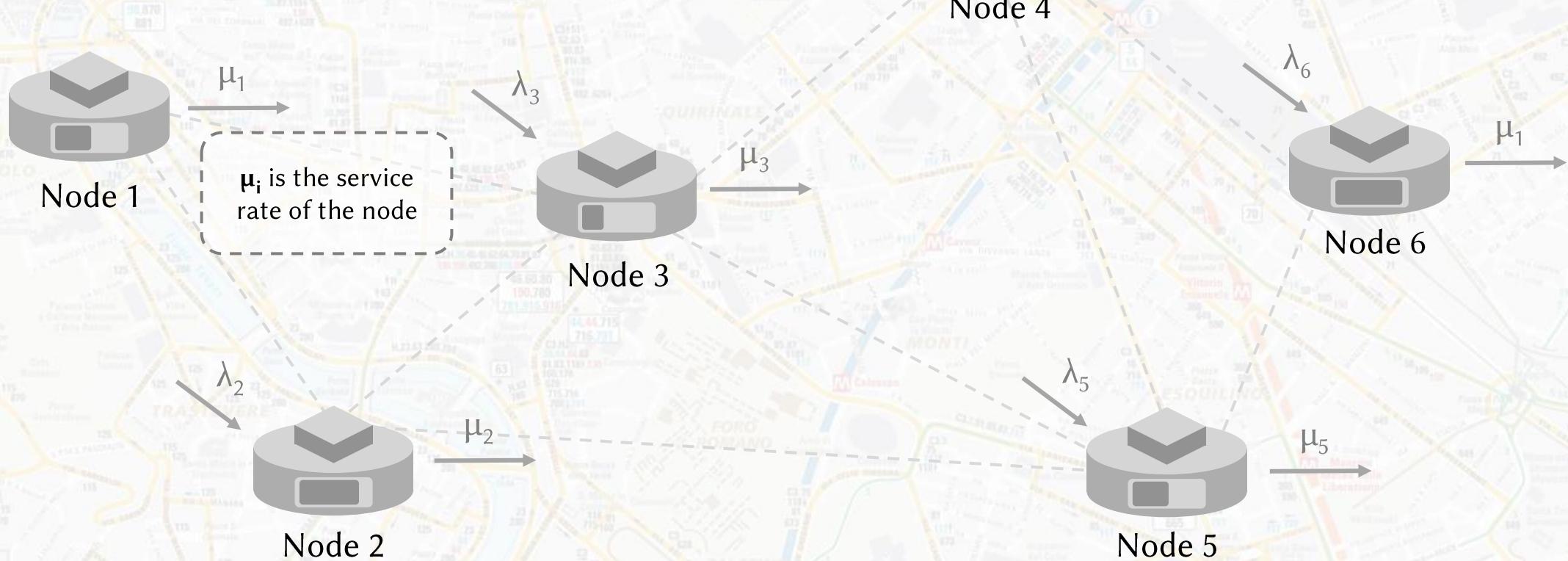
# Context

## The Fog/Edge Layer

We define  $l_i(t)$  as the latency that users experience at node  $i$  at time  $t$

$\lambda_i$  is the requests rate coming from the **clients**

$\mu_i$  is the service rate of the node

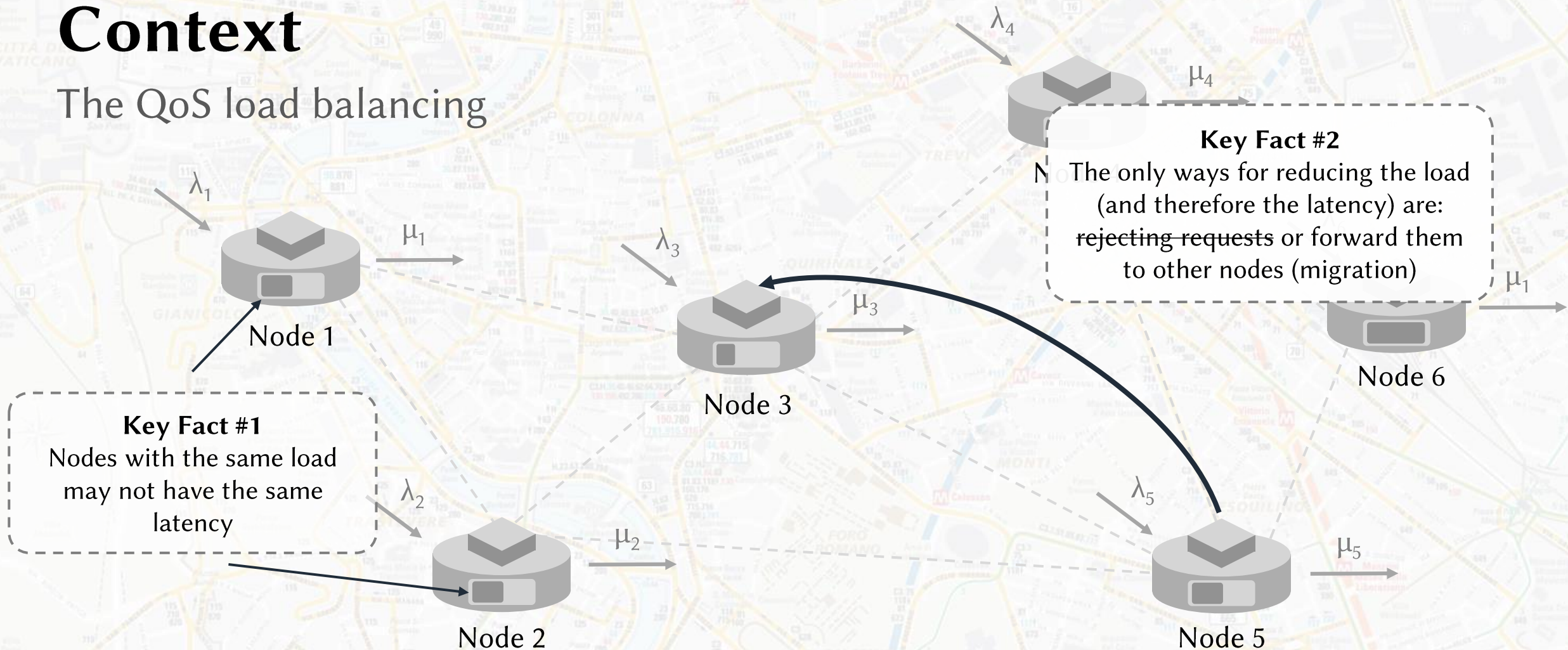


We suppose that each node can **cooperate** with each other, and they are **heterogeneous**. Nodes can act both as schedulers (◀) and workers (▶), we assume them as M/M/1/K queues



# Context

## The QoS load balancing



For guaranteeing a QoS-oriented load balancing, focused on latency, there are some key facts to keep in mind for reaching the goal: **each user must experience the same latency**

# Challenge

The main challenge of the work is finding a **load balancing algorithm** which is able to **level the latency across all the nodes**, considering that:

- the algorithm must be **fully distributed**, no central node or entity
- the algorithm should be **adaptive**
- we want to mathematically design a **system model** for understanding if a solution exists
- nodes are heterogeneous and they can be arranged in different **topologies**
- latency is proportional to the load ( $l_i(t) \propto x_i(t)$ )

In this work, we propose a **mathematical model** and an **adaptive heuristic** which allows to find a cooperation configuration that enable the latency-leveilling. Results has been shown in simulation and in an experimental setup.

# Related Work

- Harnal et al. in *Load Balancing in Fog Computing with QoS* (2022) propose the (OLBA) framework, which takes into account turn-around time and service delay and relies on Particle Swarm Optimization (PSO) for finding the best load balancing strategy, but the approach **is not fully decentralized**
- Tripathy et al. in *Secure-M2FBalancer: A Secure Mist to Fog Computing-Based Distributed Load Balancing Framework for Smart City Application* (2022) focuses on the QoS parameters but in a smart city setting and a smart allocation scheme is performed through a genetic algorithm. However, the **approach is not “online”**
- Nguyen et al in *Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy* (2022) propose a proxy-based approach that periodically monitors the pods' state, and according to the load, it forwards the requests to balance it; however, the approach **does not consider node heterogeneity**
- Singh et al. in *Container-based load balancing for energy efficiency in software-defined edge computing environment propose a container-as-a-service (CaaS)* (2021) load balancing strategy that is focused on energy efficiency, however, the approach is based on **two steps service level agreement** while our tries to use only one, moreover the results are only provided in **simulations**.

Our work instead provide a **fully decentralized algorithm**, that performs an **online scheduling** also addressing nodes **heterogeneity** and we provide results from a mathematical **model**, **simulations** and from an **experimental setting**.



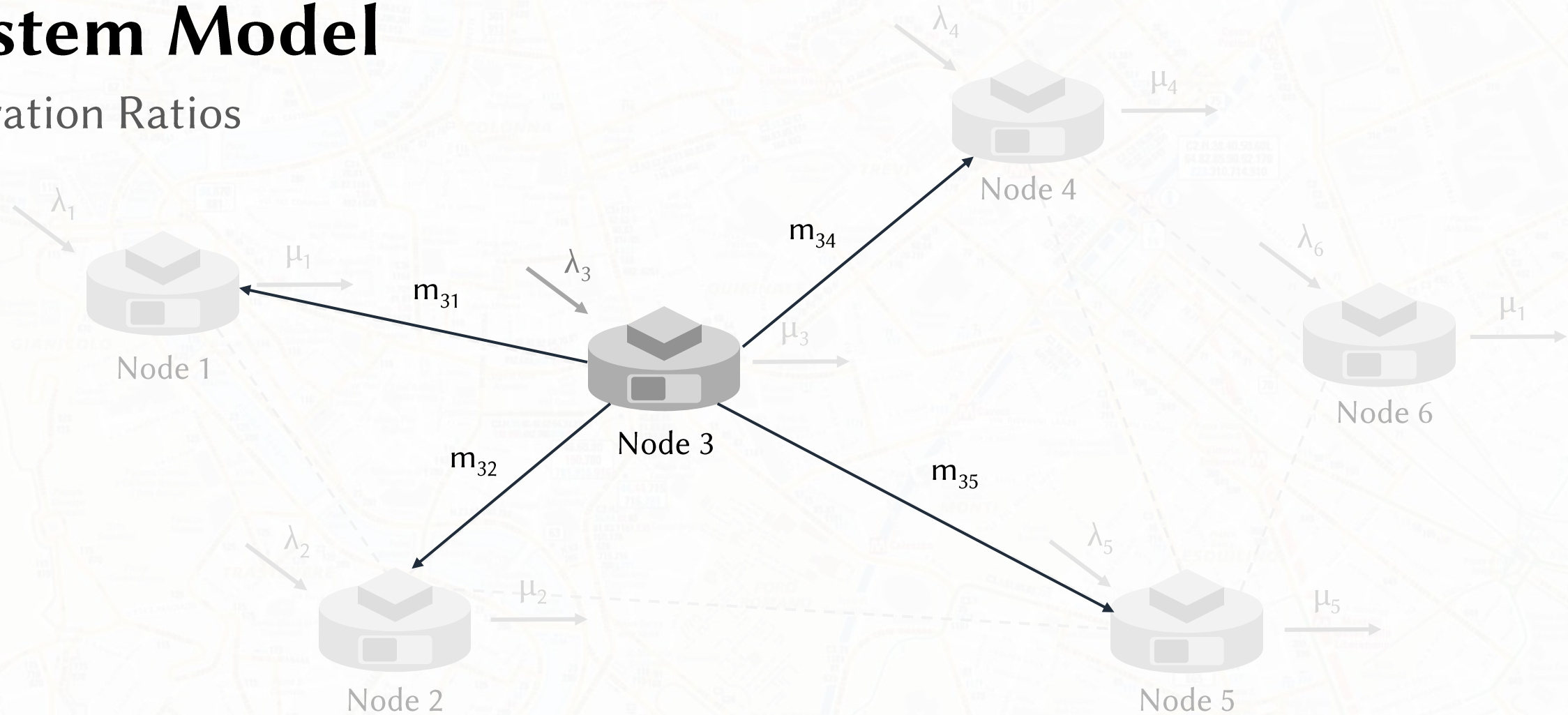
2

# Dynamic System Model

*The 25th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*

# System Model

## Migration Ratios



Each node has to decide a set of **migration ratios** ( $m_{ij}$ ) which describe the portion of the  $\lambda_i$  requests to forward to each node in order to balance the load

# System Model

## System Description

We now model the state of each node over time, called  $x_i(t)$ , considering the migrations capabilities. We wonder which is the **total load** (in term of requests per second) that a node sees over time.

$$x_i(t) = \lambda_i - \underbrace{\sum_{j \in V} a_{ij} \lambda_i m_{ij}(t)}_{\text{Total traffic **given** to all neighbours}} + \underbrace{\sum_{j \in V} a_{ji} \lambda_j m_{ji}(t)}_{\text{Total traffic **received** from all neighbours}}$$

Net Load at time  $t$  →  $x_i(t)$

Traffic from underlying clients of node  $i$  (fixed over time) →  $\lambda_i$

Adjacency, 0 or 1 →  $a_{ij}$

Adjacency, 0 or 1 →  $a_{ji}$

**Migration ratio** from  $j$  to  $i$  at time  $t$ , they are our unknown →  $m_{ij}(t)$  and  $m_{ji}(t)$

# System Model

## Dynamic of the load

If we would find the  $\mathbf{m}_{ij}(t)$  ( $\forall i,j$ ) the problem is solved. Since it is difficult to define them directly, we try to define their **dynamics**, then we will go back to their definition, at least numerically.

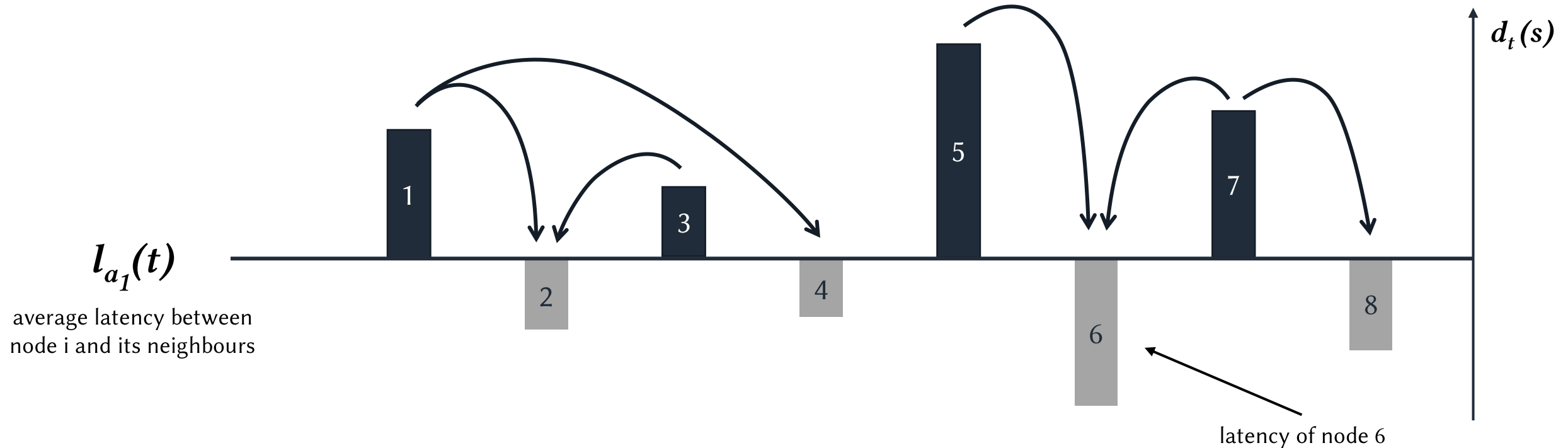
For doing so, we compute the derivative of the state equation defined earlier:

$$\dot{x}_i(t) = - \sum_{j \in V} a_{ij} \lambda_i \dot{m}_{ij}(t) + \sum_{j \in V} a_{ji} \lambda_j \dot{m}_{ji}(t)$$

The result is a system of ordinary differential equations (ODE) which can be solved once we have defined the **dynamic** of the **migration ratios**

# Levelling Property

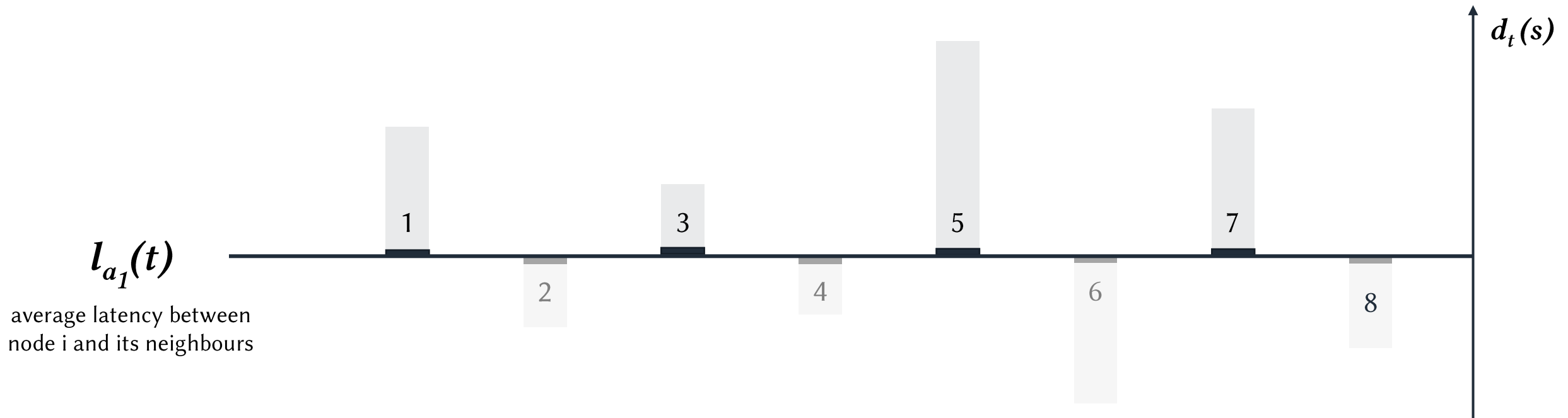
Definition of the dynamic of  $m_{ij}(t)$



The main is that the nodes that are **above the average** must give away part of their load, the others which are **below the average** have only to receive load

# Levelling Property

Definition of the dynamic of  $m_{ij}(t)$

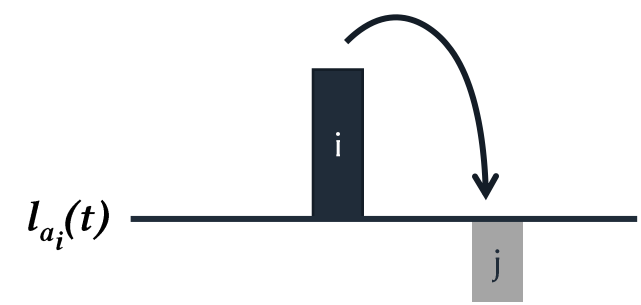


The main is that the nodes that are **above the average** must give away part of their load, the others which are **below the average** have only to receive load



# Levelling Property

Definition of the  $m_{ij}(t)$  dynamics



There are three main points to keep in mind and that translates in the dynamic. The migration must be performed between node i and j only if:

1.  $[\alpha]$  latency of i  $l_i(t)$  is greater than the average latency  $l_{a_i}(t)$
2.  $[\beta]$  latency of i  $l_i(t)$  is greater than latency of j
3.  $[\gamma]$  latency of j  $l_j(t)$  is lesser than the average latency  $l_{a_i}(t)$

These conditions translate into three factors:

$$\dot{m}_{ij}^{\alpha}(t) = \max \left[ 0, \frac{l_i(t) - l_{a_i}(t)}{l_i(t)} \right]$$

$$\dot{m}_{ij}^{\beta}(t) = \max \left[ 0, \frac{l_i(t) - l_j(t)}{l_{h_i}(t)} \right]$$

$$\dot{m}_{ij}^{\gamma}(t) = \max \left[ 0, \frac{l_{a_i}(t) - l_j(t)}{l_{k_i}(t)} \right]$$

The final ratio is then  $\dot{m}_{ij}(t) = \dot{m}_{ij}^{\alpha}(t) \cdot \dot{m}_{ij}^{\beta}(t) \cdot \dot{m}_{ij}^{\gamma}(t)$

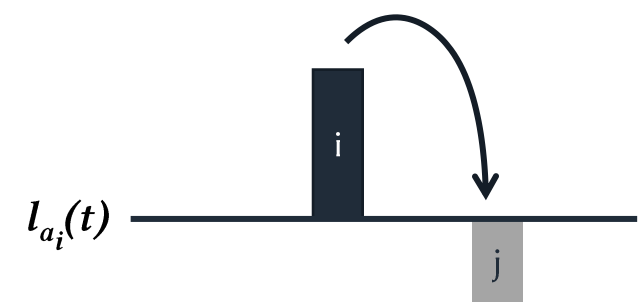
# Levelling Property

Definition of the  $m_{ij}(t)$  dynamics

## Condition $\alpha$

Latency of the node i higher than the average

$$\dot{m}_{ij}^{\alpha}(t) = \max \left[ 0, \frac{l_i(t) - l_{a_i}(t)}{l_i(t)} \right]$$



The core idea is that the dynamic of the ratio must (i) tend to 0 for  $t \rightarrow \infty$ , (ii) **be 0** when the **condition is met** and (iii) always  $< 1$

## Condition $\beta$

Latency of i greater than the latency of j

$$\dot{m}_{ij}^{\beta}(t) = \max \left[ 0, \frac{l_i(t) - l_j(t)}{l_{h_i}(t)} \right]$$

Summation of all the differences

$$l_{h_i}(t) = \max \left[ 0, \sum_{j \in V} l_i(t) - l_j(t) \right]$$

## Condition $\gamma$

Latency of j is lesser than the average latency

$$\dot{m}_{ij}^{\gamma}(t) = \max \left[ 0, \frac{l_{a_i}(t) - l_j(t)}{l_{k_i}(t)} \right]$$

Summation of all the differences

$$l_{k_i}(t) = \max \left[ 0, \sum_{j \in V} l_{a_i}(t) - l_j(t) \right]$$

# Levelling Property

Solution of the  $m_{ij}(t)$

Once the trajectory  $x_i(t)$  is found for every  $i$ , we can derive the trajectory of  $\dot{m}_{ij}(t)$ . From that, it will suffice to compute (numerically) the integral:

$$m_{ij}(t) = \int_0^t \dot{m}_{ij}(\xi) d\xi$$

The integral is done up to  $t = t^*$  that is the time in which the system converges to the same latency and the dynamics of the ratios is stopped.

# Results

## Trajectories

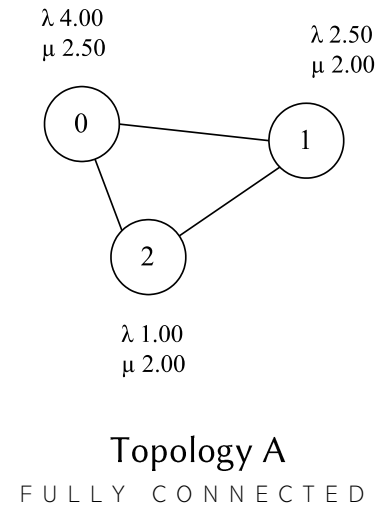


Figure 2.1 Latency for Topology A

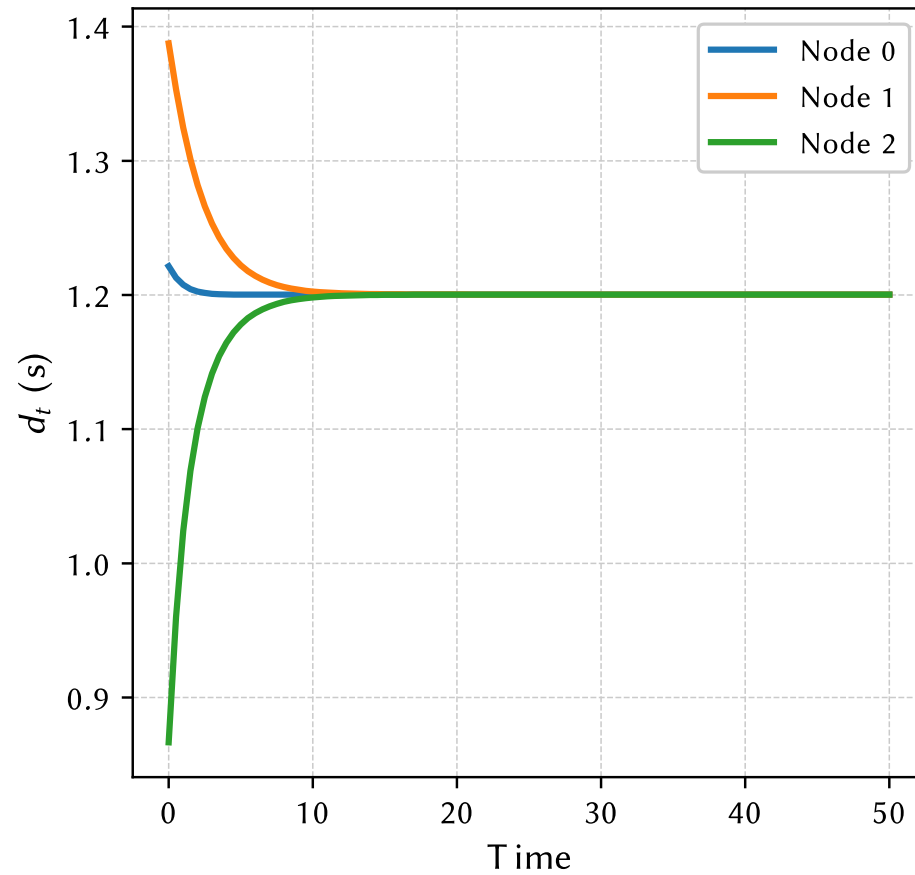
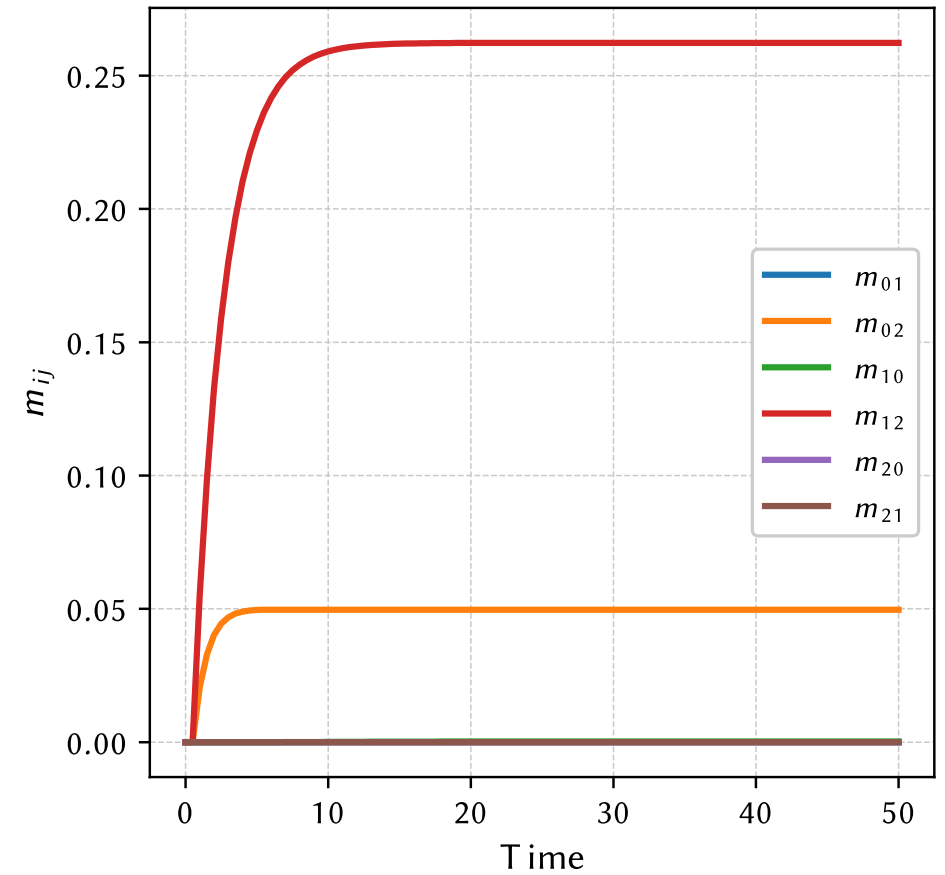


Figure 2.2 Migration ratios for Topology A



# Results

## Trajectories

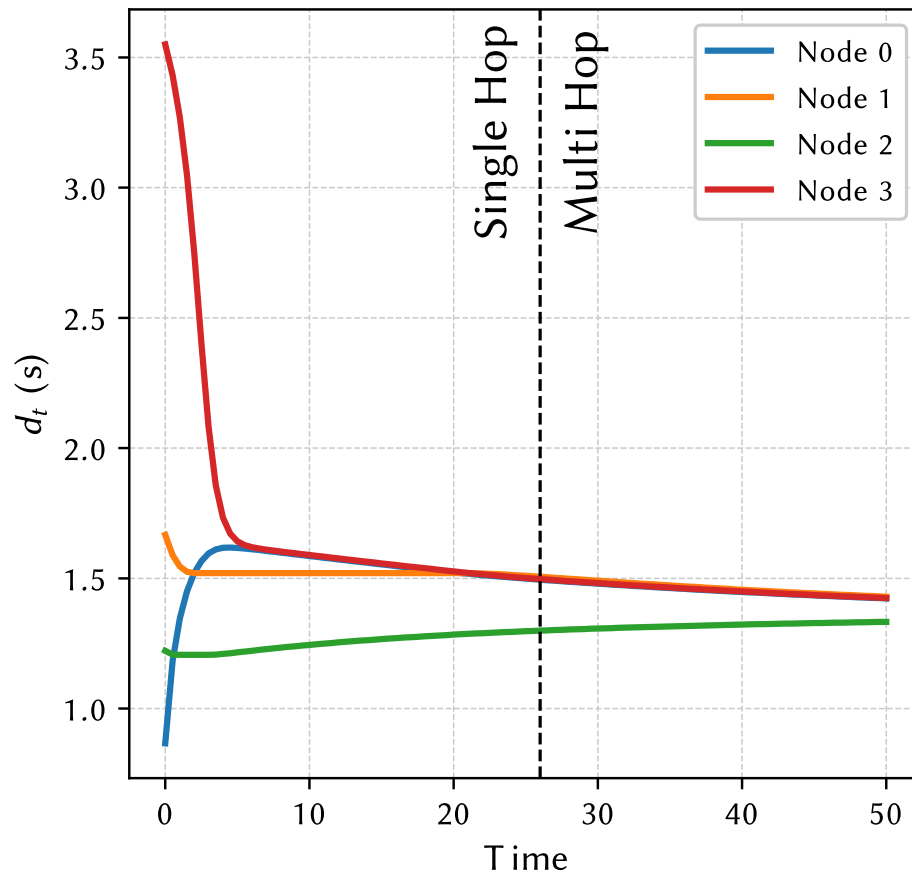
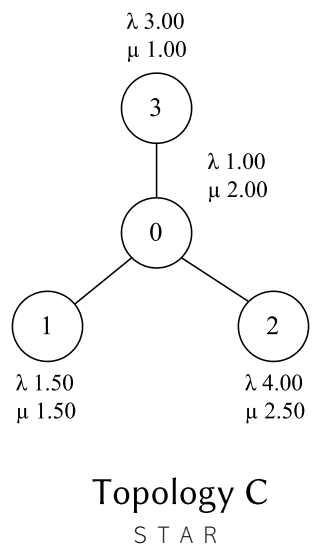


Figure 2.5 Latency for Topology C

$$\text{for some } i \quad \sum_{j \in V} m_{ij}(t) > 1$$

$$x_i(t) = \lambda_i - \sum_{j \in V} a_{ij} \lambda_i m_{ij}(t) + \sum_{j \in V} a_{ji} \lambda_j m_{ji}(t)$$

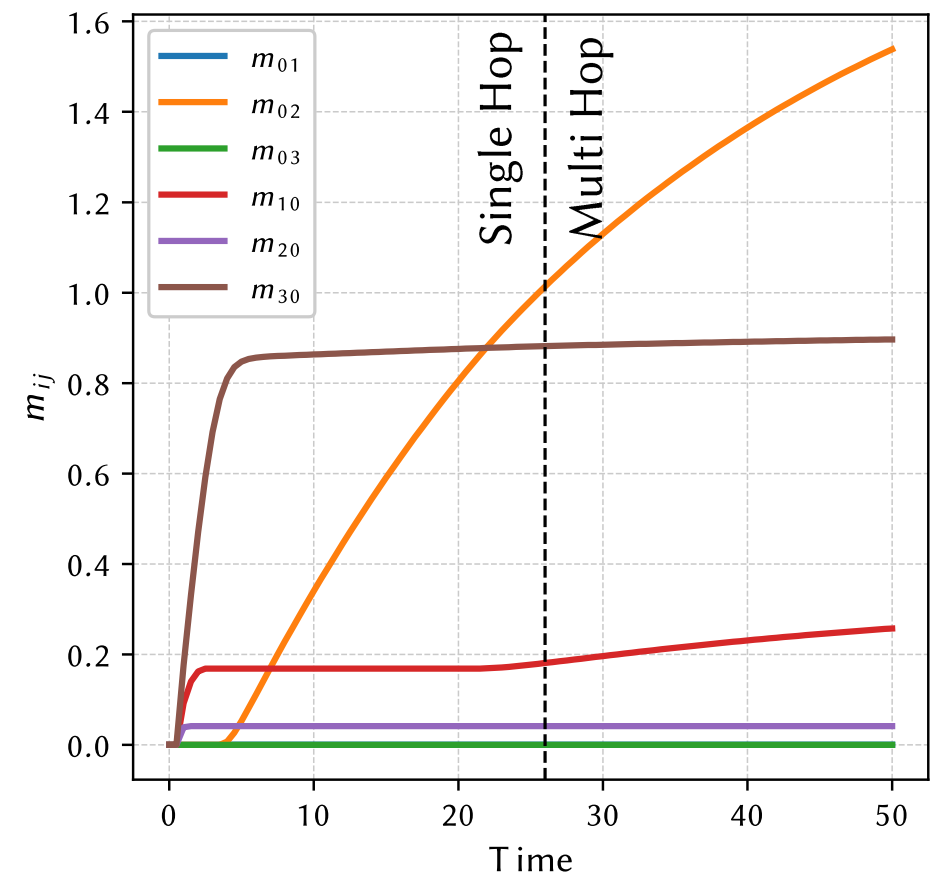


Figure 2.6 Migration ratios for Topology C

**3**

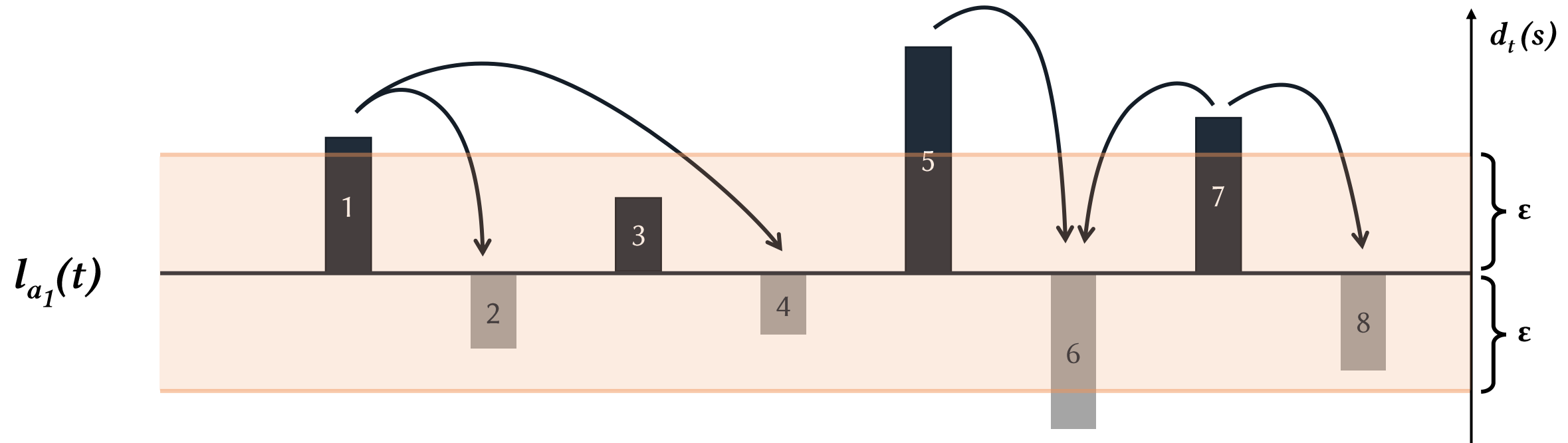
# **Adaptive Heuristic**

*The 25th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*



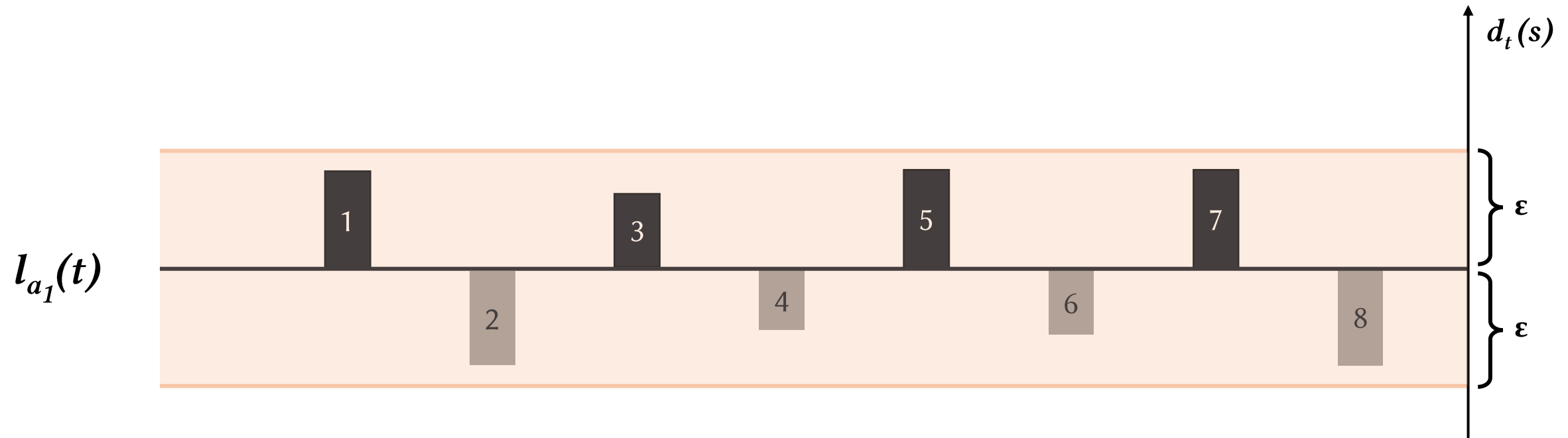
# Adaptive Heuristic

The key idea of the adaptive heuristic is to find the migration ratio **adaptively**.



The heuristic progressively adjust the migration ratio towards each node until the latency reaches a **balanced zone** of size  $2\epsilon$

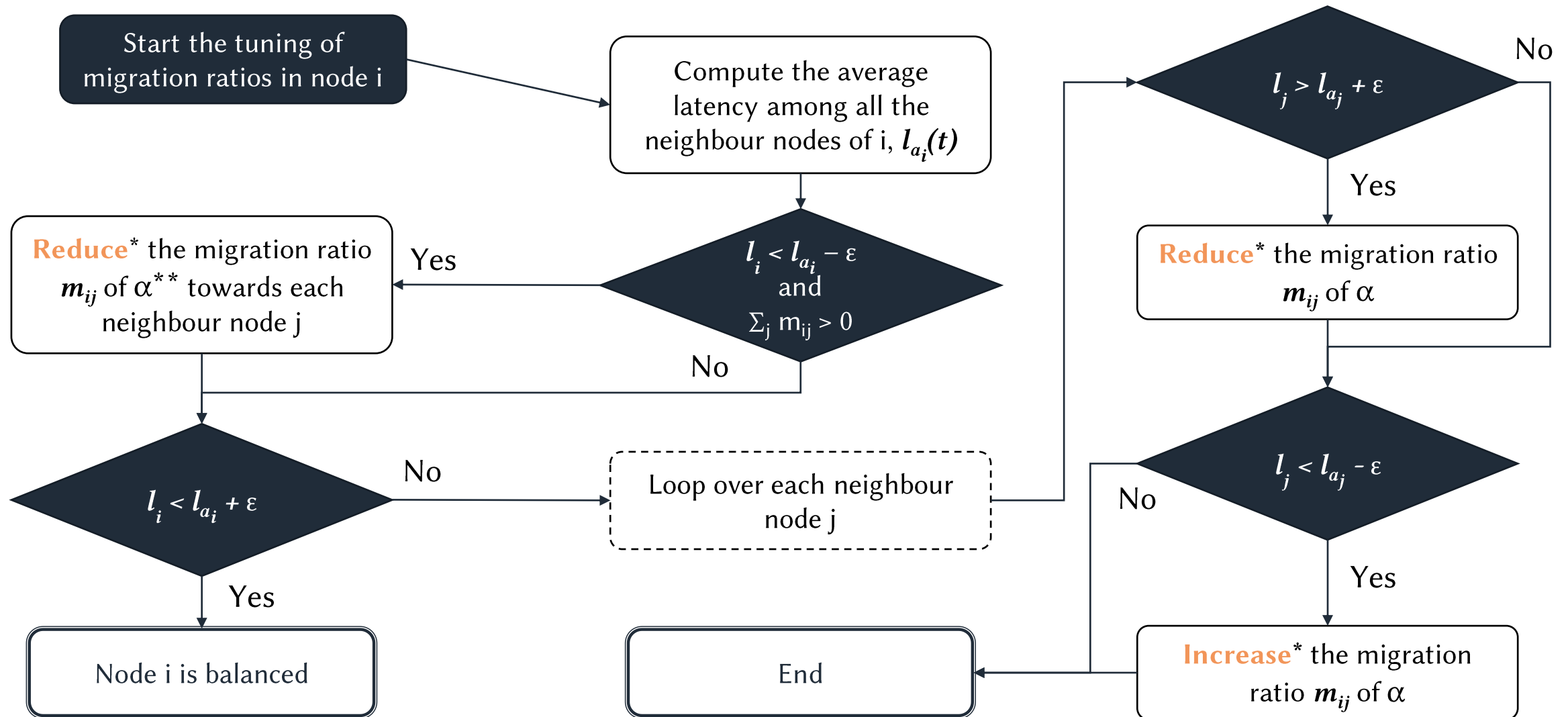
# Adaptive Heuristic



Once the zone is reached the migration ratios are no more adjusted, unless a node is too much under the zone and it must reduce the migration ratios

# Adaptive Heuristic

\*the reducing or the increasing of the migration ratio is done only if it does not exceed 1.0 and 0.0  
\*\* $\alpha$  is set to 0.01



4

# Experimental Results

*The 25th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*

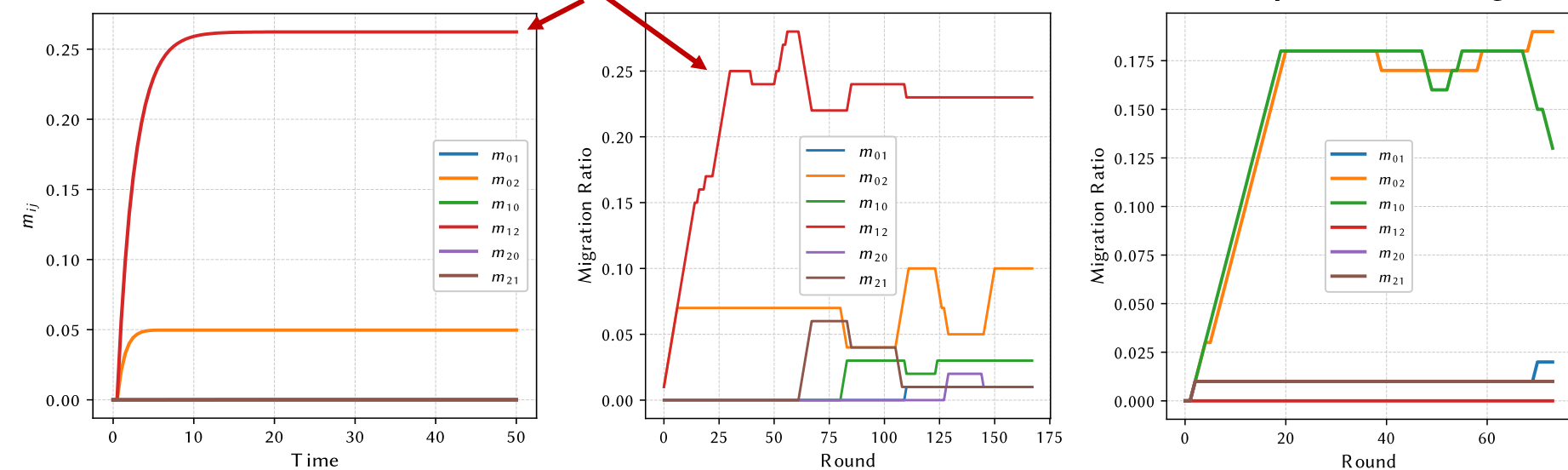
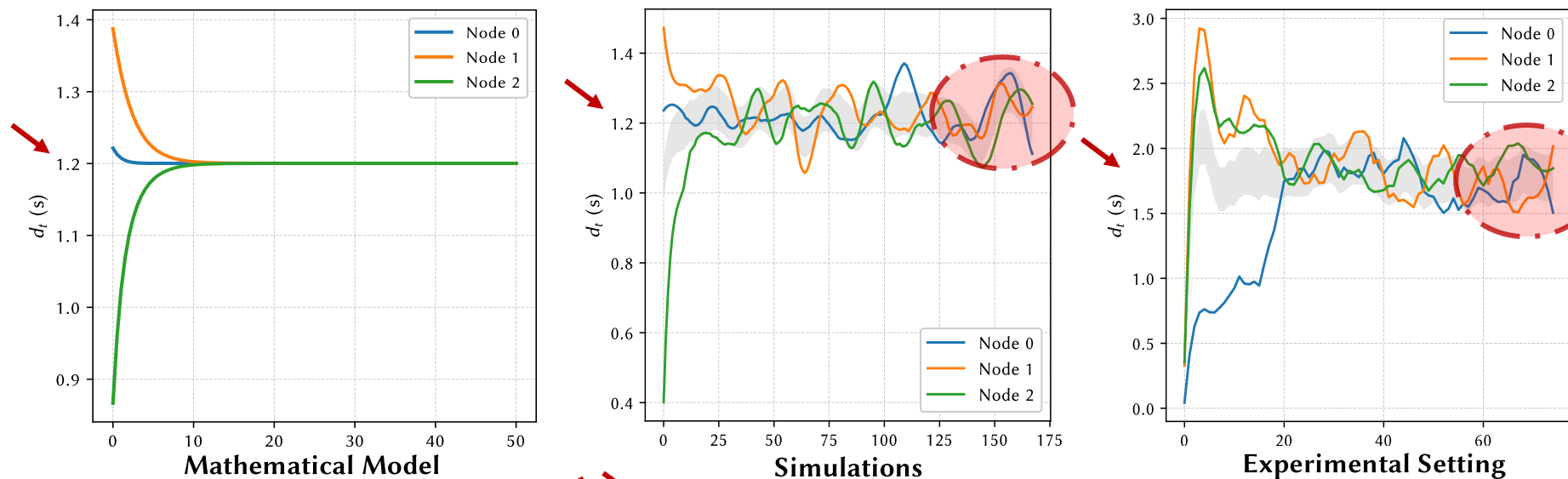
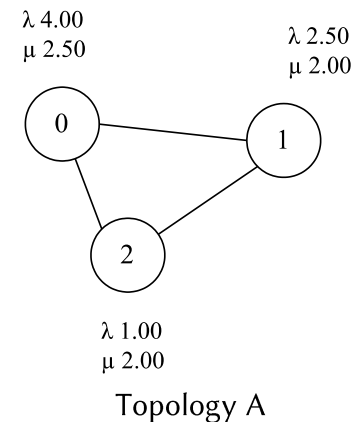
# Experimental Results

The proposed heuristic has been tested in two environments:

1. a **simulated environment**, based on the discrete event simulator called Simpy (Python library), no communication latency has been considered
2. a **real environment**, in which the algorithm has been implemented in a testbed of 12 Raspberry Pis connected with Gigabit ethernet and deployed with OpenBalena framework

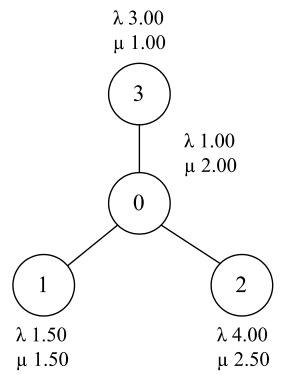
What follows is the comparison of latency and migration ratios between the mathematical model, the simulations and the real environment.

# Results





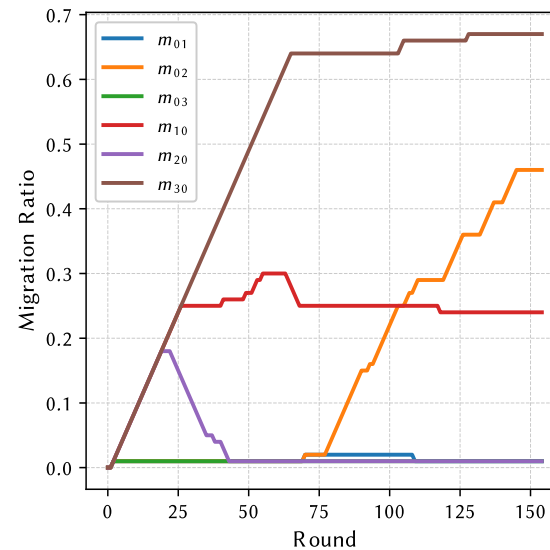
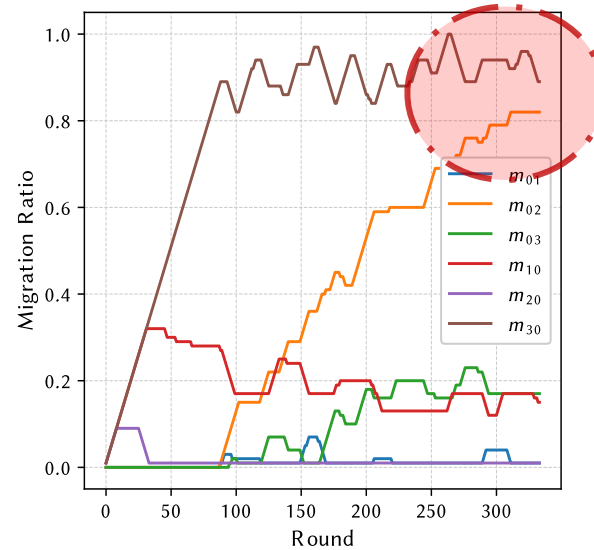
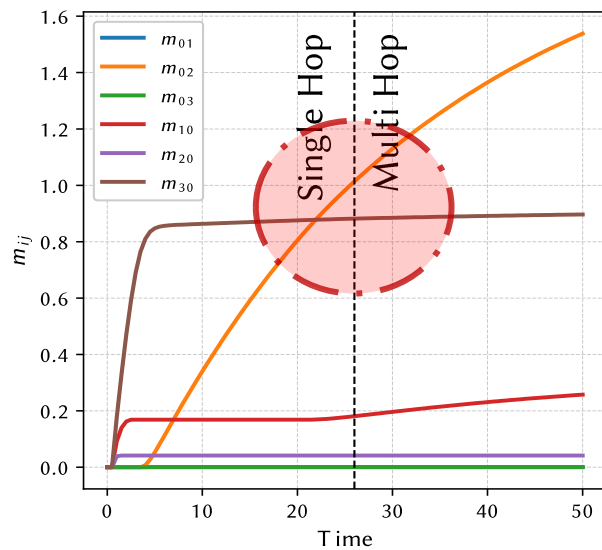
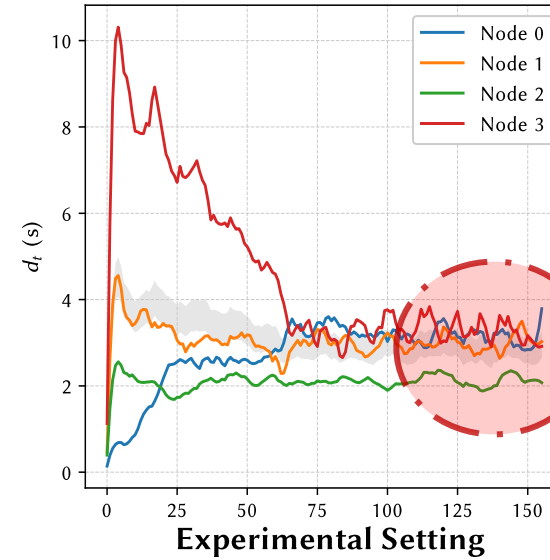
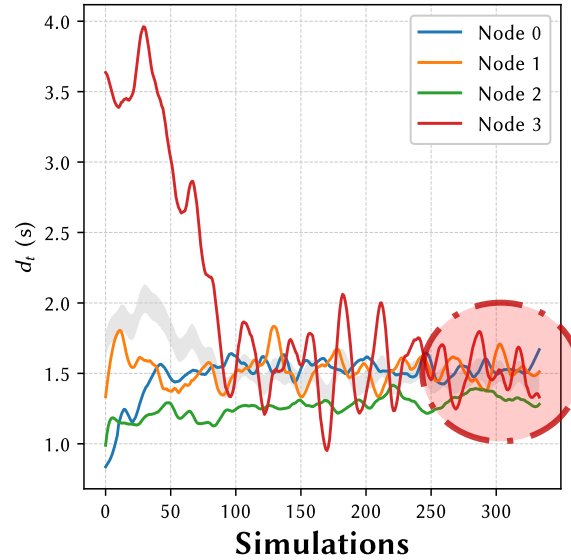
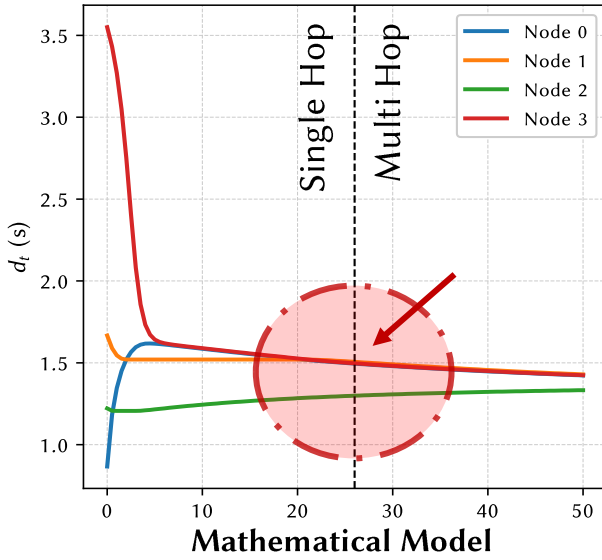
# Results



Topology C

**Figure 4.5** Latency for Topology A from the mathematical model, the simulation and the experimental setting

**Figure 4.6** Migration ratios for Topology A with the proposed adaptive heuristic from the mathematical model, the simulation and the experimental setting



# Conclusions

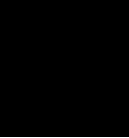
# Conclusions

In this work we presented:

- the **dynamic model** of a system of  $n$  nodes which level their latency
- an **adaptive heuristic algorithm** which allows to find the *migration ratios* in a fully distributed environment with no central entity or scheduler
- **results** from the implementation of the heuristic in simulation and in a cluster of 12 Raspberry Pi nodes

The results showed that model, simulations and experimental results are in aligned in finding a set of migration ratios which level the latencies, when this is feasible. However, some points need further study:

- design a mathematical model which includes the forwarding/communication **latency**
- design a *more accurate* model **node**, since M/M/1/K may be not enough
- consider the load from clients  $\lambda$  as **not fixed** over time,  $\lambda(t)$



# A Latency-Levelling Load Balancing Algorithm for Fog and Edge Computing

Gabriele Proietti Mattia, Marco Magnani, Roberto Beraldi

talk and presentation

**Gabriele Proietti Mattia**

[gpm.name](#)

Department of Computer, Control and Management Engineering “Antonio Ruberti”, Sapienza University of Rome, Italy