**The 1st Workshop on Adaptive, Learning PervAsive Computing Applications (ALPACA 2022)**
21st March, 2022
Virtual

# On real-time scheduling in Fog computing: A Reinforcement Learning algorithm with application to smart cities

Gabriele Proietti Mattia, Roberto Beraldi

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

proiettimattia@diag.uniroma1.it • gpm.name

Sapienza Università di Roma

DIAG
Dipartimento di Ingegneria informatica, automatica e gestionale Antonio Ruberti

# Outline

1

# Introduction

# Context

## Fog Computing

~50ms*

Average round-trip-time cloud-to-device


*https://geekflare.com/google-cloud-latency/

# Context

## Fog Computing

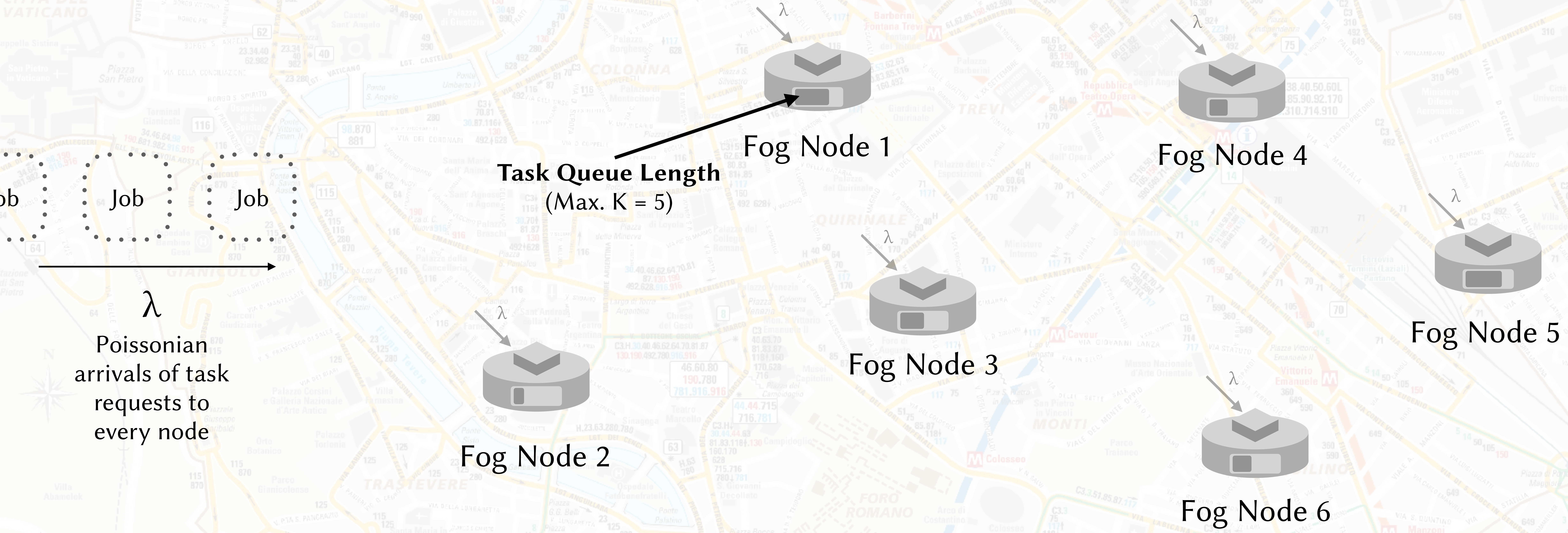
~10ms*

Average round-trip-time fog-to-device


*https://geekflare.com/google-cloud-latency/

# Context

## The Power-of-choice in Fog Computing

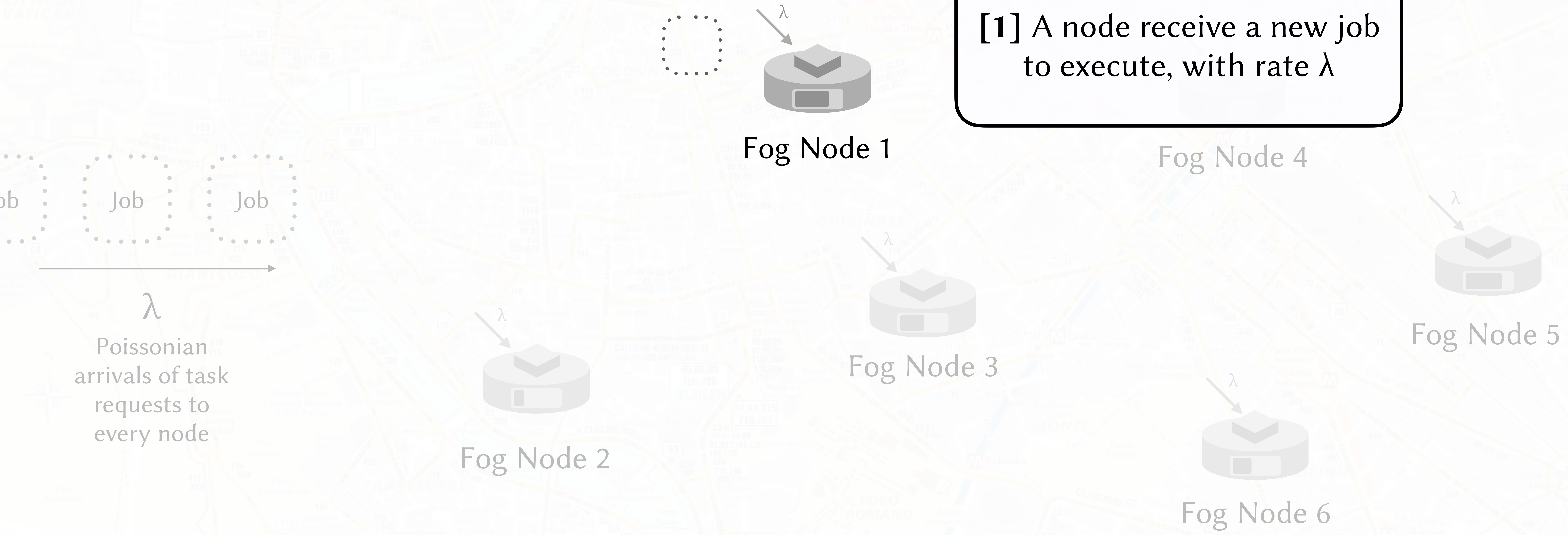



Suppose that devices can communicate with each other and have the **same computing power.**
They can act both as schedulers ( ) and workers ( ).

# Context

## The Power-of-choice in Fog Computing

**[1]** A node receive a new job to execute, with rate λ

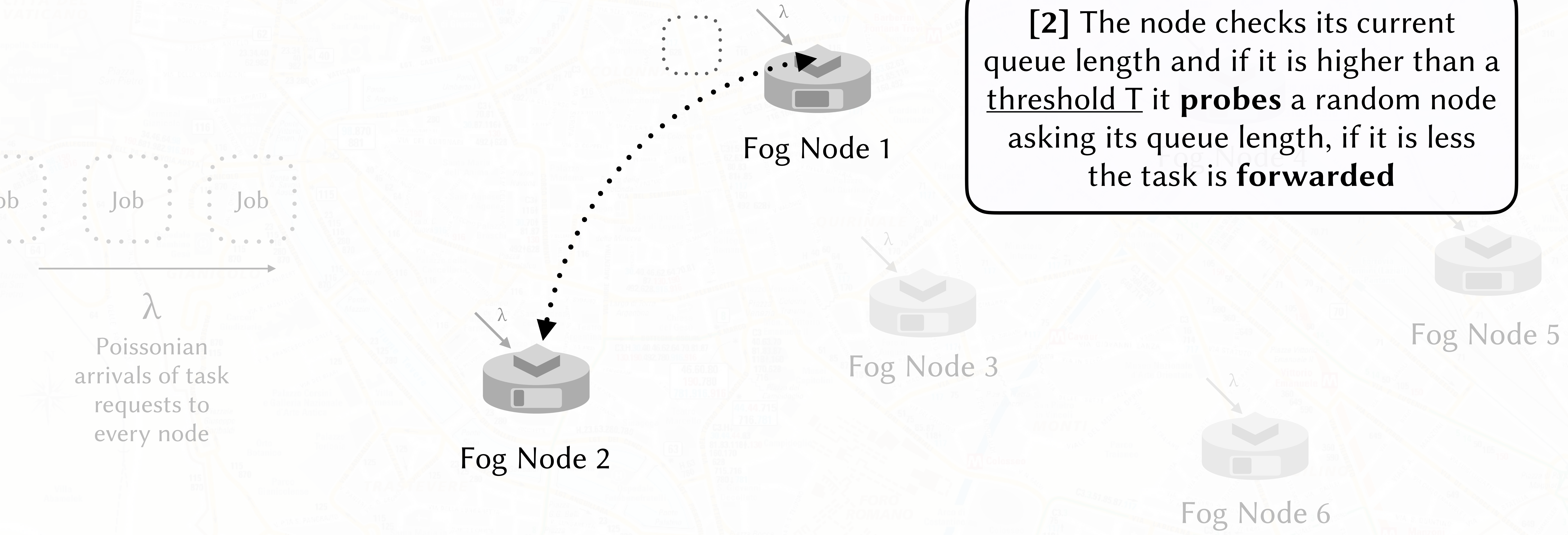



Suppose that devices can communicate with each other and have the **same computing power.**
They can act both as schedulers ( ) and workers ( ).

# Context

## The Power-of-choice in Fog Computing

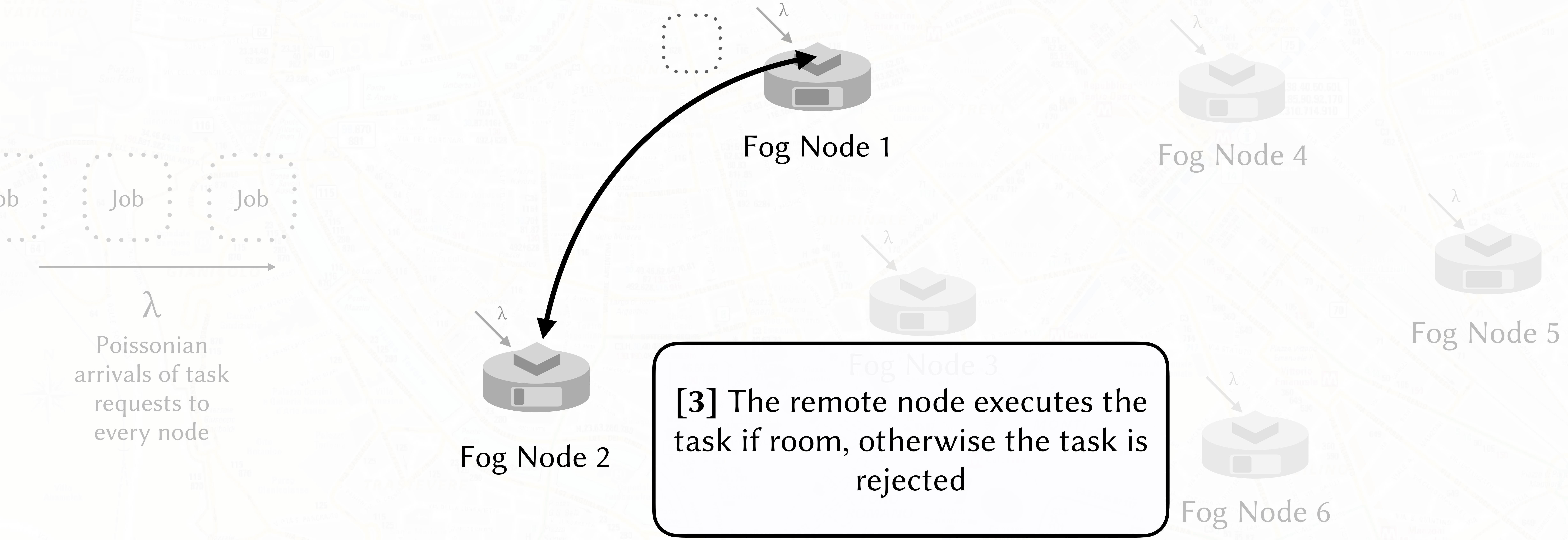
**[2]** The node checks its current queue length and if it is higher than a threshold T it **probes** a random node asking its queue length, if it is less the task is **forwarded**




Suppose that devices can communicate with each other and have the **same computing power.**
They can act both as schedulers ( ) and workers ( ).

# Context

## The Power-of-choice in Fog Computing




The power-of-choice strategy is proven to be very performant but has limitations

# Challenge

The Power-of-choice in Fog Computing • Limitations

However, the PoC scheduling policy (i.e. when to trigger the probing) has some limitations:

- it is a **fixed step function** of the current load;
- it is also **fixed over time** and it cannot react to load variation on the nodes, and finally;
- it doesn't take **task heterogeneity** into account.

The purpose of this work is to overcome these limitations by designing a **dynamic scheduling policy** based on the Reinforcement Learning (RL).

# Contribution

In the light of these challenges, the main contributions of this work are:

- Design of a **decentralised RL-based scheduling algorithm** to be implemented in every fog node that is able to choose the best scheduling decision according to the current load situation
- Study of a **geographic setting** which involves six fog nodes deployed in the city of New York and in which the algorithm can be deployed.
- **Simulation results** on a delay-based simulator which prove the efficiency of the algorithm in a previously defined geographic environment compared to the classic power-of-choice strategy

# State-of-the-art

- R. Beraldi and G. P. Mattia, "*Power of random choices made efficient for fog computing*" (2020) presents the power-of-choices in fog computing with limitations as **described earlier**
- L. Ale et al. in "*Delay-aware and energy-efficient computation offloading in mobile edge computing using deep reinforcement learning*" (2021) the authors present a Deep Reinforcement Learning approach, based on Q-Learning for selecting the best edge server for offloading in order to minimise the **energy consumption**
- M. K. Pandit et al. in "*Adaptive task scheduling in iot using reinforcement learning*" (2020) based again on **two DNNs**, but they are used for two different decisions, the first one is in charge of deciding if the task should be offloaded to the cloud, but if not, the second decision level chooses the best suitable Fog node
- S. Park et al. in "*Real-time scheduling using reinforcement learning technique for the connected vehicles*" (2018) proposes a vanilla RL approach but is not **online and focused on vehicles**
- T. Sen et al. in "*Machine learning based timeliness-guaranteed and energy-efficient task assignment in edge computing systems*" (2019) propose again a vanilla RL approach but is based on a **different environment** and **focused on energy consumption**

The main points of novelty of our work resides on the facts that we focus on **real-time tasks** (duration ~20ms) training the learner according to the hit of a task **completion deadline**, we focus on **online scheduling** and we set the study by using a **geographic approach**.

2

# System Model

# RL Rationale

The idea is to make each scheduler of a cluster a learner agent, model the problem as a Markov Decision Process and solve the learning task with a RL framework. All of the following entities must be defined.

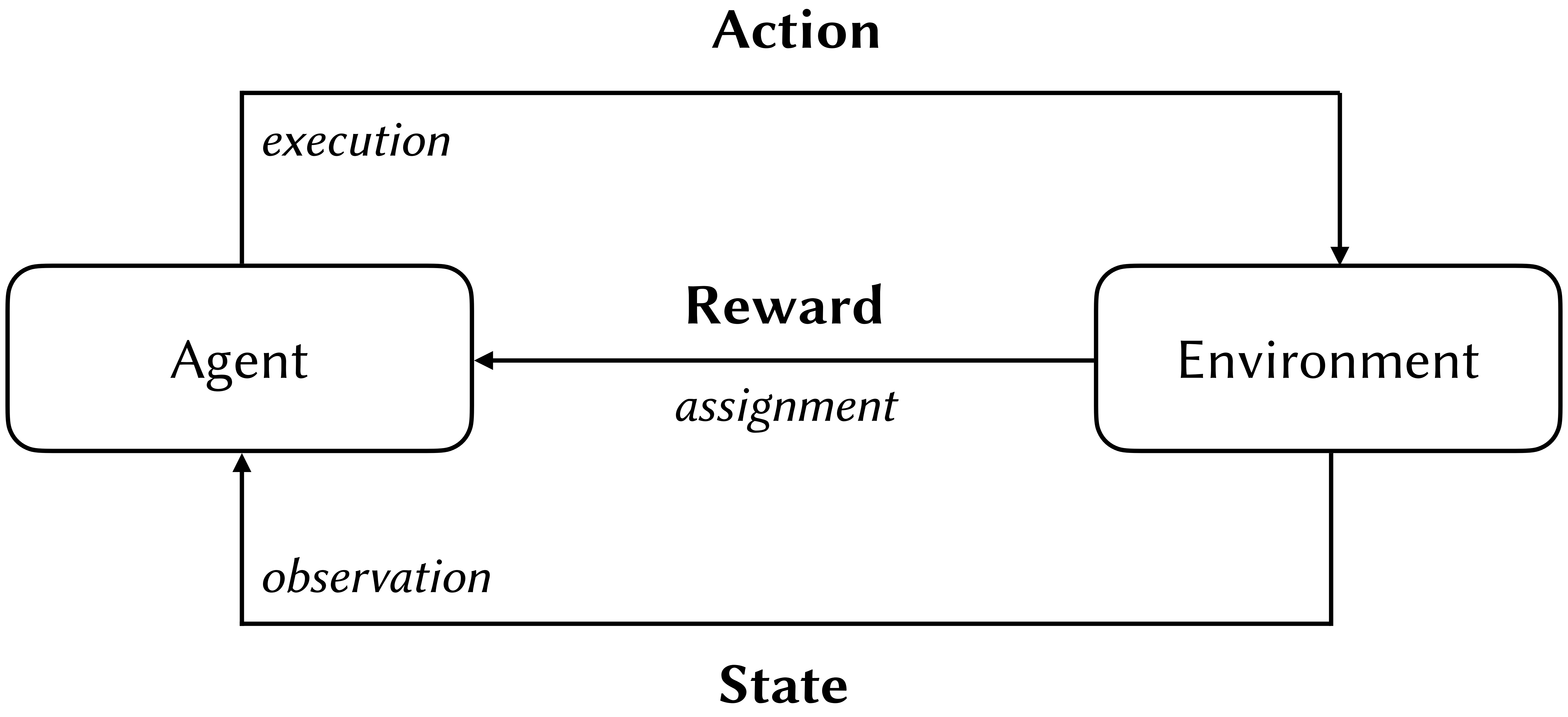
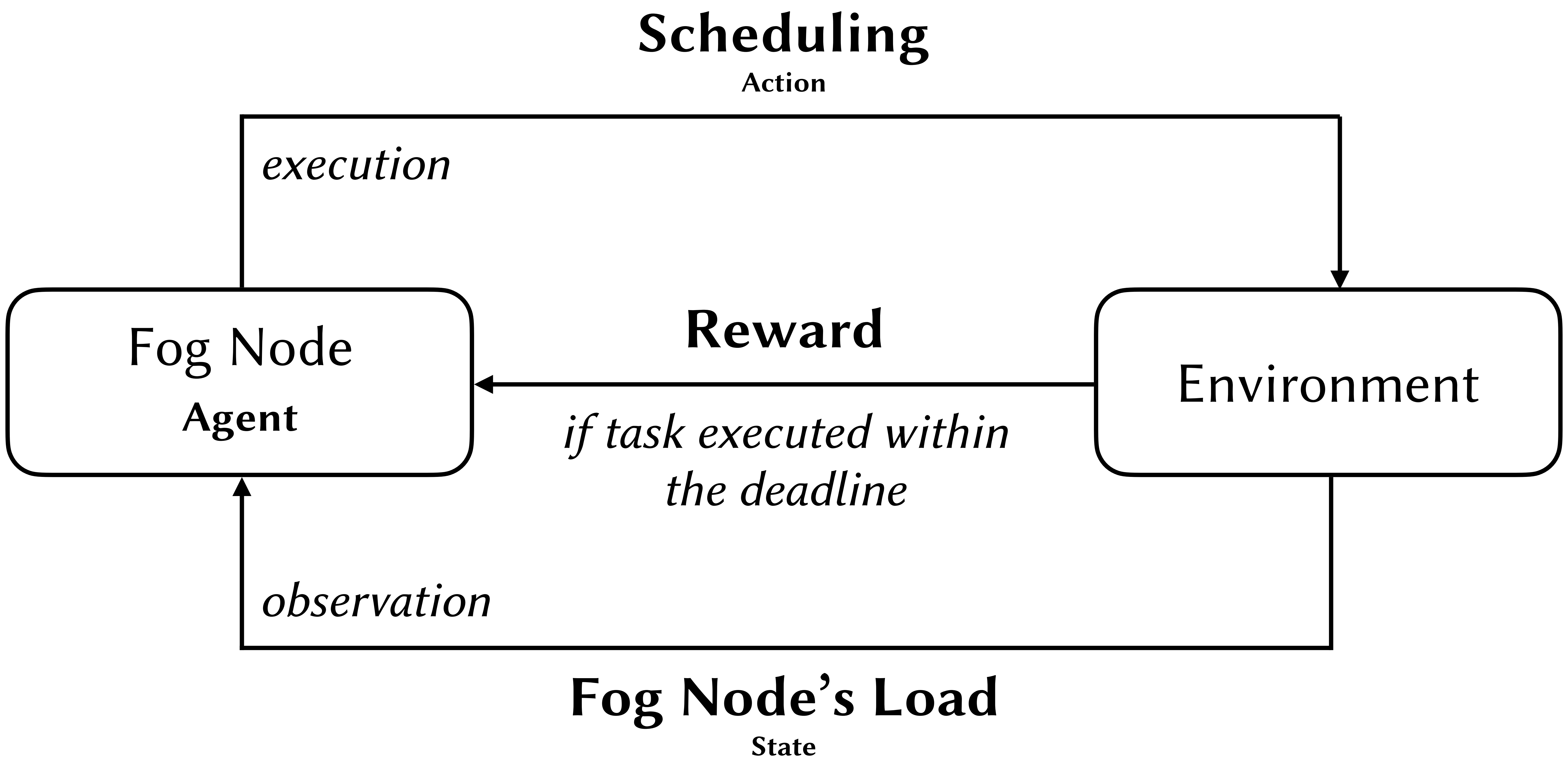



**Figure 2.1** The classic Markov Decision Process representation

# RL Rationale

The action is taken by the scheduler module of the node, and it can be
**schedule locally** or **forward the task** to another node




Namely the number of tasks that the node is currently executing

**Figure 2.1** The classic Markov Decision Process representation with the assigned entities

# State and Actions

Upon the arrival of a task execution request to the cluster *i* the action that can be performed by the scheduler is a scheduling action, one of the following:

- reject
- forward to cloud
- forward to random node
- forward to neighbour node 1
- ...
- forward to neighbour node n

$\mathcal{A}_i$

$\mathcal{A}'_i$

The actions are grouped in **two sets** because we performed two kinds of experiments

When a task is forwarded we wait for its completion in order to derive the **reward**.

# Tasks and delay model

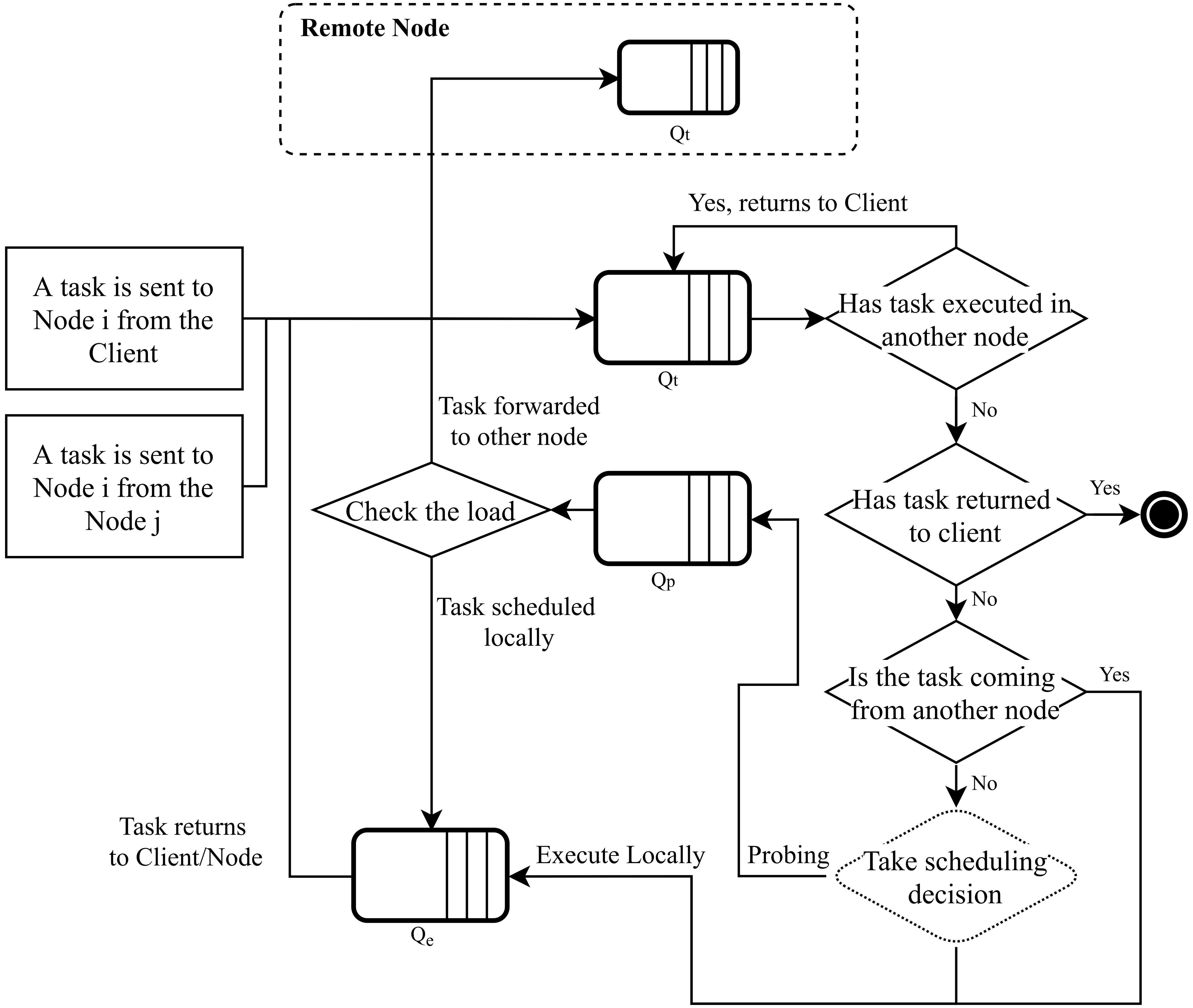



**Figure 2.2** The logic of the delay model

In our model, to each node three queues are attached:

- Qt the **transmission** queue used for simulating the task transmission
- Qp the **probing** queue used for simulating the request of the state to another node
- Qe the **execution** queue used for simulating the execution of the task

# Performance Parameter

## Reward definition

To each task we associate an **execution deadline** T and the reward is assigned as, given W the completion time:

$$R_j(s,a) = \begin{cases} 1 & \text{if } W \leq T \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The performance taken into consideration is $\iota$ that is the reward gained in every second, namely the number of jobs that are executed within the deadline in a second.

# Geographic Setting

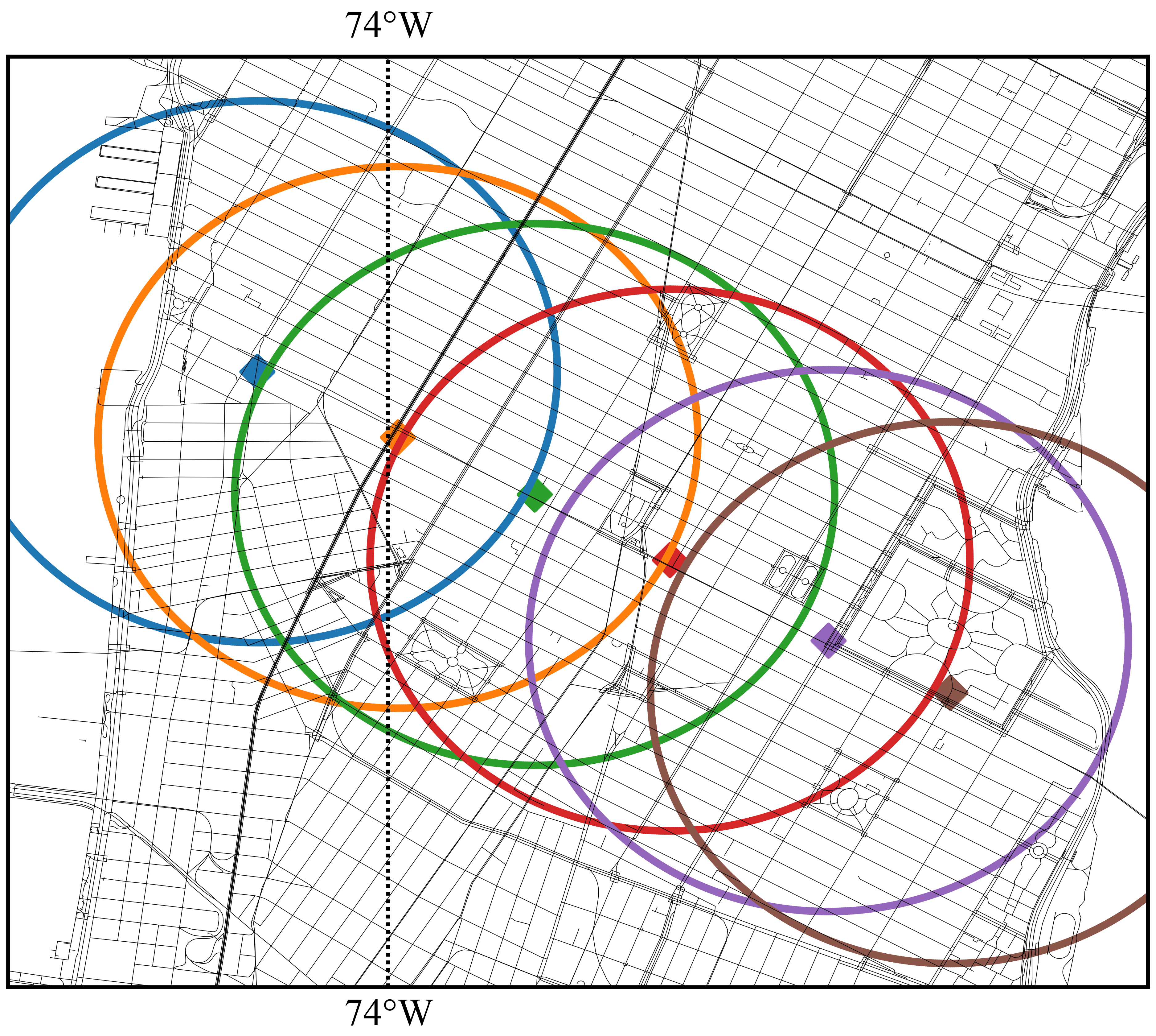



**Figure 2.3** Fog nodes position (diamond symbols) in New York city used in the experiments, from left to right Node 0 to Node 5. The radius of the circle for each node is 1 km.

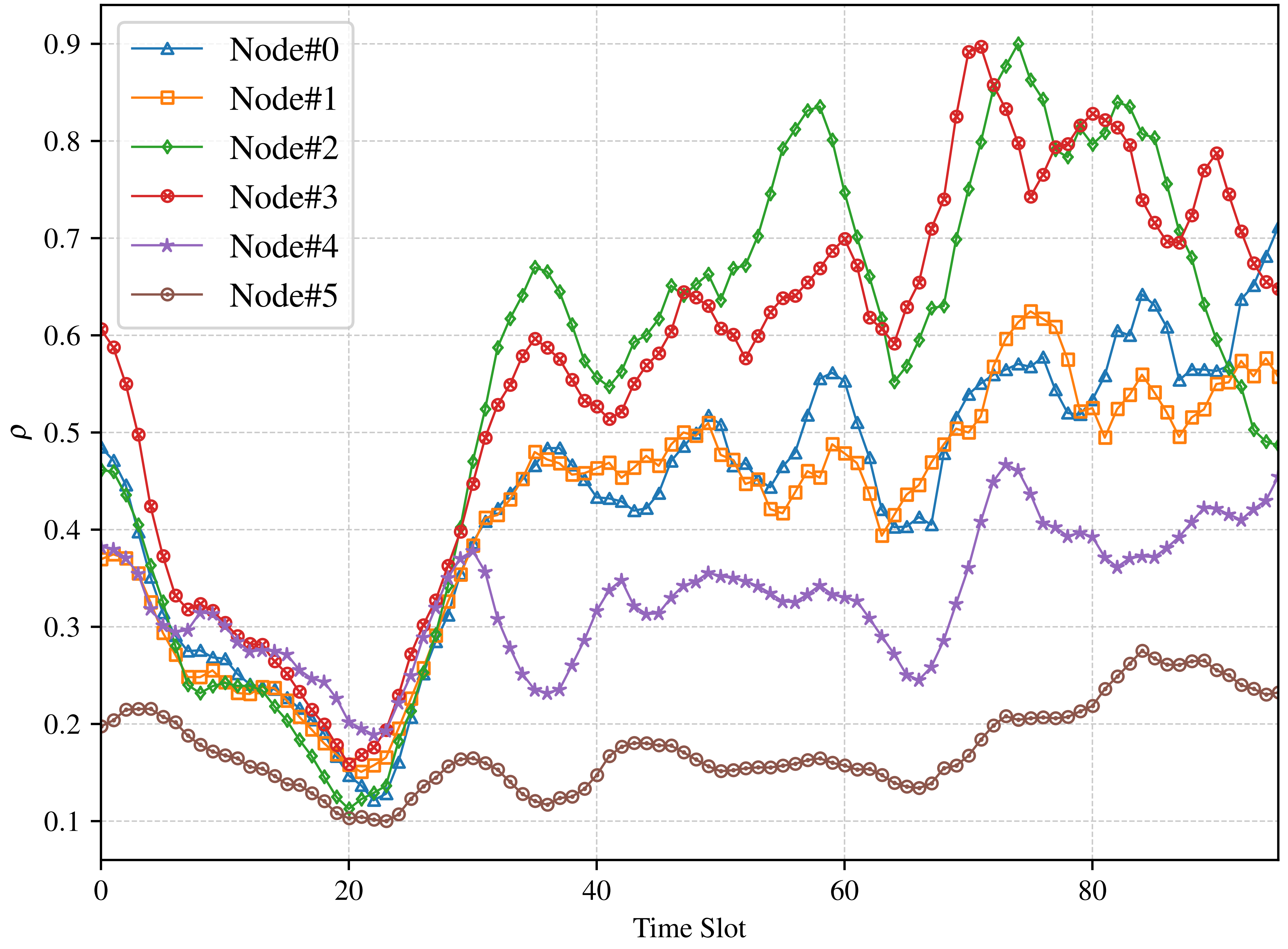
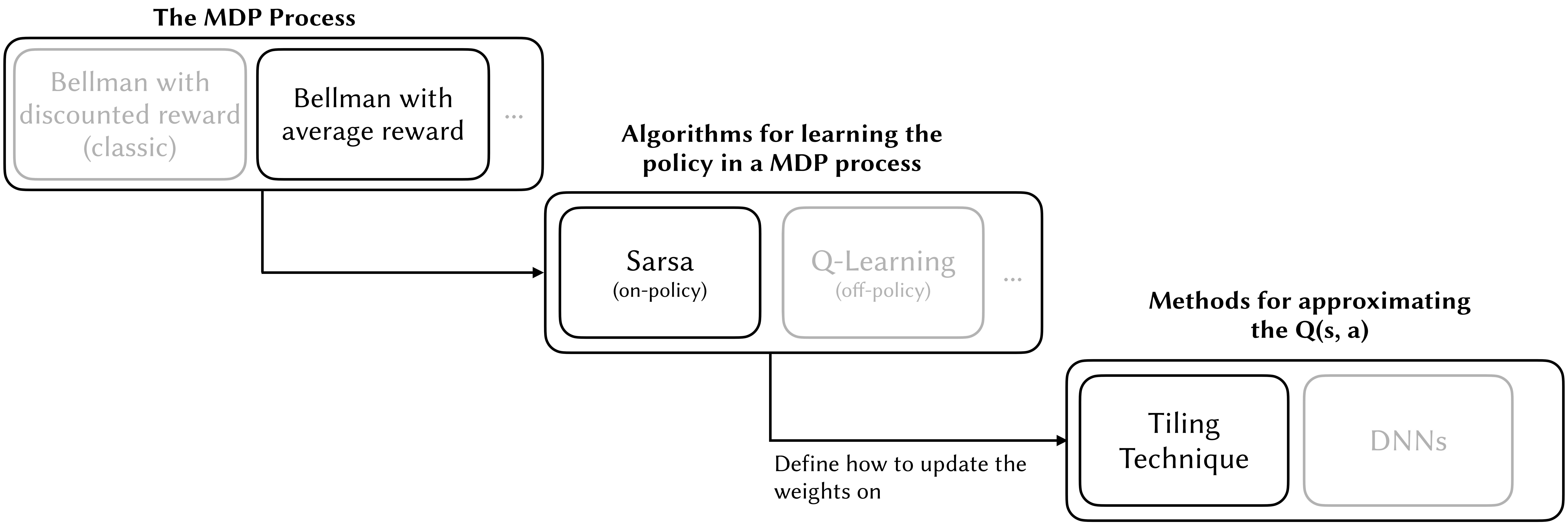



**Figure 2.4** The average distribution of the traffic during the day for the picked Fog nodes.

3

# Reinforcement Learning for online scheduling

# RL Theoretical Stack




Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

# Time Differential Sarsa w/ avg. Reward

Decisions are taken by approximating the q(s,a) action-value function, that returns the value of an action *a* given the state *s*. For approximating the q(s,a) function we can take into consideration the difference between $q(s,a)_t$ and $q(s,a)_{t+1}$ that is defined as (supposed we are at t+1):

$$\delta_t = R_{t+1} - \bar{R}_{t+1} + \hat{q}(S_{t+1}, A_{t+1}, \vec{w}_t) - \hat{q}(S_t, A_t, \vec{w}_t)$$

The **immediate** reward, right after executing the task

The current **average** reward

The value of the action chosen in the current state by using the same policy

The value of the previous action (that lead to $S_{t+1}$) in the previous state $S_t$

4

# Results

# Validity Check

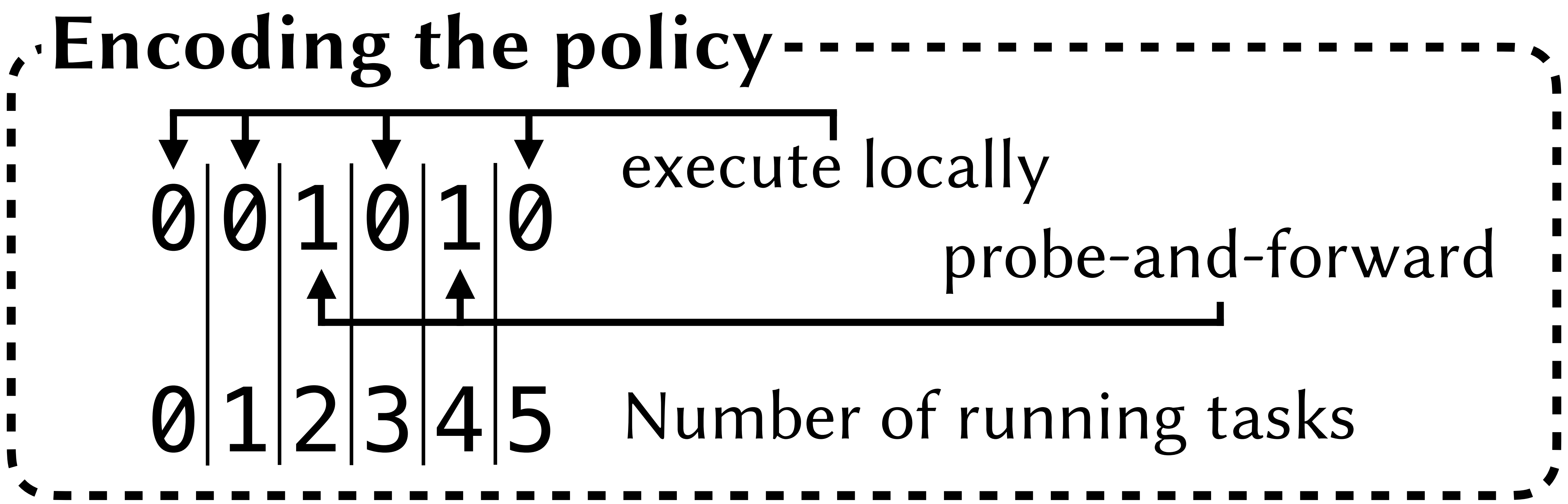
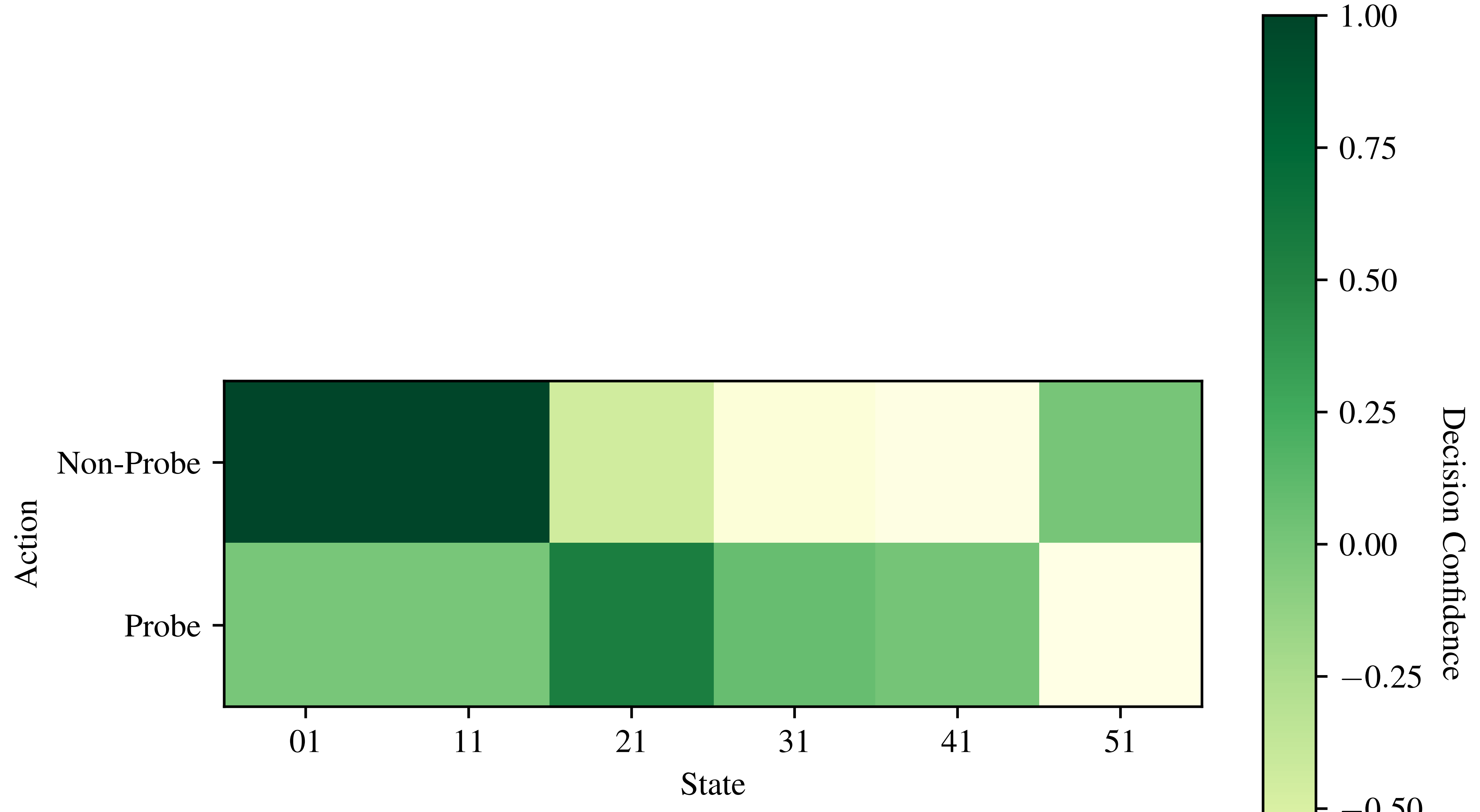




**Figure 4.1** Percentage of In-Deadline, Rejected and Forwarded tasks of all the possible policies with 6 nodes, a policy is encoded in binary where 1 means probing and 0 means executing locally. In this experiment $\rho = 0.6$, deadline is 0.043s and job duration is 0.020s.

# Validity Check

In these experiments the agent can only choose among three actions: reject, execute locally and probe-and-forward.




The learned policy is `001110`

Figure 4.2 The policy learned by the agent in the same setting of Figure 4.1

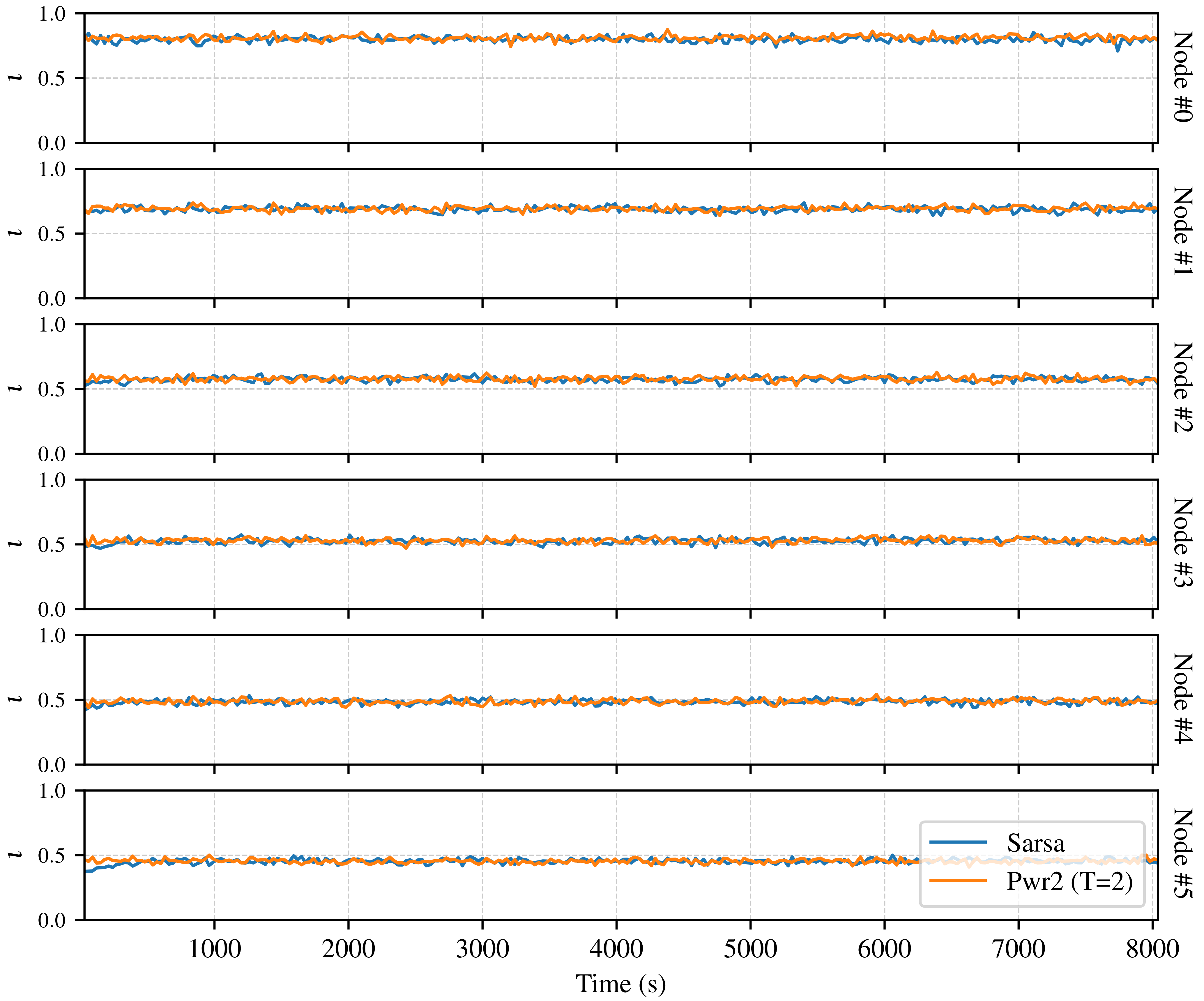
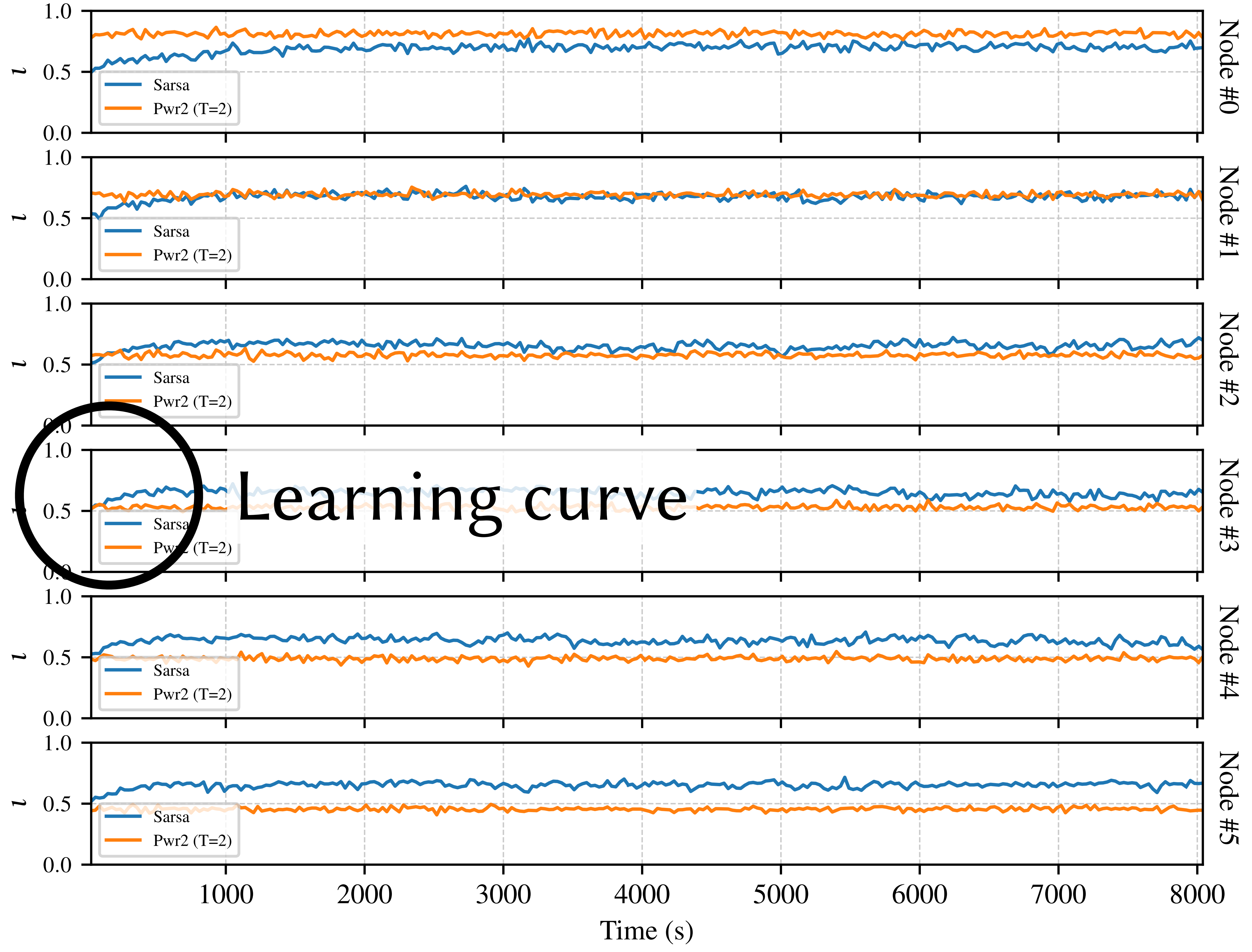



**Figure 4.3** Comparison between Sarsa and Pwr2: behaviour of the in-deadline rate ι for every node when the load is fixed and the same to every node

# Results

In these experiments the agent can only choose to: reject, execute locally, probe-and-forward and directly forward to a given node.



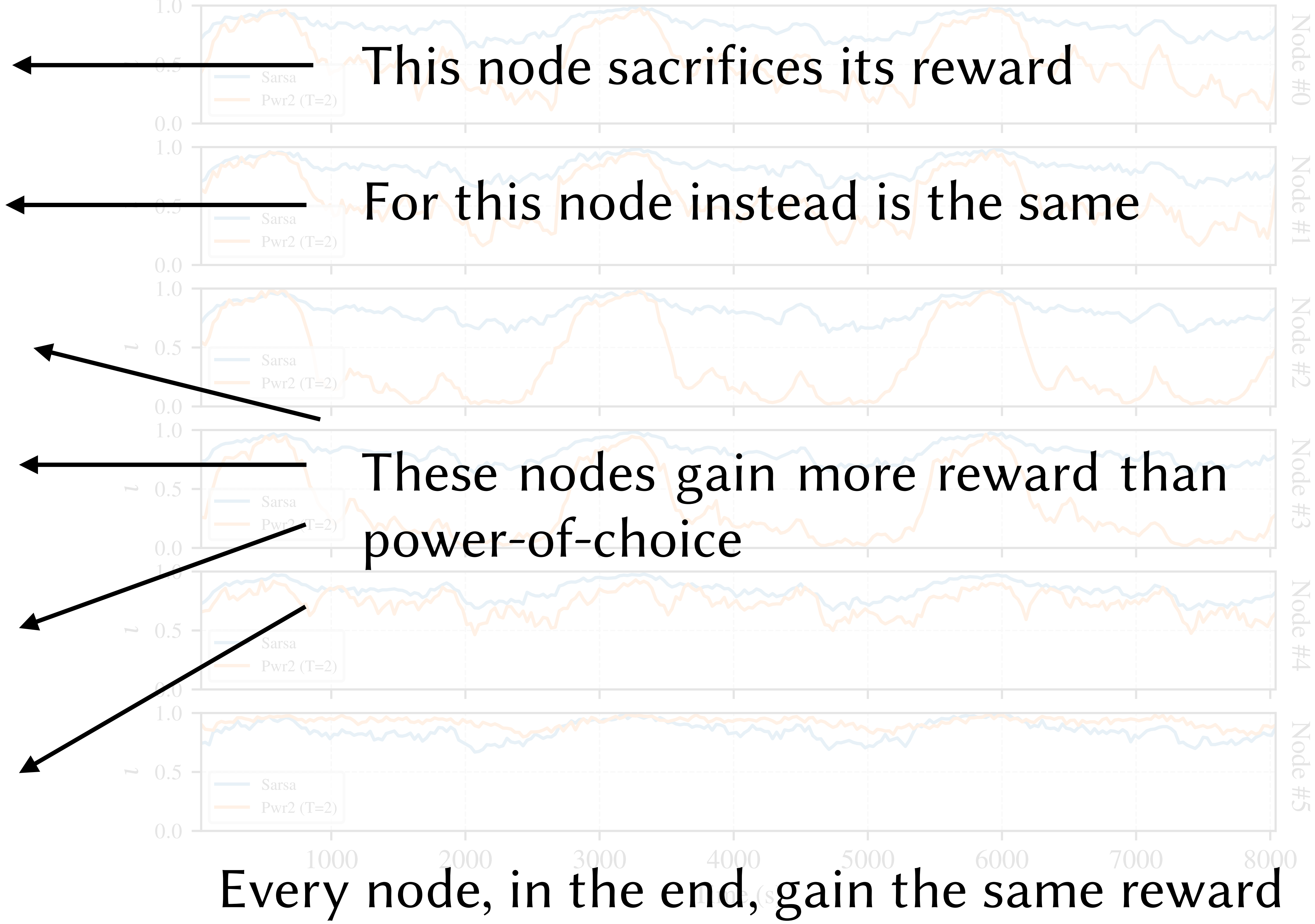
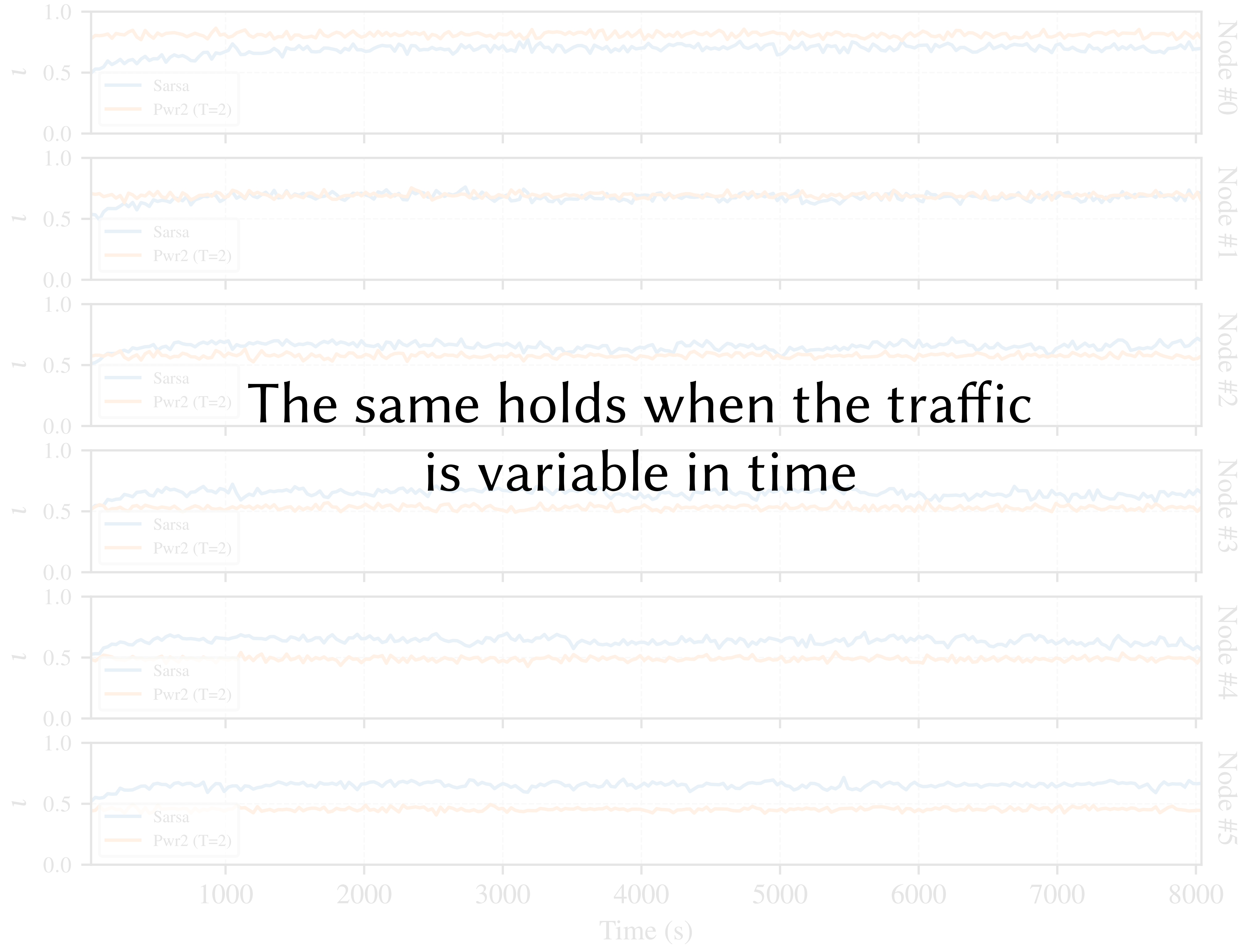




**Figure 4.4** Comparison between Sarsa and Pwr2: behaviour of the in-deadline rate $\iota$ for every node when load is fixed but heterogeneous

**Figure 4.5** Comparison between Sarsa and Pwr2: behaviour of the in-deadline rate $\iota$ for every node when the load is variable according to the geographic scenario

# Results

In these experiments the agent can only choose to: reject, execute locally, probe-and-forward and directly forward to a given node.




The same holds when the traffic is variable in time

**Figure 4.4** Comparison between Sarsa and Pwr2: behaviour of the in-deadline rate ι for every node when load is fixed but heterogeneous

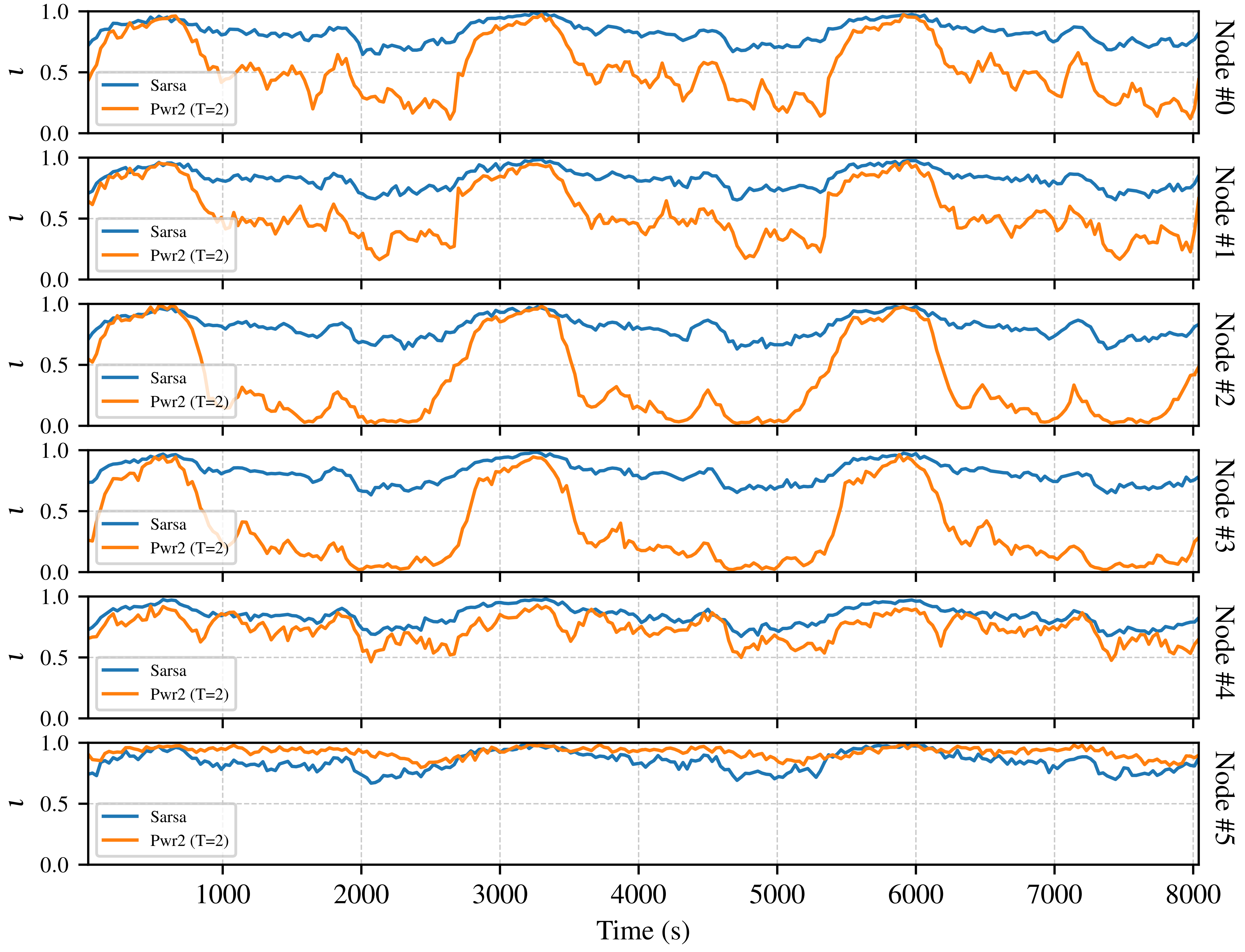



**Figure 4.5** Comparison between Sarsa and Pwr2: behaviour of the in-deadline rate ι for every node when the load is variable according to the geographic scenario

5

# Conclusions

# Conclusions & Future Work

- in the presented work we **designed** and **run in simulation** a fully distributed reinforcement learning based algorithm for dealing with online scheduling in the fog computing environment
- we showed how the algorithm can take a **step forward** the standard power-of-choice approach by inferring the best scheduling policy
- we showed how the approach can level the reward of every node making them not behaving selfishly

**Future work**

- consider variable **communication delay** between nodes
- increase the **complexity** of the state
- consider tasks with **different deadlines**
- study the **learning time**, how fast the algorithm learn the policy

# On real-time scheduling in Fog computing: A Reinforcement Learning algorithm with application to smart cities

Gabriele Proietti Mattia*, Roberto Beraldi*

TALK & PRESENTATION
Gabriele **Proietti Mattia**

*Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Italy