UNIVERSITÀ DI PISA

DEPARTMENT OF COMPUTER ENGINEERING

Edge Computing
Project Documentation

December 27, 2024

Team:

Cavedoni F.
Monaci M.
Pinna F.

ACADEMIC YEAR 2024/2025

Contents

Chapter 1

Introduction

A cellular network is composed of M base stations, placed within a 2D floorplan of size L $\times$ H according to a regular grid. Each base station also has edge computing capabilities, i.e., it can receive computing tasks from cellular network's users and serve them at a rate equal to S instructions per seconds following a First Come First Served (FCFS) policy. Assume that all base stations are interconnected between each other via mesh topology.

1.1 Problem description

We consider N users placed at random locations `(x, y)` within the same 2D floorplan, where coordinates `x` and `y` are random variables to be defined later. Each user generates a new computing task request every T seconds, and each request consists of I instructions to be executed. T and I are exponentially distributed RVs. In particular, a user sends each new task request to its serving base station (i.e., the closest one), which in turn can follow one of methods below:

a) serve the request locally

b) forward the request to the less-loaded base station

1.2 Objectives

This project aims to fulfill the following:

- Evaluate the time required to complete a computing task for various values of N comparing method a) against method b)
- Evaluate the following scenarios:

  `x` and `y` are *uniform distributed* random variables in range

  $$x \in [0, width] \quad y \in [0, height]$$

x and y are distributed as a *lognormal distribution* with parameters defined in section 2.2

- `TODO...`

1.3 Performance indexes

- Response time of the system (when a packet leaves the system)
- Response time of each queue
- Packet loss
- `TODO...`

Chapter 2

Modeling

Our system is described by the following scheme

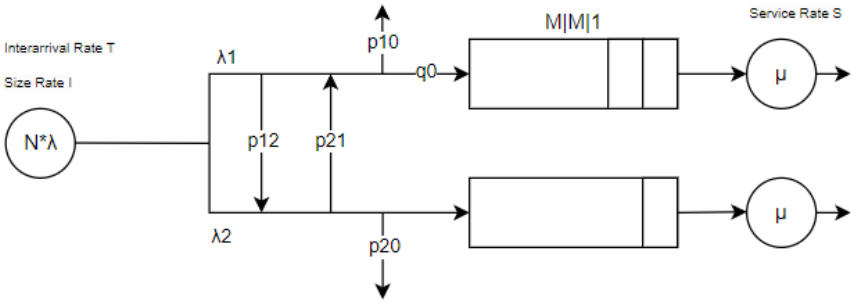


Figure 2.1: Scheme

As shown in Figure 2.1, each base station is modeled as an `M/M/1/k` system, as rates λ_i and μ_i are exponentially distributed random variables $\forall\ i \in [1, M]$.

2.1 Assumptions

We make some assumption in order to simplify the implementation of the system.

1. Rerouting propagation delay is constant for each base station.
2. Every job entering a queue is destined to be served and cannot exit the queue in any other way.
3. Each queue length is *finite* with `k` slots.

4. *(work in progress)* Width and height of the grid are such that it results in a square area.

2.2 Factors description

- `N`: number of users
- `k`: length of each queue
- λ_i: Interarrival rate for base station i
- μ: Service rate for each base station
- μ_{log}: Average of lognormal distribution
- σ_{log}: Standard deviation of lognormal distribution

Chapter 3

Implementation

3.1 Modules

The following modules have been defined:

- **EdgeNetwork**: compound module which represents the system and hosts the following simple modules
  - **BaseStation**: simple module which receives, precesses and (in case of scenario b) forwards packets sent by users according to the specified distribution.
  - **User**: simple module which generates and sends packets (with length dependent by the specified distribution) to the nearest base station.

3.2 Modules' behaviour

3.2.1 EdgeComputingNetwork

- Hosts base stations and users physically and memorizes their parameters in order to retreive them during the simulation via parent pointers.

3.2.2 BaseStation

- Receives packets from users in form of `cMessage` objects and depending on the current scenario performs one of the following actions:

Locally managed: if the queue has enough free slots the base station enqueues the packet to be processed, ignoring all other base stations.

Forwarding: when a packet is received the base station searches for the base statio with the lowest load on their queue and forwards the packet to it. If and only if every other base station has a higher load, the packet is served locally.

- It also records statistics every time a packet is *dropped* or *forwarded* (as number of packets) and about *queue length* and *response time* (in order to compute its average).

3.2.3 User

- Generates packets with length and rate which are exponentially distributed and sends them to the *nearest*[1] `BaseStation`.

[1] According to the euclidean distance

Chapter 4

Verification

Right away we describe the tests made to verify if our system tends to remain stable under extreme conditions or degenerate ones.

4.1 Scenario 1 - no users

As we expect in this degenerate case the system remains stable as there is nothing for it to precess or forward.

4.2 Scenario 2 - very high number of users

In this case we test our system against an enormous number of users and just a few base stations ($M = 2$ $N = 100$)

The system results to be robust to this kind of issue, maintaining a stable *queue length* and a consistent *response time*, as shown in Figure 4.1 and Figure 4.2

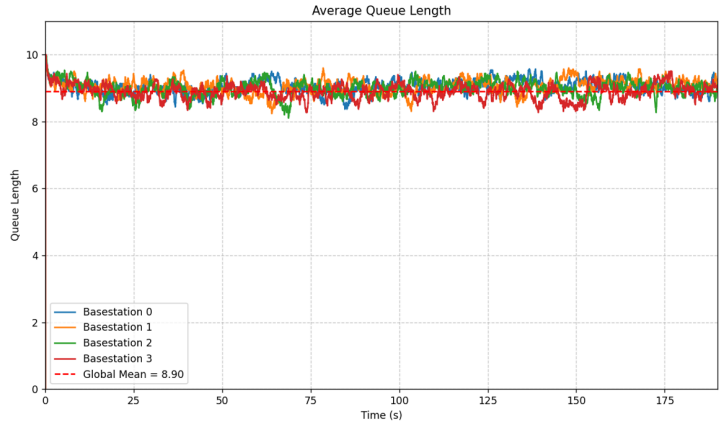


Figure 4.1

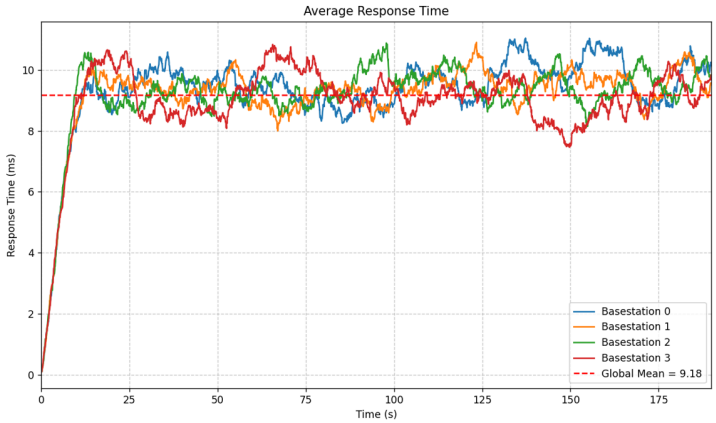


Figure 4.2

4.3 Scenario 3 - very small service rate

Now we want to know how the system reacts when the service rate is a lot lower than the interarrival rate.

Throughout simulations we find that the queue length tends to be stable after about $50s$ (Figure 4.3). Instead the response time has a *linear behaviour* over time (Figure 4.4).

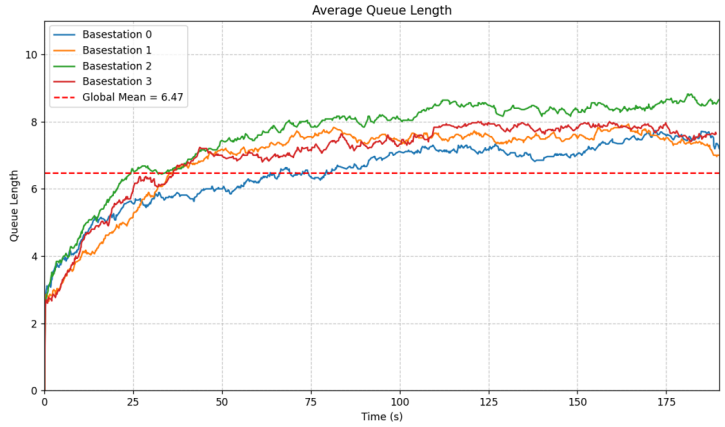


Figure 4.3

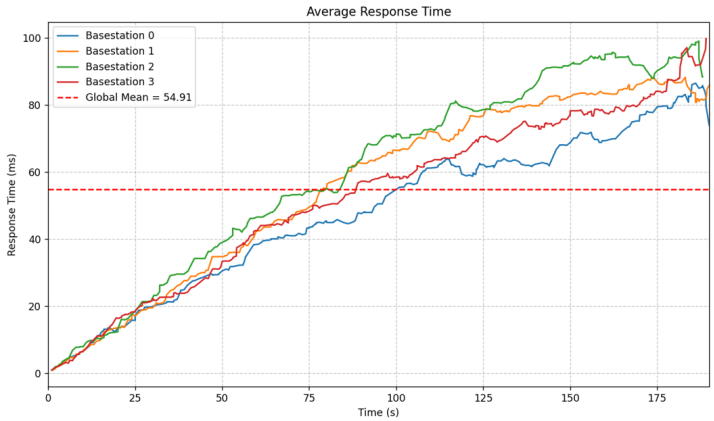


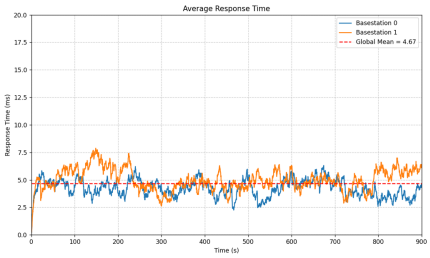
Figure 4.4

4.4 Consistency test

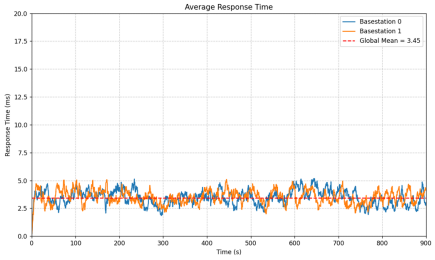
In this test we verify that the system produces coherent results in both configurations and those results to be feasible.

We compare the results of simulations in both '*Locally managed*' and '*Forwarding*' configuration.

It is clear that in both cases we end up with similar means ($\Delta = 1.22$), even though forwarding configuration is slightly better. What catches the eye is the difference between the variation of the two cases, once again favoring the forwarding.



(a) Locally managed



(b) Forwarding

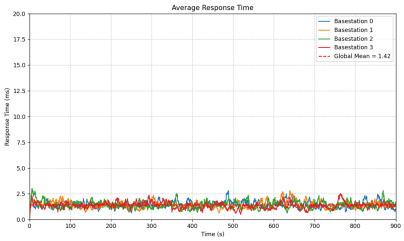
4.5 Continuity test

This test ensures that the system responds without discontinuity while gradually varying the parameters.

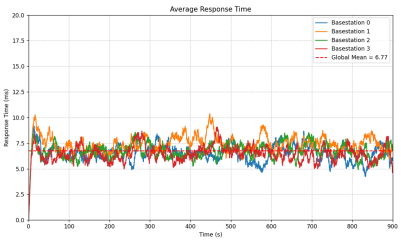
We inspect the response time while incrementing two parameters: the number of users and the interarrival rate.

4.5.1 Increment of users - $(10,\ 50,\ 100)$

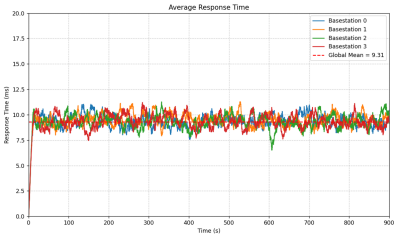
As we might expect the response time becomes higher with the rising of the number of users but the mean always remains under $10s$. Once the system is underway, it remains stable with the majority of the oscillations belonging to the case $N = 50$ (Figure 4.7b)



(a) $N = 10$



(b) $N = 50$

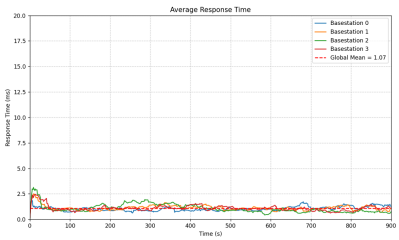


(c) $N = 100$

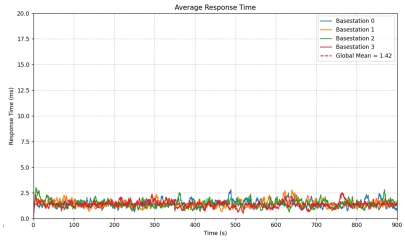
Figure 4.6

4.6 Increment of requests' load - $(\lambda = 0.1,\ 1,\ 5)$

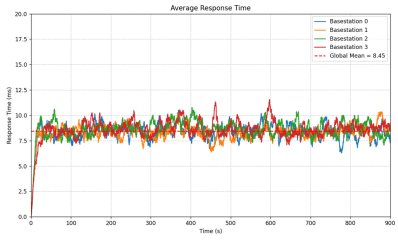
In this case the mean response time remains comparable for the first two values of lambda, the third one shows a mean value of almost an order of magnitude of difference and a higher variability.



(a) $\lambda = 0.1$



(b) $\lambda = 1$



(c) $\lambda = 5$

Figure 4.7

4.7 Monotonicity test

This test evaluate if the output of the system follows a monotonic relation with any of the input parameters.

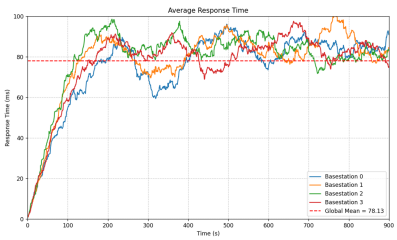
4.7.1 Number of users

This case has already been already analyzed in subsection 4.5.1.

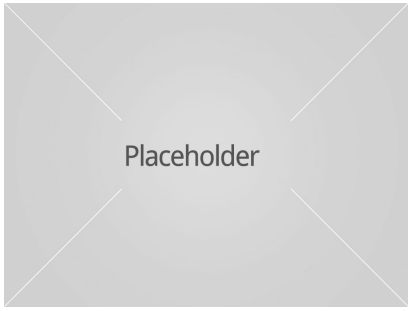
4.7.2 Service rate

With the variation of the service rate ($\mu = 1000,\ 5000,\ 10000$) the mean response time changes drastically over the cases.

In particular in the first one there aer tho orders of magnitude of difference with the others.



(a) $\mu = 1000$



(b) $\mu = 5000$

(c) $\mu = 10000$

Figure 4.8

4.8 Verification against the theoretical model

Now we compare the results of various simulations with those predicted by a theoretical model, which is described as follows.

4.8.1 Theoretical analysis

This is based mostly on *queueing theory*, in particular on the $M|M|1$ model which describes our base stations, remembering the fact that they are characterized by a queue of finite length k.

Nowing that the service rate is equal to the product of the *number of instructions processed per second* and the *interarrival rate*

$$\mu = S \cdot \lambda$$

we compute the *response time* for both configurations we simulated:

Locally managed:

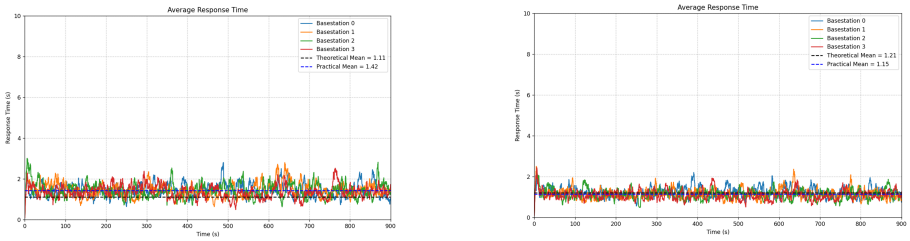
$$t_r = \frac{1}{\mu - \lambda}$$

Forwarding:

$$t_r = D + \frac{1}{\mu - \lambda}$$

Where D is a constant delay intrinsic in the system (i.e. network delay or computational delay)

In Figure 4.9a and Figure 4.9b we compare the results of theoretical and practical analysis in case of locally managed and forwarding.



(a) Case a.

(b) Case b.

Figure 4.9

Chapter 5

Calibration

We now discuss how factors and other parameters have been chosen throughout the developement.

5.1 Factors calibration

The aim of this section is to fix the intervals of the factors to correctly reproduce reality in our system.

5.1.1 Number of base stations

This number depends on where is the area located geographically and on it's dimentions. In suburban areas, are commonly spaced 2/3 km apart and in dense urban areas, they may be as close as 400/800 m apart[1].

5.1.2 Number of users per base station

This number changes in the range $[10, 50]$ depending on how much an area is effectively crowded.

[1] Source: Wikipedia

Bibliography