Simulation project of MEC Network

Rumi Alberto

University of Milan,
Simulation

1 Introduction

In this project I simulated the performance and possibility to develop a driving assistance Application with a Multi-Access-Edge Computing (MEC) network. The simulation is developed with the AnyLogic personal learning edition[1] framework, the pre-processing and post-processing steps are implemented with python.

2 Problem description

I decided to set the simulation in the city of Bergamo. Here, at the tollgate, cars arrive with a certain rate. Each car will then attach to the closest base station and will ask for a MEC Application instance. After the allocation of the MEC App, the car will start moving through a destination decided at random between 4 of the most important places in the city (hospital, upper town center, lower town center, station, stadium) and a random point in the city. During this movement the car will send information about his location and other parameters to the Base Station in order to let the MEC App process them. In this simulation the processing of the data is limited to the speed of the user that will change based on the area in which it is located; areas statically defined as GISRegions.

All the base stations will periodically communicate to all the users within their coverage area. The users, during their journey, will constantly check which base station's signal is the best one and, if it is needed, will trigger an handover procedure for the relocation of the MEC App.

The handover procedure can be triggered between base stations controlled by the same MEC Host or by different MEC Host. In the first case the handover is directly performed between the two Base Stations (Mx2 interface) otherwise a communication between the MEC Hosts is needed to migrate the MEC App.

3 Data pre-processing

To create the simulation environment I used the data provided from [2] in order to locate the different base stations in the territory filtering only for the region of interest ("BERGAMO").

According to the data, there are 9 different base stations located in the area, so I decided to use 3 different MEC Hosts in my simulation. Once the base stations locations are identified the MEC Hosts locations are defined. The MEC Hosts are located under three specific base stastions. These are selected by a simple greedy center selection algorithm I've implemented starting from a central base station and then selecting the other 2 maximizing the minimum distance between the selected points.

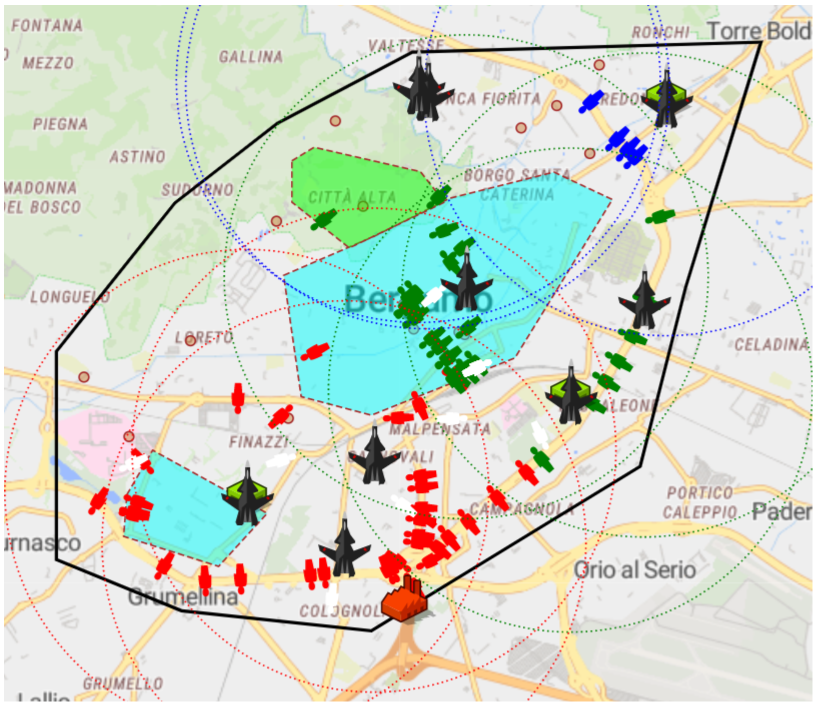


Figure 1: Screenshot of a simulation run

These pre-processing steps are implemented using python in the jupyter notebook file *PreProcessing_data.ipynb*.

I've then delineated the different GIS regions where the cars will have to keep a certain speed. In Figure 1 is possible to see the different areas:

- Blue: urban region, speed at 50 km/h
- Green: pedestrian region, speed at 30 km/h
- Other: suburban region speed at 90 km/h

4 Agent implementation

I've used different agents and agent types for the overall project, in this section is given a brief description of the implementation and the roles of the different components.

4.1 UserToBSMessage

This agent represents the message used to communicate from a user to a base station.

msgType	UserFrom
destination	pckDimension
BSTarget (for handover message)	

Figure 2: UserToBSMessage

It has the msgType parameter which allows to distinguish the different messages that a user want to send to a base stations:

- **0** → connect to base station,
- **1** → update message,
- **2** → MEC App allocation request to a specific destination,
- **3** → trigger handover procedure,
- **4** → disconnecting from base station,
- **11** → handover completed.

In a real situation when an update message is sent, it will take not jsut the location of the user, but also some sensor data in order to adjust all the parameters of the car. To model the possibility to have different update messages I used the `pckDimension` parameter which is modeled as an exponential random variable. The expected value of this random variable is set to 454, according to [3], which describes different possible packet size distributions in LTE. This model is used for any packet dimension field in every message.

4.2 BSToUserMessage

This agent represents the message from a base station to a user.

msgType	BSFrom
updateParameter	

Figure 3: BSToUserMessage

In this case the msgType field describes:

- **0** → broadcast signal message, in order to let the user control the power of the signal,
- **1** → connection accepted,
- **2** → set speed parameter,
- **3** → handover,
- **4** → arrived to destination.

4.3 MECHostToBSMessage and BSToMECHostMessage

These two messages are simple messages used in two situation: at the start of the simulation in order to let the base stations attach to the closest MECHost (msgType = 0) and to ask to the MECHost the allocation of a MEC App (msgType = 1).

msgType	pckDimension
BSFrom/MECHostFrom	

Figure 4: Message format MECHost-BaseStation

4.4 HandoverMessage

This message is used in the overall handover procedure and is exchanged between MEC Hosts and Base Stations.

It keeps track of both the Base Station and MEC Host pairs: relay (from where the handover procedure starts) and target (where the user wants to attach).

BSRelay	MECHostRelay
BSTarget	MECHostTarget
User car	pckDimension

Figure 5: Handover Message

4.5 Tollgate

This agent represent the node of spawn of the cars, for simplicity in this simulation is just a source component which dynamically adds user cars in the simulation. The inter-arrival times are modeled as a Poisson process (exponential random variable), when events occur independently at a constant average rate. I decided to simulate with 3 different average rates:

- 1 car/second ($\lambda = 1.0$),
- 20 car/minute ($\lambda = 0.33$),
- 6 car/minute ($\lambda = 0.167$).

4.6 UserCar

This agent represent a car which spawns at the tollgate. A UserCar agent will be idle until he receives a BACH (broadcast access channel) message from a base station in order to connect to it. After the connection phase, it will then ask for the allocation of a MEC App in order to start moving through his destination. While moving the car will periodically send informations regarding his location and state to the base station he is connected to. This is modeled with the event `sendInformationsToBS` happening at random rate described as an exponential random variable with $\lambda = 1.0$. This is because in a real situation it's not known how often a user sends data to the application, but I suppose that the average would be 1 update message per second.

During the movement through the destination the user will keep scanning for messages from the MEC App (BaseStation) in order to adjust the speed if it is requested.

The user will also start receiving different broadcast Messages from different base stations, when this happens the user checks if the signal power of the new base station is stronger then the one he is attached to. In order to make this measurement it is taken into account only the distance from the user to the base stations as the crow flies. If the new base station has a stronger signal then the older one, a procedure of handover is triggered sending an handover request message to the base station.

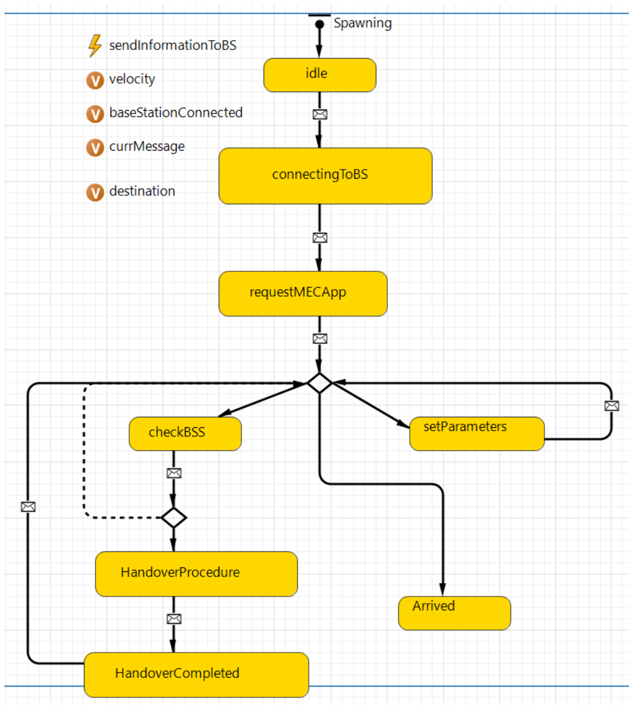


Figure 6: UserCar state diagram

4.7 BaseStation

The base station starts its lifetime connecting to the closest MEC Host. Once this step is completed it starts scanning for incoming messages and sending periodic messages to the users in his coverage area.

The broadcast messages are modeled as the `sendSignalBACH` event triggered every second as soon as the base station connects to the MEC Host. The base station will then wait for incoming messages.

When it receives a message, based on the msgType received, the base station forwards the message into different possible discrete events routines.

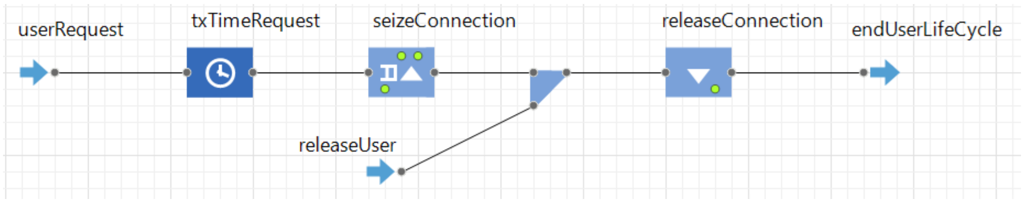


Figure 7: BaseStation: User connection request

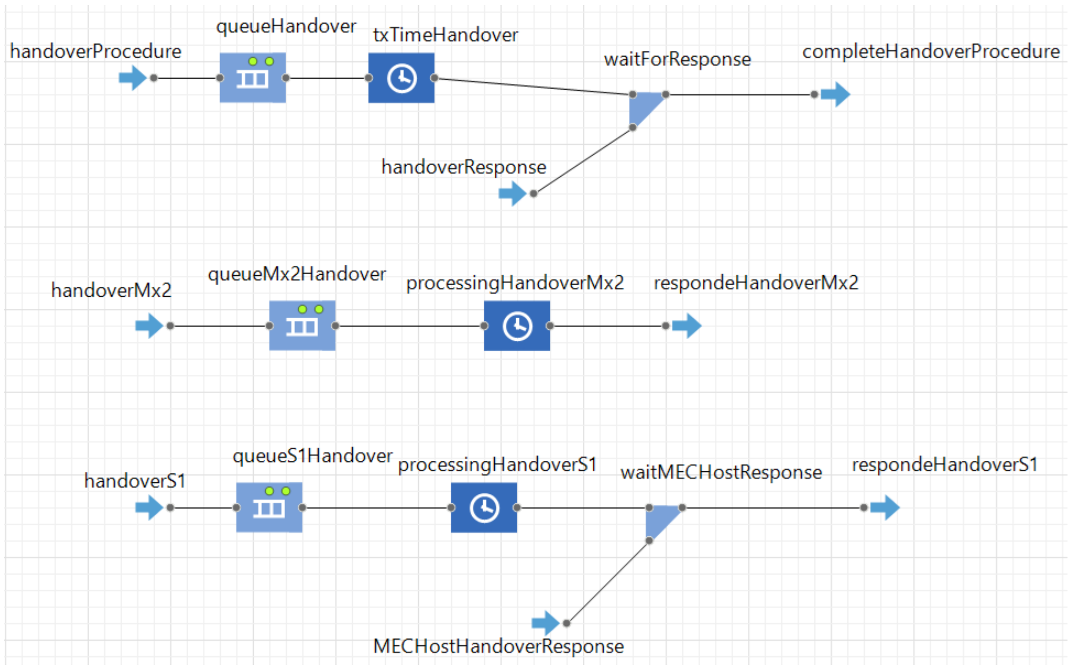


Figure 8: BaseStation: Handover request

In Figure 7 is described the procedure for the connection of a user to the base station. In order to do that there must be a connection available from the `connectionAvailable` resource pool. When the resource is seized, a connection accepted message is sent to the user. Once the user migrate to a different base station or arrives to the destination, the connection resource is released.

In Figure 9 there is the process for the MEC App allocation and also for the manage of the updates. The MEC app allocation request is forwarded to the MEC Host assigned and after it is provided a response the app is allocated and an OK message is forwarded to the user. After this process the User is then moved through the destination he requested. When a request for an update arrives, after the delay of the transmission and the processing, the drive parameters (in this simulation just the speed of the UserCar) are sent to the user car.

In Figure 8 is shown the manage of the handover procedure, it controls if the handover is under the same MEC Host or not:

- If it is under the same MEC Host, a request is sent to the Mx2handover procedure to the base station target, this base station will seize a connection resource with an internal userRequest message and send the handoverResponse message to the base station relay.

- otherwise a longer road is taken, the message is forwarded to the han-

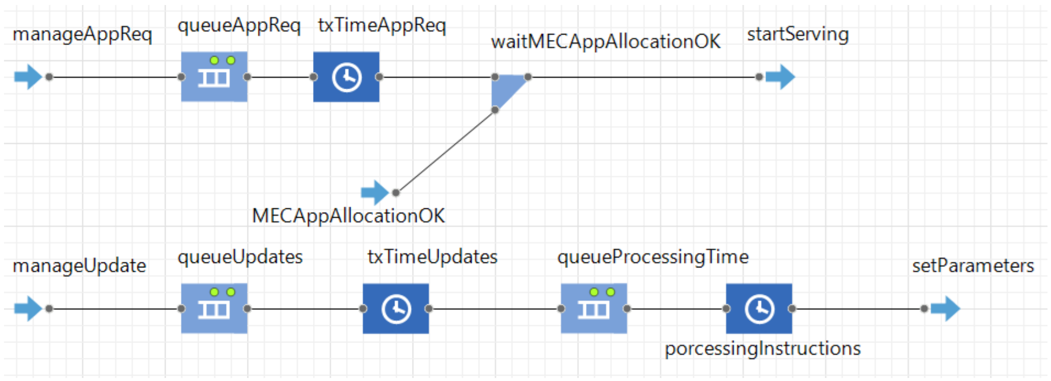


Figure 9: BaseStation: MEC App allocation and update management

doverS1 procedure which will forward the request for the allocation of the MEC App to the MEC Host Relay which will itself forward to the MEC Host target. The MEC Host target will then ask to the base station a connection from the connection pool and, when this is provided, will seize a MEC App resource. When all this procedure is done, a message to the MECHostHandoverResponse port to the base station relay is sent. The handover procedure can then terminate.

5 MEC Host

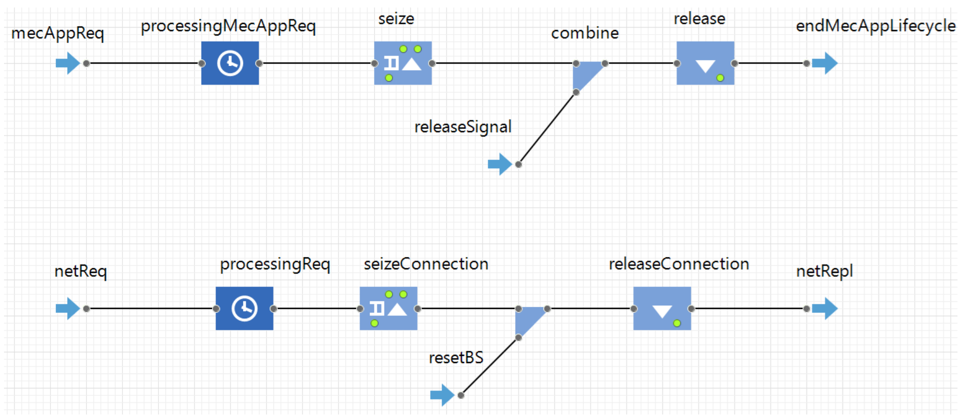


Figure 10: MECHost: network setup and MEC app allocation

The MEC Host will always scan for incoming messages. These messages can be network requests or MEC App requests, while the handover procedure is directly triggered by the base station relay. The MEC Host has two resource

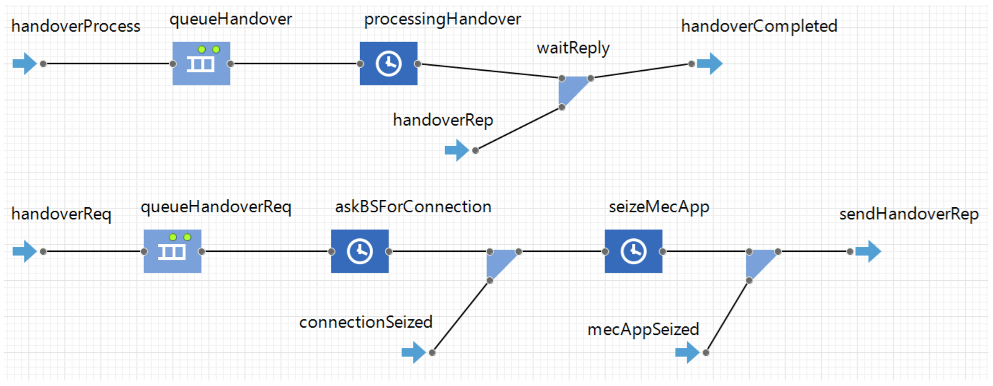


Figure 11: MECHost: handover process

pools:

- mecApps, which is the maximum number of MEC applications it can manage,
- networkConnection, which is the maximum dimension of the network it can manage.

The network request and the MEC App request procedures are similar procedures both with the schema represented in Figure 10. After the request of a service arrives, the relative resource is seized and a OK message is sent back to the source. Then when a user migrates over a different network or arrives to the destination the MEC App resource is released, while the connection is released only when the base station fails.

In Figure 11 is displayed what happens in the MEC Host when an handover procedure (between different MEC Hosts) is triggered. A handoverReq is sent to the MEC Host target, here the target MEC Host will ask for a connection to the base station target and when this is provided a MEC App resource is seized with an internal message to the mecAppReq procedure. When this two resources are seized, a handoverRep message is sent to the MEC Host Relay which will release the MEC App resource locally and will tell to the base station to release the connection resource itself.

6 Transmission and processing delay tuning

The reason why is convenient to develop edge computing application is due to the real-time capabilities of the response time of the applications. This is because the application is as close as possible to the user and it is migrated in order to keep the minimum distance possible.

In order to model the processing time and the delay caused by the distance user-MEC App, I used the delay component. In the first delay component of any communication user-base station the processing time is delayed by a the **transmission delay** factor. In order to model this factor there are several assumption and references I used.

Due to the fact that the simulation is in a relative close range field and the interfaces between MEC Hosts and base stations are probably in optic fiber I decided not to deal with the transmission delay between these components, while the transmission between the user and the base station needs to be processed in a more accurate way.

Each base station has the same bandwidth assumed to be equal to 10 MHz [4]. In order to calculate the maximum capacity of the channel used for the communication the shannon capacity formula is applied:

$$C = Blog_2(1 + SNR)$$

Where **B** is the bandwidth and **SNR** is the signal-to-noise-ratio. In order to calculate the signal power with reference to the distance user-base station I used the path-loss schema, precisely the log distance path loss model[5] which allows to calculate the received signal power as:

$$P_{Rx_{dBm}} = P_{Tx_{dBm}} - PL$$

As transmission power I used 18dBm, as suggested from [6]. While in order to calculate the PL the relative formula is used:

$$PL = PL_0 + 10\gamma log_{10}(\frac{d}{d_0} + X_g)$$

Where PL_0 is the path loss at reference distance $d_0 = 10m$ in the free space model[7], using as frequency one of the frequency used by the base stations in LTE = 1710MHz [4] :

$$PL_0 = 20log_{10}(d_0)+20log_{10}(f)-27.55 = 20log_{10}(10)+20log_{10}(1710)-27.55 = 57.11dBm$$

So the received power can be calculated, using a γ for the air interface equal to 2.2 [5], assuming no attenuation caused by flat fading ($X_g = 0$) :

$$P_{Rx_{dBm}} = P_{Tx_{dBm}} - PL = 18dBm - (57.11 + 10 \cdot 2.2log_{10}(\frac{d}{10})$$

With the assumption of just the thermal noise as noise in the connection, which is a strong but ok assumption due to the fact that the transmission power of the user is already tuned in the work[6], is possible to calculate the SNR (with the conversion in Watts):

$$SNR = \frac{P_{Rx_{toWatt}}}{ThermalNoise_{toWatt}}$$

Using the thermal noise as the Johnson–Nyquist noise $Th = -101dBm$, now the channel capacity can be calculated in function of the distance user-base station:

$$C = Blog_2(1+SNR) = 20Mhz{\cdot}log_2(1+\frac{(18dBm - (57.11 + 10 \cdot 2.2log_{10}(\frac{d}{10})))_{toWatt}}{(-110)_{toWatt}})$$

Once this is computed the time for the transmission can be easily calculated as the packet dimension parameter over the capacity of the channel. In the main of the project there is also a parameter called avgPLR which is the average packet-loss-ratio of the radio interface. This is set to a constant value of 0.3[8]. I've used this parameter in order to model the packet loss of each transmission as a geometric random variable X $\sim G(1 - avgPLR)$ which represents the number of retransmissions needed before a correct transmission. So finally the transmission delay is computed:

$$txDelay = X \cdot t$$

with **t** as the time calculated above.

Another processing time that needs to be modeled is the time needed to allocate a MEC App from the user. This value is extrapolated from [9], setting the corresponding delay value as a random variable $\sim N(1.53, 0.03)$.

For the processing time in the overall project I decided to assign a fixed number of CPU Cycle to each base station and MEC Host to 1GHz. This computation is distributed over the number of the connections or MEC Apps actually running. Then I modeled that a CPU cycle is needed for each bit in the packet dimension field of the currently processed packet. So the processing time is modeled as deterministic time depending on the number of connections activated and the dimension of the packet processed.

7 Results

In order to understand how the different delays affect the process, while a packet is transmitted and processed the movement is blocked until it exits the chain. So in each discrete event process chain the user involved is blocked until the process is completed and it receives a response. So the transmission delay and the processing delay can be evaluated with the overall time a user spend in order to arrive to the destination.

Lambda	Station	LowerTown	Hospital	Stadium	UpperTown
0.167	1168	1467	1990	2687	3821
0.33	1177	1474	1991	2681	3824
1.0	1188	1484	1993	2683	3838

Table 1: Average time for each destination

It is possible to see from Figures 12, 13,14 the different plot of the time spent for reaching a specific destination with the variation of the traffic parameter,

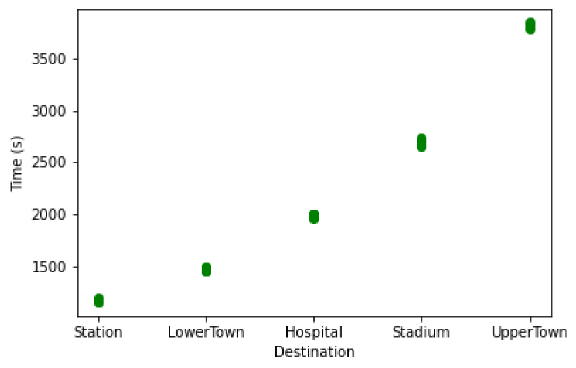


Figure 12: Scatterplot $\lambda = 0.167$

and in the mean value is extrapolated in Table 1. Is possible to see an increase of the average time with an increase of the traffic as expected and also the increase in the variance. It is also notable how for the Hospital destination the average time do not change that much as in the other cases, even if the journey is longer than for the Station destination. This is due to the fact that this journey is the only one which doesn't require a migration of the MEC App between different MEC Hosts, which is the process that introduces the most delay out of all the processes.

In Figure 15 is reported the busy factor of the different base stations and MEC Hosts in the case of $\lambda = 0.33$. This describes the amount of connections and MEC Apps activated for the different components. It is possible to notice that there isn't a uniform distribution of the connection load between the base stations, mostly because of the path the users take that are more related to a specific MEC Host. The location of the base stations that I got from [1] may also not be optimal, having two base stations that are really near each other as it is possible to see in Figure 1.

During every simulation run, the user_cars will also change color during their journey based on which MEC Host is serving them.

The graphs and the table are extrapolated in the file *Post-Processing.ipynb*, while the Time stack chart in Figure 15 is computed with Anylogic statistics.

8 Conclusions

In this project I worked on a possible Multi access edge computing environment in Bergamo with different assumptions on signal transmission factors for the implementation of a driving assistance MEC application.

Different aspect of this situation are confronted and analyzed with different traffic parameters: the delays the user might encounter and the load balance of the base stations and MEC hosts.

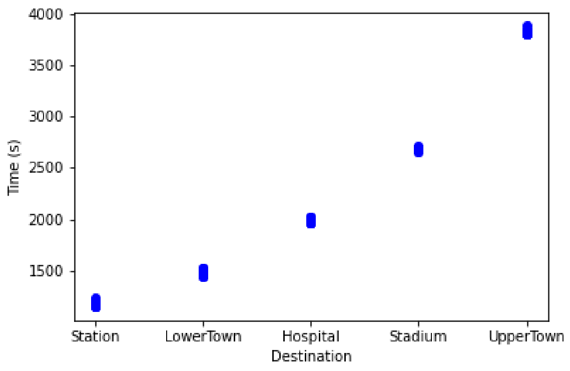


Figure 13: Scatterplot $\lambda = 0.33$

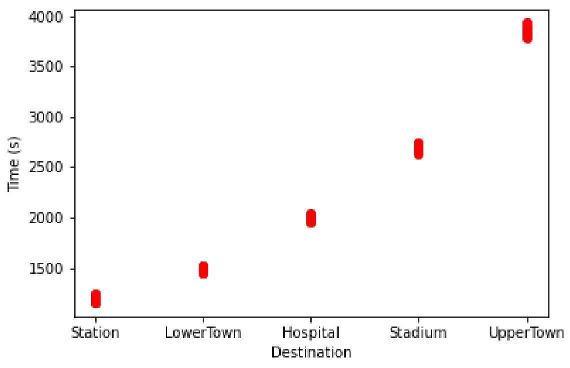
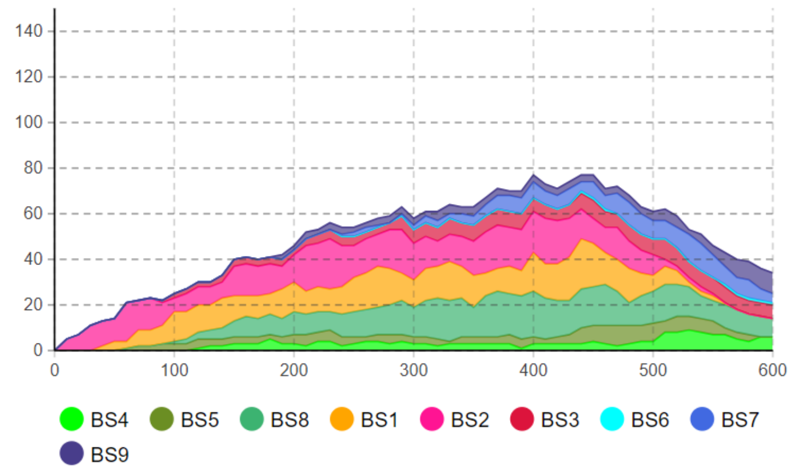


Figure 14: Scatterplot $\lambda = 1.0$



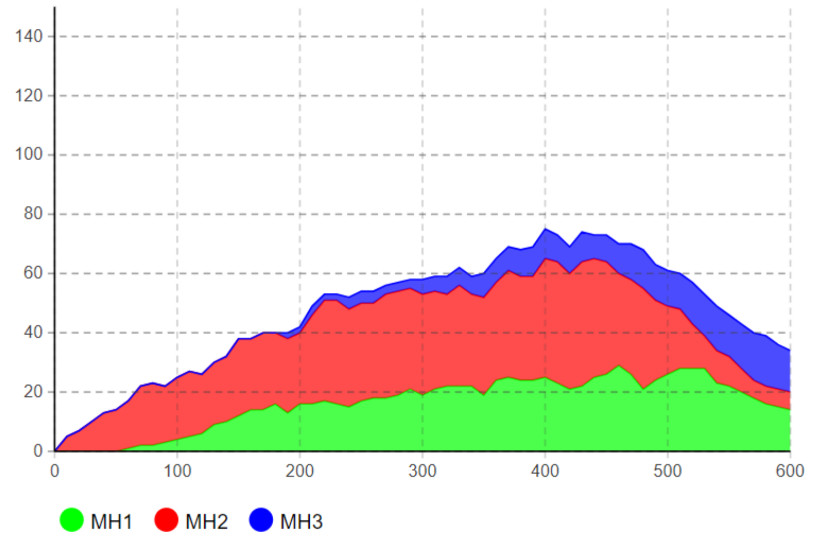


Figure 15: Busy amount of Base stations and MEC Hosts with $\lambda = 0.33$

References

[1] "Anylogic simulation framework." www.anylogic.com.

[2] "Opendata base stations in italy." http://www.datiopen.it/it/opendata/Mappa$_{d}elle_{a}ntenne_{i}n_{I}talia.$

[3] G. Dán, V. Fodor, and G. Karlsson, "Packet size distribution: An aside?," *Quality of Service in Multiservice IP Networks*, 2005.

[4] G. T. . version 14.3.0 Release 14, *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception.*

[5] "Log-distance path loss model." https://en.wikipedia.org/wiki/Log-distance$_{p}ath_{l}oss_{m}odel.$

[6] A. Haider and S.-H. Hwang, "Maximum transmit power for ue in an lte small cell uplink," *Electronics*, 2019.

[7] "Free space path loss." https://en.wikipedia.org/wiki/Free-space$_{p}ath_{l}oss.$

[8] K. R. K. Dakuri Chiranjeevi, Mahender, "Channel aware scheduling algorithms for 3gpp lte downlink," *International Journal of Emerging Trends in Engineering Research (IJETER), Vol. 3 No.1, Pages : 10 – 18*, 2015.

[9] K.-T. Seo, H.-S. Hwang, I.-Y. Moon, O.-Y. Kwon, and B.-J. Kim, "Performance comparison analysis of linux container and virtual machine for building cloud," 2014.