Management and Content Delivery for Smart Networks: Algorithms and Modeling

Laboratory activities report

GROUP 11
Andrea Avignone s279954
Tommaso Carluccio s276909
Vincenzo Madaghiele s277028

MASTER's degree in ICT for Smart Societies 2020-2021

Contents

1 Description of the system

The simulation scenario involves the portion of an Internet of Things (IoT) network, consisting of a group of sensors for real-time monitoring, a Micro Data Center (MDC) and a Cloud Data Center (CDC). The Micro Data Center is in charge of computing locally the requests related to the collected data, exploiting specific fog nodes capabilities. According to the system configuration, if a dedicated buffer is present, the incoming data can be buffered when all the fog nodes are busy. Otherwise, requests are forwarded to the Cloud Data Center, which is also able to provide proper resources for the most demanding data elaborations. A representation of the system is shown in Figure 1.

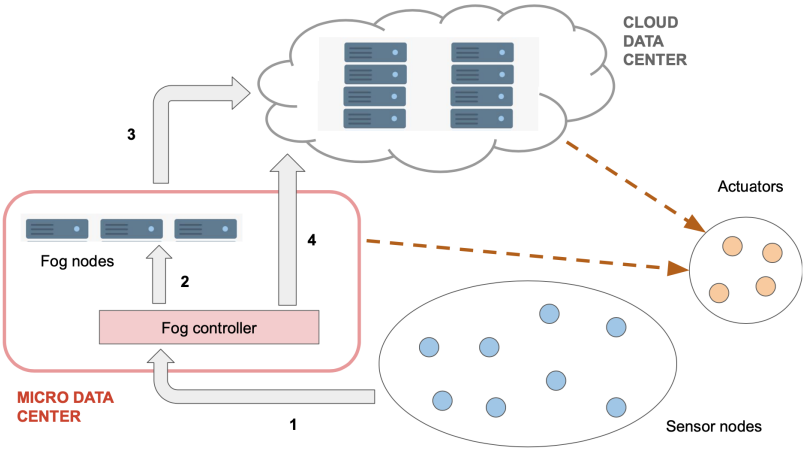


Figure 1: Representation of IoT system

Laboratory 1 is focused on the analysis of the Micro Data Center operations, starting with a simple model and progressively evaluating the effect of variations of system parameters such as buffer size, number of servers and server assignment policy. In Laboratory 2 the simulations are aimed at the evaluation of the CDC system, first by itself and then concerning operations of the system as a whole with multiple metrics.

In the following sections, different configurations and parameters settings are presented to simulate the behaviour of the system, analyzing the obtained results in comparison to the outcomes expected from theoretical modelling. In particular, the main objective concerns the evaluation of proper indicators to highlights how the different changes impact the system performances. All simulations are carried out within Python environment and are available at `https://github.com/vincenzomadaghiele/MCDSM_labs`.

2 Laboratory 1 - Internet of Things system simulation

This Laboratory is focused on the Micro Data Center (MDC) operations. The MDC is composed of one or more processing units called Fog Nodes and a buffer with variable size common to all fog nodes. If the MDC is idle, the incoming packets are immediately processed, otherwise they are stored in the buffer for later processing. Instead, if the customers number exceeds the MDC buffer size, packets are directly forwarded to the Cloud Data Center, which is not considered in this Laboratory.

2.1 Micro Data center with single fog node

The baseline simulation scenario consists of a single Fog Node serving the incoming data. Two distinct simulations are performed, taking in consideration the case with no buffer and finite buffer size.

2.1.1 Single fog node and no buffer

In this simulation the MDC is composed of a single Fog Node and no buffer, so the incoming data is processed only if the single processing unit is idle. This scenario can be modelled as a M/M/1/1 queue (sometimes referred to as M/M/1/0), which can be read as simplified version of the popular Erlang-B model with $C = 1$. The model is represented in Figure 2, highlighting both the flow of customers entering the system and the the flow of the ones that are forwarded to the Cloud Data Center.

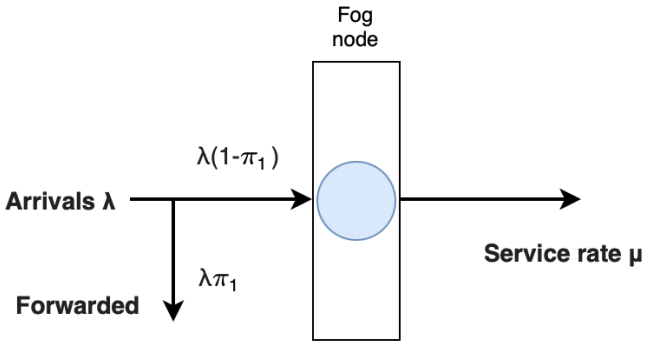


Figure 2: Erlang-B model for M/M/1/1 queue

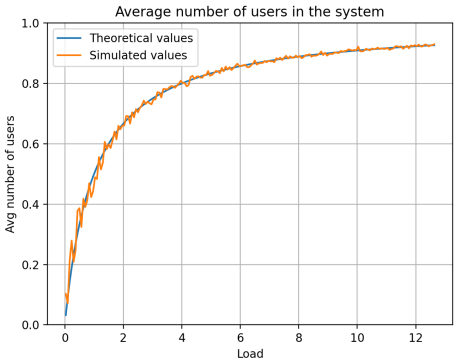
Since in this case no buffer is present, if customers arrive when the single Fog Node is busy, they are directly forwarded to the CDC. Keeping fixed the value for average clients arrival time to $\lambda^{-1} = 30\ ms$, different average service time values μ^{-1} are then employed, varying therefore the load of the system defined as $\rho = \lambda/\mu$. Given a simulation time $t = 300\ s$ (5 minutes), Figure 3 shows the experimental results obtained for Average Number of Users and Forwarding Probability (fraction of customers forwarded to the CDC) as a variation of the system load.

A system of this kind is particularly inefficient since every time the single Fog Node is busy data is forwarded to the CDC. Consequently, the two curves in Figure 3 are identical, growing logarithmically as the service time increases. For high load values, the whole data tends to be sent directly to the Cloud Data Center without any pre-processing, because the single Fog Node is always busy.

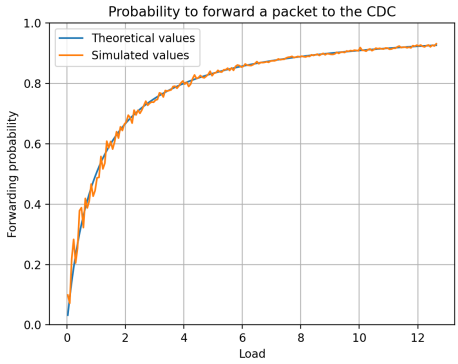
The consistency of these simulation results is confirmed by the theoretical values, also shown in Figure 3. The theoretical forwarding probability is computed as the loss probability in Equation 1.

$$L = \frac{\lambda \cdot \pi_1}{\lambda} = \frac{\frac{\rho^c}{C!}}{\sum_{k=0}^{C} \frac{\rho^k}{k!}} = \frac{\rho}{1+\rho} \tag{1}$$

Given the absence of a waiting line, the time spent in the system is exclusively related to the service time $E[T] = \mu^{-1}$ and therefore proportional to the system load.



(a) Average number of users



(b) Forwarding probability

Figure 3: Results for system with single service and no buffer

2.1.2 Single fog node and finite buffer

The previous configuration showed poor performance and therefore the Micro Data Center was not able to sufficiently relieve the effort expected from the Cloud Data Center. However, a first improvement can be obtained including in the model a waiting line to buffer the arriving customers when the fog node is not idle. In this case, the system is modelled with a M/M/1/B queue where $B-1$ is the buffer size. The value of this parameter impacts the general performance of the system, as shown in Figure 4. However, improvements are considerable only for relatively small loads. The analysis takes in consideration the behaviour for $B = 4$, $B = 6$, $B = 11$ which correspond to a buffer size of $Buf = 3$, $Buf = 5$ and $Buf = 10$ respectively, as case study. Observing Figure 4b, for larger values of ρ, the three curves tend to overlap, nullifying the effect of the buffer size. The marginal improvement using $Buf = 10$ is present only for $\rho <= 1.5$. In general, depending on the specific context and objectives of the system, a larger buffer is not always a wise solution. In fact, the improvements in terms of loss probability are valid only within limited circumstances, while the delay experienced by the customers due to the waiting phase is not a negligible aspect. In Figure 4a, the difference in average time is evident according to the buffer size choice. Doubling the length of the waiting line implies a considerable increase of the time spent in the system, which can be inconvenient. Instead, adding more servers for pre-processing represents a more reliable solution (Section 2.2). By observing the comparison among theoretical values and experimental results, simulations can be assumed coherent with the corresponding queue models. In particular, the theoretical average time in the system has been evaluated through Equations 3 and 4 using Little's Law and considering the fraction of customers lost in time unit due to the finite buffer capacity. The theoretical loss probability is described by Equation 2 and in the simulated case it represents the fraction of packets that are immediately forwarded to the CDC.

$$L = \frac{\lambda \cdot \pi_B}{\lambda} = \pi_B = \frac{1-\rho}{1-\rho^{B+1}} \cdot \rho^B \tag{2}$$

$$E[T] = \frac{E[N]}{E[\Lambda]} \tag{3}$$

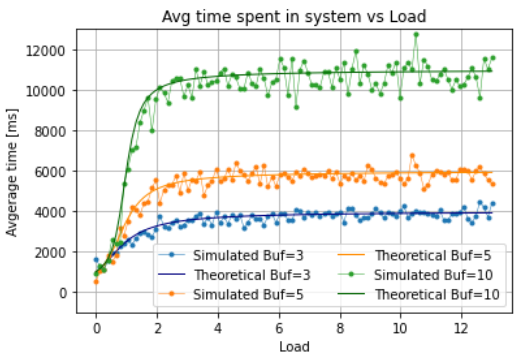
$$E[\Lambda] = \lambda - \lambda \cdot \pi_B \tag{4}$$

2.2 Micro Data center with multiple fog nodes

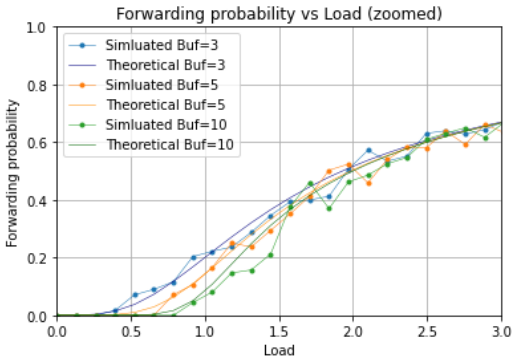
In this section, the system is extended to include two fog nodes in order to analyze the performance variations. Different configurations have been tested, considering distinct choices for the buffer size: no waiting line, infinite waiting line and medium to short waiting line.

2.2.1 Two fog nodes and finite buffer

Considering the case in which an extra fog node is added to the MDC, an analysis of the system performance has been carried out comparing the results obtained for different configurations, including the case of a single Fog Node.



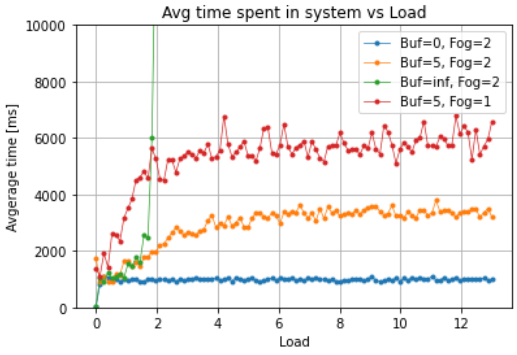
(a) Average time in the system



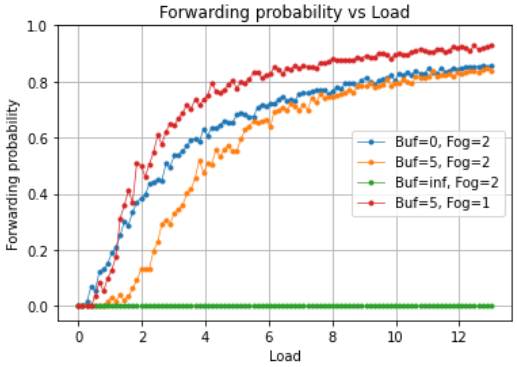
(b) Forwarding probability

Figure 4: Results for system with single service and waiting line

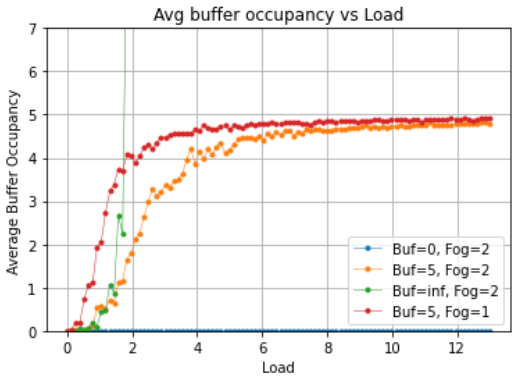
Figure 5a confirms the results obtained in the previous section, showing an average time in the system which increases with the buffer size; this result is particularly evident considering an infinite buffer. On the other hand, by keeping fixed the buffer size and increasing the number of fog nodes, the system improves performances since more packets are served at the same time, reducing in this way the average time spent in the system. Figure 5b shows how the forwarding probability changes varying the system load in different scenarios. As shown for the single node case, increasing the buffer is crucial to reduce losses (with an infinite buffer the probability drops to zero). Looking instead at the two curves with the same buffer size but with one and two fog nodes respectively, the latter shows a smaller forwarding probability because the queue moves faster by relying on two servers and the buffer fills up slower as also shown in Figure 5c. Therefore, having an extra fog node (even with a smaller buffer) is a good choice to reduce losses and queuing time. In fact for an increasing load, the forwarding probability curve for the zero buffer and two nodes case is always below the one for the buffer equal to five with just one node.



(a) Average time in the system



(b) Forwarding probability



(c) Average buffer occupancy

Figure 5: Results for system with two fog nodes and different waiting lines

2.2.2 Different scenarios with multiple fog nodes

Considering a system with multiple fog nodes, it is important to define a policy to assign a client to a specific node choosing between the available ones. An additional simulation has been carried out considering four possible algorithms. The first one simply performs a "sorted search" for an idle service, starting from the first one and moving to the next if it is busy. The second algorithm chooses at random between the idle nodes, while the third performs a round robin search and the last one takes as input a vector of costs assigned to each node choosing each time the least costly node available. The performances of each method are compared in Table 1 in terms of both percentage of time that each node is busy and total operational cost. The cost is obtained multiplying the total busy time of each node times an assigned unitary cost, also reported in Table 1. All the presented results are the average over ten simulations with the same average service time for each node ($1000ms$) and a load of 2.

	Node 1	Node 2	Node 3	Node 4	Node 5	**Overall**
Node Unitary Costs	1	0.8	0.6	0.4	0.2	**Cost** ($\cdot 10^3$)
Sorted assignment	74.57%	54.58%	35.30%	22.93%	13.06%	453.80
Random assignment	41.26%	41.57%	39.11%	40.57%	40.60%	367.30
Round Robin	39.72%	41.12%	38.33%	38.36%	38.68%	356.30
Least Cost	11.13%	22.22%	37.69%	52.98%	75.69%	**263.77**

Table 1: Relative busy time per node with different assigning algorithms

The distribution of the load between the nodes is balanced in the random assignment and the round robin cases. Instead, the sorted assignment method tends to prefer the first node, which in this case is the most expensive one in terms of unitary cost, resulting in the highest overall cost. The cheapest method is the least cost since it is optimized for the analyzed case, choosing always the available node with the smallest unitary cost.

A second scenario has been simulated considering different service time for each node. Moreover, it was assumed that faster nodes are characterized by proportional operational cost, as shown in Table 2a. The performances are compared in terms of total operational cost and average queuing delay, reporting the results in Table 2b. Imposing a different service time for each node does not influence the choice of the tested algorithms. In fact, even the Least Cost approach selects the node based on the vector of unitary costs without considering the service rate. For these reasons the relative busy times for each node are not affected with respect to the previous scenario.

	Node 1	Node 2	Node 3	Node 4	Node 5
Node Unitary Costs	1	0.8	0.6	0.4	0.2
Average Service Time [ms]	500	750	1000	1250	1500

(a) Assigned costs and corresponding service time

	Overall Operational Cost ($\cdot 10^3$)	**Average Queuing Delay [ms]**
Sorted assignment	322.20	**711.82**
Random assignment	305.00	1018.14
Round Robin	306.63	1028.20
Least Cost	**282.14**	1172.46

(b) Results

Table 2: Different assigning algorithms with different service rate for each fog node

In this case, the overall operational cost for the Sorted method is the smaller one since the first node (which is the preferred one by the algorithm) is also the faster one. Concerning the Least Cost method, a larger operational cost is present since it continues to choose the node with the smallest unitary cost, which is also the one with larger average service time. Therefore, the Least Cost approach shows evident queuing delay, as opposed to the significantly smaller delay of the Sorted assignment. The Random and Round Robin methods better balance the load on the nodes, representing a trade-off among the other two mentioned algorithms, in terms of both overall cost and average queuing delay.

2.2.3 Different distribution of the service time

A new simulation has been performed modeling the MDC as an M/G/1 queue hence considering still a Markovian arrival process and varying the distribution of the service time. In particular, in addition to the exponential distribution, both constant and uniform distribution are considered for the service time. The obtained results in terms of average number of users and average time in the system are reported in Table 3 and compared with the expected theoretical results obtained with the Pollaczeck-Khintchin (Equation 5 and Equation 6). Different values of the variance have been considered for the uniform distribution and the coefficient of variation of service time C_s^2 has been computed accordingly using equation 7. The simulation has been carried out under ergodicity conditions, considering a load $\rho = 0.8$ (and an average service time $E[S] = 1000ms$) to guarantee the correctness of the P-K formula.

$$E[N] = \rho + \rho^2 \cdot \frac{1 + C_s^2}{2(1 - \rho)} \tag{5}$$

$$E[T] = E[S] + \rho \cdot E[S] \cdot \frac{1 + C_s^2}{2(1 - \rho)} \tag{6}$$

$$C_s^2 = \frac{E[S^2] - E^2[S]}{E^2[S]} \tag{7}$$

	Experimental values		Theoretical values	
Distribution of service time	**Av. time in the system [ms]**	**Av. number of users**	**Av. time in the system [ms]**	**Av. number of users**
Constant	**2949.90**	**2.39**	3000.00	2.40
Exponential	4513.59	3.70	5000.00	4.00
Uniform - var=200	3011.43	**2.39**	3000.40	2.40
Uniform - var=500	3108.85	2.54	3001.00	2.40
Uniform - var=1000	3205.54	2.60	3002.00	2.40

Table 3: Simulation of the system with different distribution of the service time

The obtained results are averaged over 50 simulations and they are coherent with the expected theoretical results. It is interesting to observe that increasing the variance of the uniform distribution has a negative effect on the system performance, in particular for what concerns the average time in the system. In fact, a larger variance increases the coefficient C_s^2 and consequently $E[T]$ and $E[N]$. For this reason, the best performances are obtained with a service time as constant as possible hence with a constant distribution of the service time ($C_s^2 = 0$), while the worst results correspond to an exponential distribution of service time ($C_s^2 = 1$).

3 Laboratory 2 - Expanding the baseline scenario

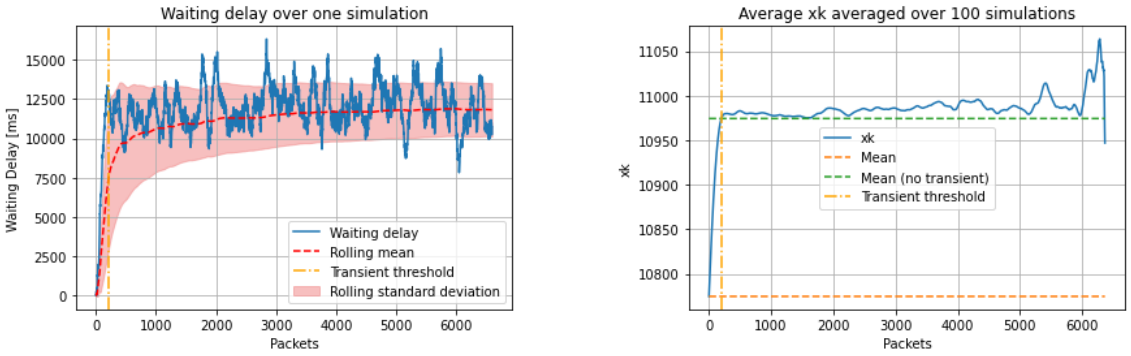
The second laboratory focuses on the analysis of the IoT system as a whole, taking also into account the Cloud Server operations. The packets can be of two different types:

- **Type A packets** require simple processing tasks and can be completely processed locally by the MDC.
- **Type B packets** are more complex; they require to be pre-processed by the MDC and then they are further processed in the CDC.

If the MDC buffer is full, the packets are forwarded to the CDC and, if needed, they are stored in the CDC buffer, waiting to be processed. The ratio of Type A and B packets arriving to the system is defined by the quantity f, which is a parameter of the system. If the MDC forwards Type B packets to the CDC with no pre-processing, their service time is longer because the CDC is expected to perform pre-processing too. In order to have a more realistic scenario, a constant propagation delay has been introduced for the packets moving from the MDC to the CDC. The system can be seen as a tandem of two queues with finite buffers. However a product form for this configuration is not possible since losses may happen at the CDC level.

3.1 Transient identification and confidence intervals

For this task we evaluate the average waiting delay of the Cloud Data Center sub-system, considering a single server for both the MDC and the CDC. Figure 6a shows the waiting delay of all the packets in the Cloud sorted by arrival time in the simulation. At the beginning of the simulation there is an evident transient period before reaching the steady state. This is best exemplified by the behavior of the rolling mean, which becomes constant as the simulation time increases.



(a) Waiting delay over one simulation

(b) $\overline{x}_k$ coefficient over 100 simulations

Figure 6: Transient identification and removal

To precisely identify the transient period we use the $\overline{x}_k$ technique. This technique is based on the computation of the mean value $\overline{x}_k$ of the observations $X = \{x_1, x_2, ..., x_N\}$ by removing the first k elements, recursively increasing k.

$$\overline{x} = \frac{1}{n} \sum_{j=1}^{n} x_j \tag{8}$$

$$\overline{x}_k = \frac{1}{n-k} \sum_{j=k+1}^{n} x_j \tag{9}$$

The values of $\overline{x}_k$ are averaged over 50 different simulations to remove randomness from the metric, yielding results as in Figure 6b. From the curve in Figure 6b it is possible to identify a clear knee corresponding to a threshold value around 200 packets. To obtain a more reliable estimate of the average waiting delay it is sufficient to compute this metric removing the delays of packets before the 200th, which correspond to the identified transient period. As can be seen by the mean value variation in Figure 6b before and after removing the transient period packets, the average value computed after transient removal is much more precise. Some observations can be

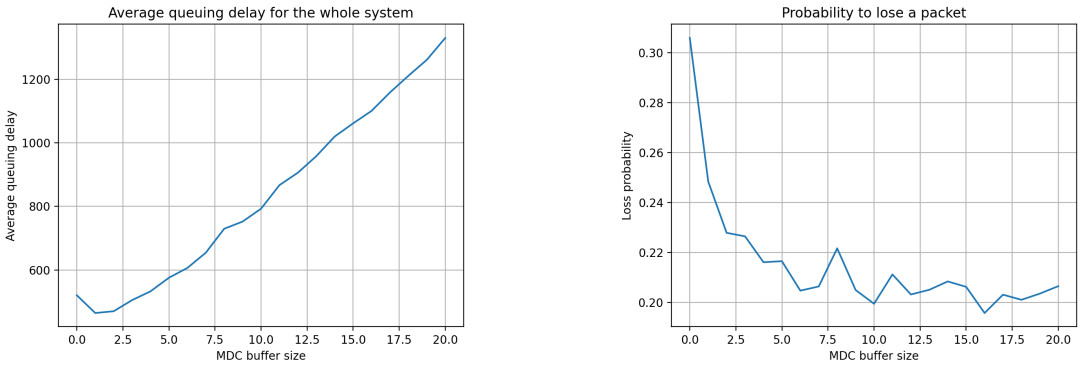
made regarding the confidence interval of this metric. As can be seen from Figure 6a, the rolling standard deviation decreases as the simulation time (and consequently the number of packets) increases. The obtained confidence interval of the average waiting delay is 10977.65 ± 242.32 ms over 50 simulations with a 99.8% confidence after removing transient values. To obtain a smaller confidence interval one could increase the number of simulations or the simulation time of each of them.

3.2 Overall system operations

For the next analysis we have considered the system as a whole, combining the operation of the MDC with the one of the CDC. The queuing delay is therefore the total time a packet spends in the whole system combining the two partial contributions of the two subsystems. An exception are Type A packets that are processed in the MDC for which the queuing delay is just the time spent in the first subsystem since they are not forwarded to the cloud. A loss in this scenario may happen only when a packet arrives to the CDC and it is dropped due to a full buffer; if the same situation happens at the MDC, the packet is immediately forwarded to the cloud. For this analysis the output values have been obtained by averaging the results of 50 simulations in order to reduce the randomness effect.

3.2.1 Impact of MDC Buffer size on the overall system

Varying the MDC buffer size allows to process more packets in the first sub-system reducing the fraction that is forwarded to the cloud without any pre-processing. As already seen in section 2.1.2, increasing the buffer results in a general increase of the average time in the system (in this case an increase in the queuing delay of the MDC). This contribution becomes more and more relevant for the general queuing delay as the buffer size increases as shown in Figure 7a. On the other hand a small buffer does not have a direct impact on the loss probability since, as stated before, a loss may happen only at the CDC level; however, as it can be seen in Figure 7b, the loss probability of the overall system tends to decrease almost exponentially as the buffer increases. This happens because with a large MDC buffer less packets are directly forwarded to the cloud decreasing a lot the load on the CDC buffer which is the only one responsible for loss in the system.



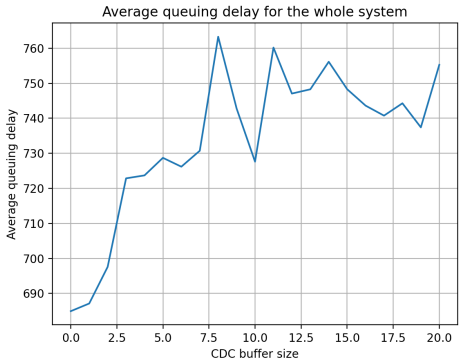
(a) Average queuing delay

(b) Average loss probability

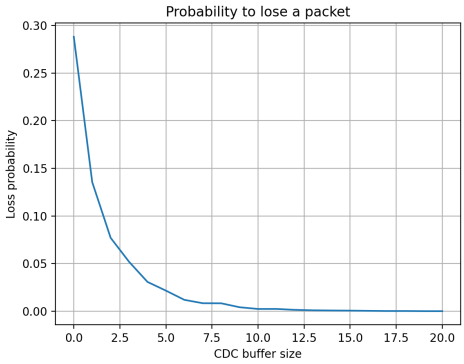
Figure 7: Simulations with progressively increasing MDC buffer size

3.2.2 Impact of CDC Buffer size on the overall system

Changing the buffer dimension in the cloud has a similar effect as changing the one in the MDC. As shown in Figure 8a, a large buffer increases the waiting delay of the packets even if the increase is smaller in this case, because the cloud server is faster than the MDC one. The loss probability on the other hand is directly influenced by the CDC buffer size as it can be seen by the exponentially decreasing curve in Figure 8b.



(a) Average queuing delay

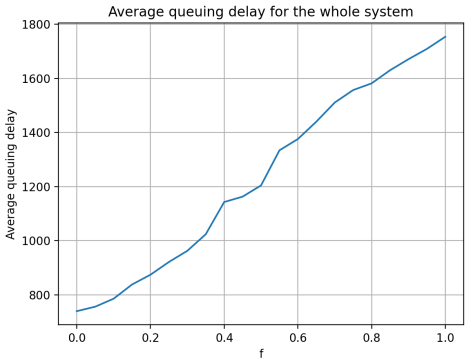


(b) Average loss probability

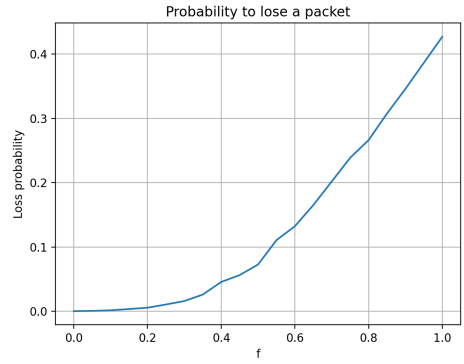
Figure 8: Simulations with progressively increasing CDC buffer size

3.2.3 How much the Ratio f affects the packet drop probability?

Varying the ratio of type B packets can heavily influence the performance of the system. Type B packets require extra processing once they have passed through the MDC and also require an higher overall service time to be processed in the CDC. The queuing delay is plotted in Figure 9a and increases linearly with the fraction of packets B influenced by the greater service time required to process those type of packets. Furthermore, type A packets that are processed in the MDC do not arrive in the cloud at all. This results in an overloaded cloud buffer as the percentage of type B packets increases and consequently in an exponentially increasing loss probability as shown in Figure 9b.



(a) Average queuing delay

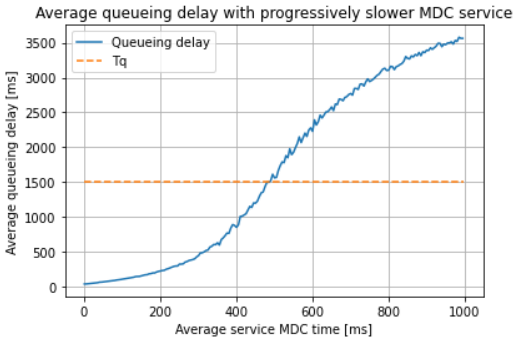


(b) Average loss probability

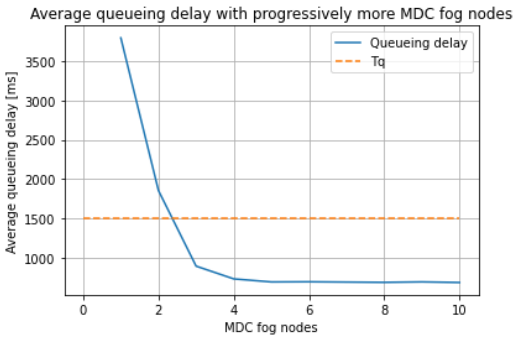
Figure 9: Simulations with progressively increasing fraction of type B packets f

3.3 Average queuing time of the overall system

We now consider the queuing time of the overall system, that is the time spent by a packet from its arrival in the MDC system to its departure either from the MDC or the CDC. We set a maximum threshold $T_q = 1500\ ms$ for the overall queuing delay. The first case we consider is that of a single fog node in the MDC. We simulate the performances of this fog node by progressively increasing its average service time, keeping a fixed inter-arrival rate. As shown in Figure 10a, in order for the queuing delay to be below the threshold we need a fog node with at most an average service time around $500ms$.



(a) Average queuing delay with progressively slower MDC service



(b) Average overall queuing delay with progressively more MDC fog nodes

Figure 10: Average queuing delay over 30 simulations

Another interesting factor to consider is the number of fog nodes. We evaluated the system performances with the same threshold, progressively increasing the number of fog nodes, each with an average service time of 1000 ms. For the overall queuing delay to be below T_q we need to have at least 3 fog nodes. We could argue that it is more convenient to have a single faster fog node than multiple slower ones. In the second case in fact, the server capacity may be not optimally exploited, especially when the load is not high; in general better performances can be achieved by aggregating all the available capacity in just one server. In the simulated case, in fact, a single fog node with an average service time of $500ms$ is sufficient to obtain satisfactory delay performances; to obtain the same results with an average service time of $1000ms$ we would need three of them.

3.4 Overall system with multiple servers and operational costs

In this section we simulate the overall system over a defined period of time of 24 hours. IoT systems and networks in general experience substantial variation based on the time of the day, it is therefore necessary to simulate these different conditions to correctly evaluate the system operations. To obtain a realistic simulation we found it useful to contextualize the system in a possible real-case application of an IoT network similar to the one of our model. We imagined our system to be a wildlife monitoring network as represented in Figure 11.

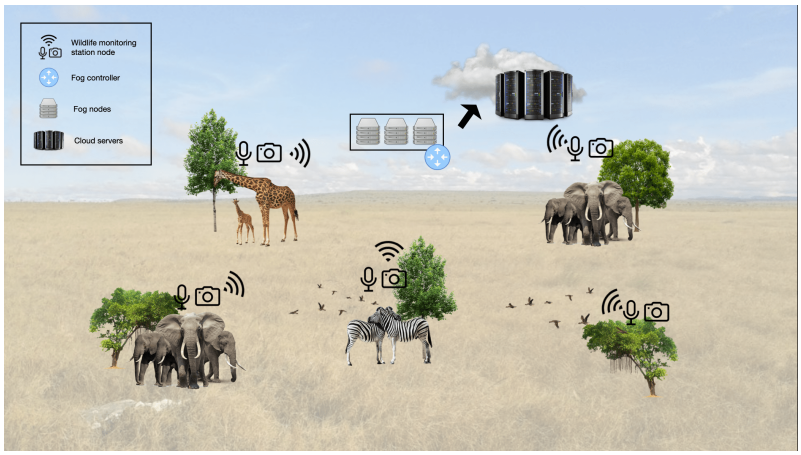
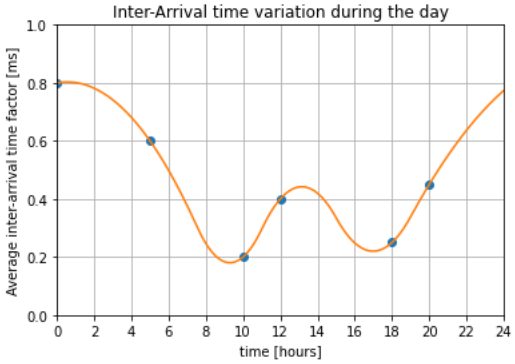


Figure 11: Wildlife monitoring simulation

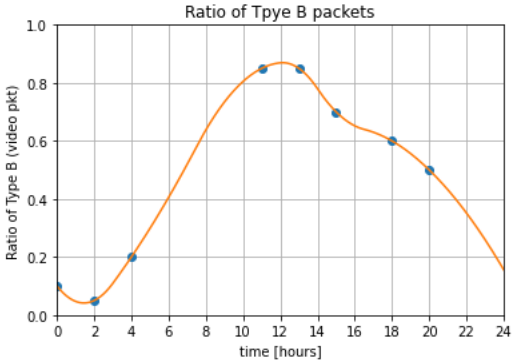
In this scenario each wildlife monitoring node has a camera continuously recording video and a microphone recording audio. Data is sent to the fog controller each time the sensor nodes detect the presence of animals. The fog nodes then extract the necessary information from the packets. The inter-arrival time is therefore not constant, as different animals have different biorhythms. It is realistic to assume that animals will be more active during the day, so the inter-arrival time is modelled to be smaller in this time period, with minimum points at 10:00 and 18:00, corresponding to more animals activities. Type A packets are audio recordings, while type B are video recordings,

which require more processing. The percentage of video packets decreases during the night, as cameras are less capable of detecting animals with less light and animals are in general less active.

We have defined functions for the inter-arrival time and the ratio of type B packets that mimic this kind of animal behavior during the day by interpolation of the blue points in the Figure 12. In particular the function in Figure 12a provides a factor (between 0 and 1) to weight the constant value for the average inter-arrival time according to the supposed traffic, while the one in Figure 12b is the f factor variation during the day.



(a) Inter-Arrival time variation factor during the day

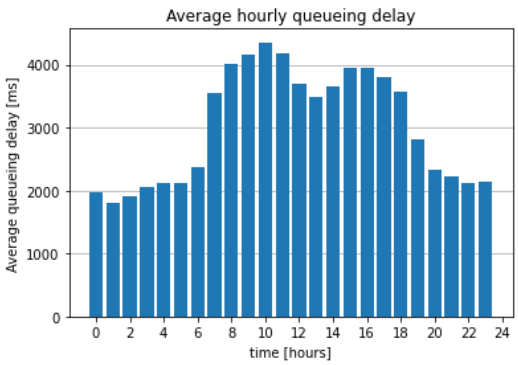


(b) Change in ratio of Tpye B packets during the day

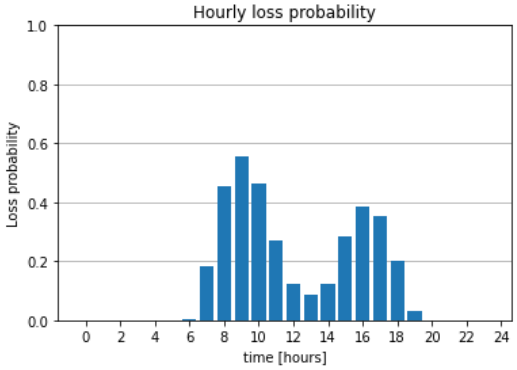
Figure 12: Changing factors in 24 hours simulation

3.4.1 Overall system performance over different arrival rate and packet type ratios

A 24h simulation has been performed using the previously described functions to vary the inter-arrival rate and the fraction of type B packets during the day. The system performance metrics have been evaluated looking at the average values for every hour and are plotted in Figure 13.



(a) Average hourly queuing delay



(b) Hourly loss probability

Figure 13: Overall hourly system performance metrics over 24 hours

As expected the average queuing delay is heavily influenced by the varying inter-arrival time as it can be seen from the corresponding (opposite) peaks of Figure 13a and Figure 12a. Furthermore the steep increase of the queuing delay from the night to the morning hours is the result of the simultaneous contribution of a smaller inter-arrival time and a rapidly increasing percentage of type B packets which require a longer service time to be processed. The decrease in delay from the afternoon to the night is instead smoother following the behaviour of the curves in Figure 12.

The hourly loss probability is plotted in Figure 13b and it shows how losses happen only between six in the morning and eight in the evening as it is reasonable to think since both the incoming traffic and the video packets fraction are higher during those hours. In particular a peak in the loss probability is present around 10 in the morning where the average inter-arrival time is minimal and the percentage of video packets is maximal. The subsequent decrease is then due to an increasing average inter-arrival time.

3.4.2 Combining different server models

This simulation scenario is characterized by four cloud servers of three different types. Each server type has its own average service time and a unitary cost as shown in Table 4a. The unitary costs have been defined to be inversely proportional to the service speed and the total cost is computed by multiplying the unitary cost of the corresponding server by the actual service time. Different combinations of cloud servers have been tested to find the best possible configuration using all three types. The results are reported in Table 4b where the first three rows show the basic scenarios with four equal servers for reference. All the following simulations exploit the Sorted Node Assignment algorithm (see Section 2.2.2) to assign the cloud servers.

Server Type	**Unitary Cost**	**Av. Service Time** [ms]
A	1	200
B	0.4	500
C	0.1	1000

(a) Types of Cloud servers

Servers configuration	**Av. Queuing Delay [ms]**	**Av. Loss Probability [%]**	**Overall Operational Cost ($\cdot 10^6$)**
A-A-A-A	1170.13	2.64	133.45
B-B-B-B	2104.92	24.63	88.33
C-C-C-C	2963.54	43.03	26.14
A-A-B-C	**1962.66**	**22.89**	82.53
A-B-B-C	2173.73	27.41	73.86
A-B-C-C	2420.44	33.22	**55.15**
B-C-A-A	2037.72	23.19	79.72
C-B-A-A	2060.41	23.39	78.04

(b) Simulated server configurations

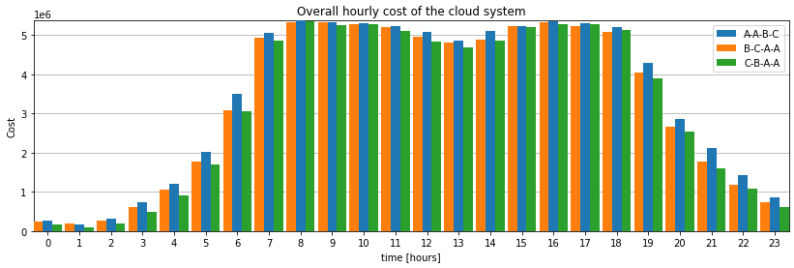
Table 4: System with different Cloud Server Types

From Table 4b it is evident that using two fast servers (A-A-B-C) is the best option in terms of average queuing delay and loss probability while the minimal overall cost is obtained using two slow servers (A-B-C-C). Comparing the results in the middle three rows of the Table the configuration with two A servers is likely the most convenient option and different permutations have been tested to check the node assigning algorithm influence on the results, as reported in the last two rows. A promising solution might be the B-C-A-A configuration, because A nodes are fast and help in case of high load (between 10:00 and 18:00), but in case of low load (during the night) there is no need for fast processing and least costly nodes are preferred. This allows to reduce the overall cost while slightly increasing the queuing delay and keeping an acceptable percentage of loss probability. Figure 14 shows the variation of cost and queuing delay throughout the day.

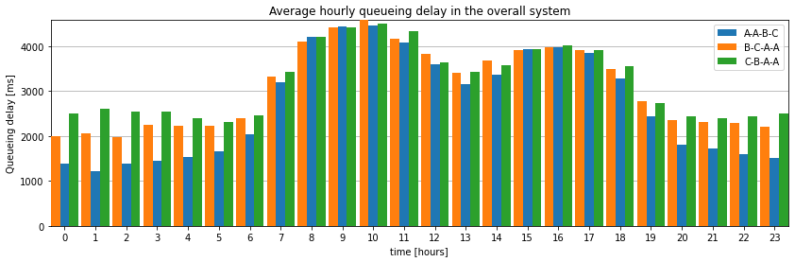
While in Figure 14b there is an evident increase in queuing delay in non-peak hours, at the same times in Figure 14a there is few proportional decrease in costs among the three configuration. However Table 4 shows a significant increase in overall costs when using A-A-B-C. These costs might seem low for a 24h simulation but when accumulated over multiple days of operations they may become significant. Instead, we might prefer to have a small increase in queuing delay in non-peak hours each day to decrease the costs by choosing the B-C-A-A configuration, which we will pick as the best compromise.

3.4.3 Lowering the number of Cloud Servers

In this section we experiment on how to obtain the same delay performances as in Section 3.4.2 with only two servers. We then show how the system reacts with respect to different packet Types in the best obtained configuration with two servers. The following results have been obtained with the same load and MDC configuration as in Section 3.4.2. A reasonable threshold for average queuing delay seems to be $T_q = 2100\ ms$, as it is satisfied by the three best combinations in Section 3.4.2. Table 5 shows the results of simulations with different combinations of two servers.



(a) Hourly cost over 24 hours under three different server configurations



(b) Hourly delay over 24 hours under three different server configurations

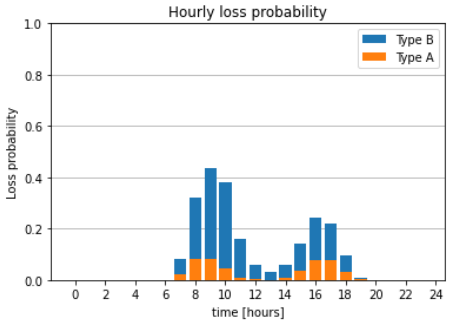
Figure 14: Hourly metrics over 24 hours under the three best performing server configurations

Servers configuration	**Av. Queuing Delay [ms]**	**Av. Loss Probability [%]**	**Overall Operational Cost ($\cdot 10^6$)**
B-C-A-A	2037.72	23.19	**79.72**
A-A	**1650.50**	**18.14**	101.65
B-A	2279.02	34.37	68.85
A-B	2260.14	34.42	68.82

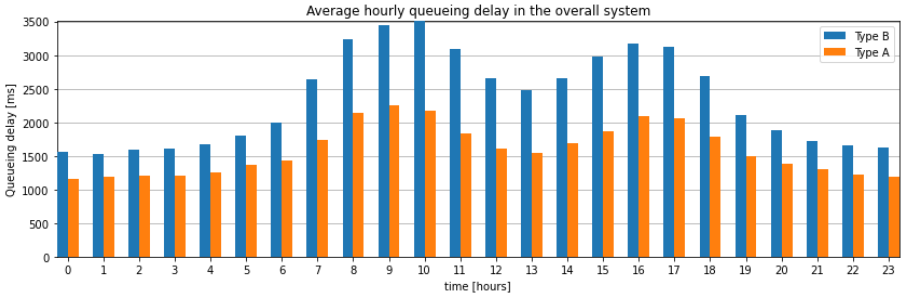
Table 5: System with different Cloud Server Configurations

In this case the only configuration that is able to guarantee an average delay below T_q is A-A, in this configuration the two servers will be busy most of the time, so having two type A servers means a significant increase of cost with respect to the B-C-A-A configuration. Substituting a type A server with a type B server does lower the operational cost but delay increases and a lot more packets are lost. This is because two servers are not enough to handle the load at peak times so they saturate and have to throw away many packets. Consequently, although more costly, we chose the A-A configuration as the best one with two servers and we computed the following results in this configuration. Figure 15 shows some hourly metrics by packet type in A-A configuration; no significant difference has been noticed with B-C-A-A on this regard, apart from a proportional decrease in loss probabilities and delays.

The Figure shows that the variation of f factor during the day influences also the loss probability and the delays experienced by these two different types of packets. The loss probability in Figure 15a is significantly different among the two types of packets; as the percentage of Type B packets increases, it is less likely to lose Type A packets. Overall, it is much more likely to lose a Type B packet both because the f factor is higher at peak hours and because they require longer processing, saturating the buffer. The queuing delay varies in a similar fashion. At non-peak hours the queuing times for the two packets types is very similar, because the MDC does not saturate and no packet is lost, so type A packets are entirely processed by the MDC. The CDC buffer is in this case empty most of the time because of low load and it is free to fast process only Type B packets, resulting in lower queuing delays.



(a) Hourly loss probability over 24 hours by packet type



(b) Hourly queuing delay over 24 hours by packet type

Figure 15: Hourly metrics over 24 hours by packet type