Performance Evaluation of Computer Systems and Networks Project Report

Lorenzo Catoni Leonardo Giovannoni Marco Lucio Mangiacapre

Contents

1 Introduction **2**

2 Implementation **2**

3 Verification **3**
3.1 Code verification . . . 3
3.2 Degeneracy test . . . 3
3.2.1 Exponential case with zero take-off time . . . 3
3.2.2 Exponential case with zero parking and landing time . . . 4

4 Deterministic case **5**
4.1 Analysis . . . 5
4.2 Stability condition . . . 5
4.3 New airplanes never have to queue before landing . . . 6
4.4 Case $t_l+t_o = t_n$ . . . 6
4.5 General case . . . 7
4.6 Summary of deterministic analysis . . . 8

5 Exponential case **8**
5.1 Preliminary analysis . . . 8
5.2 Considerations on the parking time . . . 9
5.3 Waiting times at different utilization levels . . . 9
5.4 Waiting times EPDF . . . 10
5.5 Parking occupancy study . . . 10

6 Final remarks **11**

1 Introduction

The presented system can be modeled as a queue system, in Figure 1 there is the scheme that summarizes its properties. We used two different colors for different branches, which are intended to indicate the path they will follow once they exit the server, in particular the red color indicates a higher priority than orange. The variables in the system are of temporal measure, therefore they will be used interchangeably according to this convention:

- $\lambda = \frac{1}{t_n}$
- $\mu_1 = \frac{1}{t_l}$
- $\mu_d = \frac{1}{t_p}$
- $\mu_2 = \frac{1}{t_o}$

The formulas of utilization will be justified later and are:

- $\rho_1 = \frac{\lambda}{\mu_1} = \frac{t_l}{t_n}$
- $\rho_2 = \frac{\lambda}{\mu_2} = \frac{t_o}{t_n}$
- $\rho = \rho_1 + \rho_2 = \lambda(\frac{1}{\mu_1} + \frac{1}{\mu_2}) = \frac{t_o + t_l}{t_n}$

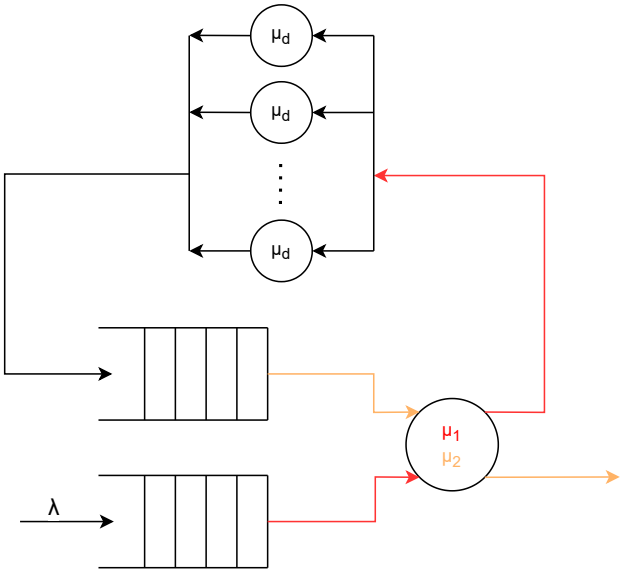


Figure 1: Queueing system

2 Implementation

We decided to implement the system in OMNeT++ using only two modules:

- **Spawner**: creates airplanes following a customizable distribution and disposes them when they finish their life cicle
- **Airport**: handles all the queueing logic and schedules airplanes on the runway following the correct priority constraints

The two modules communicate via a two-way channel, the Spawner sends airplanes to the Airport and the Airport sends them back when they finish the cycle. Since OMNeT++ gives the possibility to define custom messages we defined a **Airplane** message that holds internally the various parameters: t_l, t_p, t_o. In this way an Airplane is responsible for his own landing time, parking time and takeoff time.

Implementing the system with only two modules allowed us to write the code in a very concise and clean way; to give an counter example: if we wanted to implement the system in more modules, splitting the queues

and the runway in different modules, we would have had a lot of overhead messages just to synchronize the various actors every time an event occurred. We are aware that in this way we traded modularlity for simplicity of code, but we think in this case if was a worthwhile trade off.

The code for this project can be found on github here `https://github.com/PuffoTrillionarioGonePublic/PECSN-Project`.

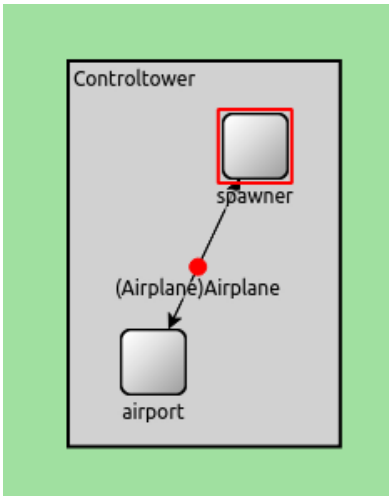


Figure 2: View of our two modules exchanging a message of type Airplane

3 Verification

3.1 Code verification

We checked that the code was working as expected with the debug GUI that comes with OMNeT++, the use of OMNeT++ specific containers (such as `cQueue`) and the use of the `WATCH` macro enabled us to execute the simulation step by step and check that all the parts of the systems are behaving as we expected.

3.2 Degeneracy test

The system was modeled as a queueing system. To verify the accuracy of the code, a series of tests of particular or borderline cases were carried out in order to verify the model.

3.2.1 Exponential case with zero take-off time

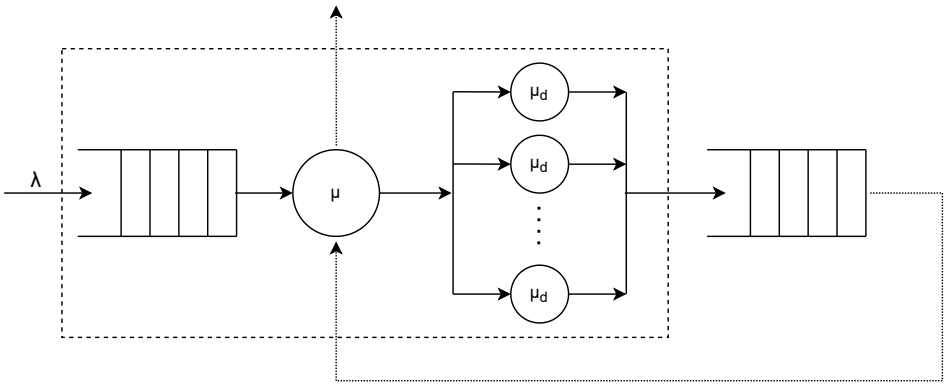


Figure 3: Simplified queueing system

With this simplification, the take-off time of the airplane has been set equal to 0, which in the figure 1 means μ_1 set equal to $+\infty$. This means that the server does not spend time serving the jobs coming from the low priority queue, and therefore the service of the high priority queue and the consequent associated statistical

parameters are not affected by what happens in the second queue. In figure 3 the flow of jobs passing from the low priority queue (on the right) has been shown as dashed as it does not affect the remaining highlighted system. Therefore, analyzing the dashed subsystem we recognize a queue system with an M/M/1 to which an M/M/∞ is connected in cascade, therefore by applying Burke's theorem it is possible to state that the delay center has a rate input equal to λ. So the mean number of jobs in the landing queue is $\frac{\rho^2}{1-\rho}$ and knowing that the probability of finding n processes within a delay center is a mean Poisson variable $\frac{\lambda}{\mu}$, $\frac{\lambda}{\mu_d}$ will be the average number of processes present in the parking lot. Other mean performance indexes could be calculated, but those could be easily derived from above expressions. The parameters chosen are those below, where t_l in particular varies from an exponential with mean 3 to mean 27. That is why we tested in this manner: $\rho = \frac{\lambda}{\mu_1}$ in the interval [0.1, 0.9] with a step of 0.1

- $T_l \sim exp(3),\ exp(6), .., \ exp(27)$
- $T_o = 0$
- $T_p \sim exp(60)$
- $T_n \sim exp(30)$

We chose a warmup period of 1 day and a simulation time limit of 60 days.

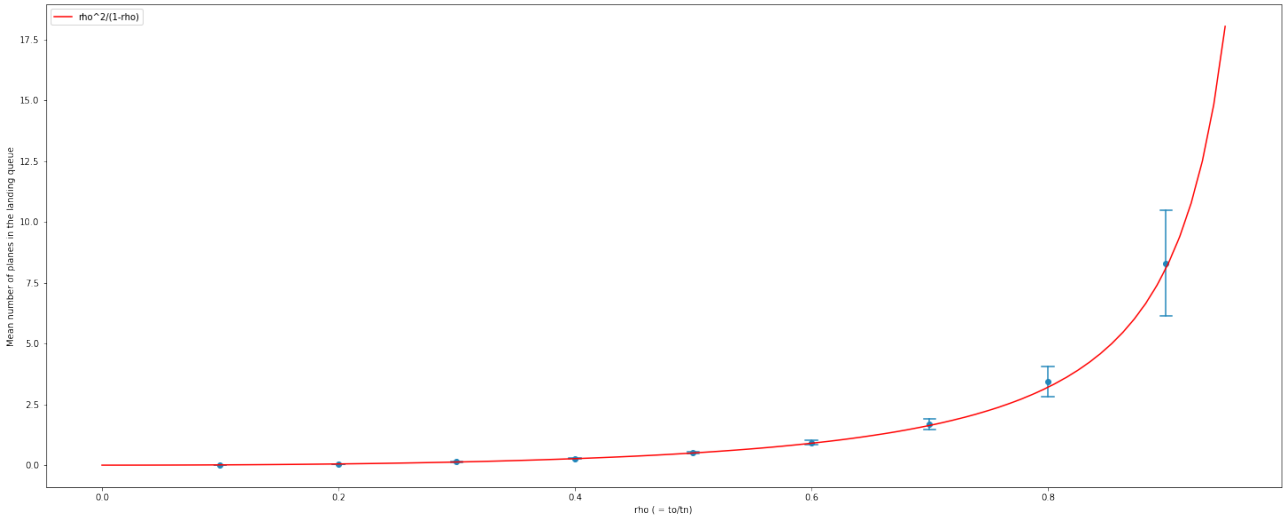


Figure 4: Mean waiting time in the land queue with varying *to* with 95% CI

So in figure 4 we can see how values from simulation fit very well theoretical model.

3.2.2 Exponential case with zero parking and landing time

Figure 5 shows a different particular case of the system, where parking time and landing time have been set to 0. It can be observed that the only time a queue forms in the high priority aircraft is when the server is serving a job from the low priority queue, otherwise, using the server zero service time for the high priority queue , it will be emptied immediately after the service of the job in progress. In this case, what happens is that as soon as there is an airplane in the high priority queue, two cases can occur:

- the server is free: in this case the queue empties, and all planes in the landing queue go to the take-off queue.
- the server is busy: a queue forms until the server is busy. However it is as if these jobs are already in the low priority queue as the high priority queue will empty in no time and they will be ready to take off.

What can be deduced is that the system in the figure is equivalent to an M/M/1, where the number of processes present in the queue of the M/M/1 equivalent system is equal to the sum of the processes present in the two queues of the system in figure 5.

In order to verify this theoretical model, we chose these parameters

- $T_l = 0$

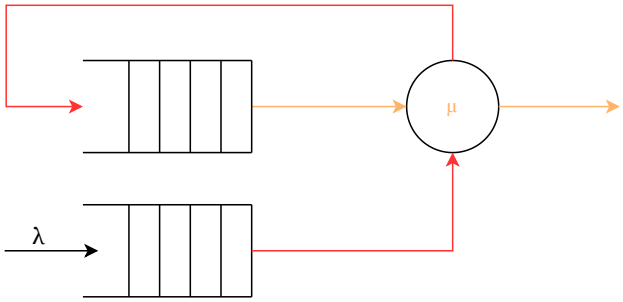


Figure 5: Simplified queueing system

- $T_o \sim exp(3),\ exp(6), .., \ exp(27)$
- $T_p = 0$
- $T_n \sim exp(30)$

For each configuration we have done 10 repetitions, the warmup time has been set to 1 day and the simulation length to 60 days. The result is shown in figure 6.

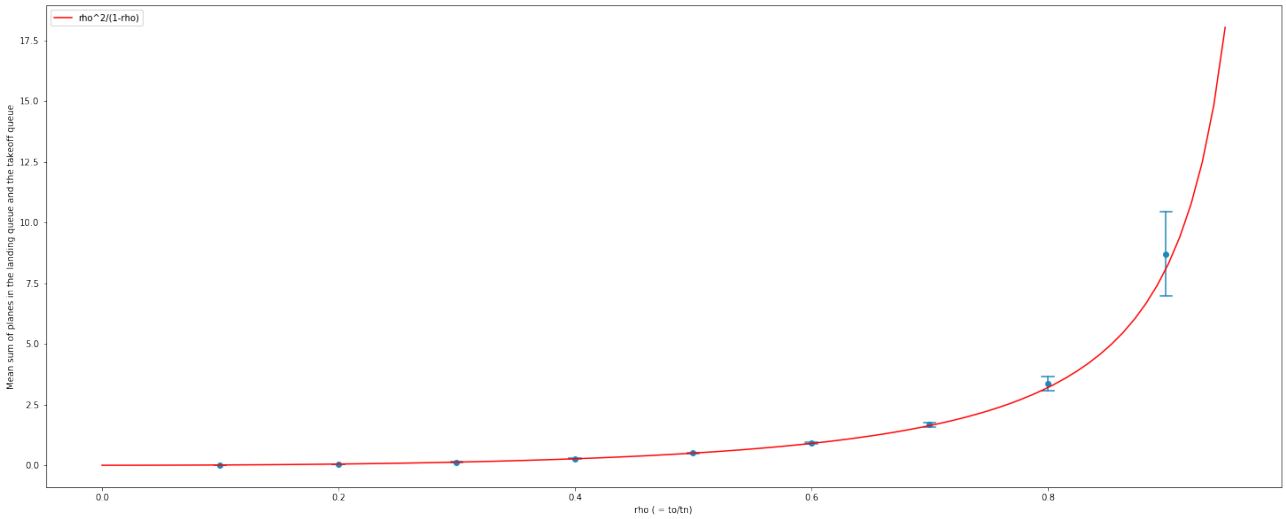


Figure 6: Mean sum of queueing length of both queues with varying *to* with 95% CI

On the x axis we have the value of ρ, while on the y axis the theoretical value, represented by the continuous curve, while the points represent the value assumed by the sum of the lengths of the tails at the respective utilization.

4 Deterministic case

In this section we present the results obtained from the analytical modelling of the system in a deterministic scenario where all the parameter of the system are in fact constants.

4.1 Analysis

We will present first few simple results about the stability condition and few special cases, then we will complete the analysis with the general case.

4.2 Stability condition

The system is stable – that is, queue lengths and parking occupancy do not growth indefinitely – if and only if:

$$t_l + t_o \leq t_n,\ t_l \neq t_n$$

Proof

- (The system is stable $\Rightarrow t_l + t_o \leq t_n$) $\Leftrightarrow$ ($t_l + t_o > tn \Rightarrow$the system is unstable): $t_l + t_o > t_n$ means the total time an airplane passes in the runway is greater than the time passing between two arrives in the system, that clearly cause instability in the system.
- ($t_l + t_o \leq t_i \Rightarrow$ The system is stable) $\Leftrightarrow$ (the system is unstable $\Rightarrow t_l + t_o > t_n$): the system is unstable if the queue length growth indefinitely, that is only possible if arrival rate is lower than total service time. Infinitely queue length means, from a certain point, the queues are never empty, but if the queue lengths are not zero an airplane is consumed in $t_l + t_o$ than is not greater than the interarrival rate planes enter the system, so the queue lengths must be bounded.
- $t_l = t_i$ causes the system to be unstable: if this is true the runway is always occupied by landing planes due the priority rule, thus no plane can leave the system and take-off queue grows indefinitely.

In the following cases we will always assume $t_l < t_n$.

4.3 New airplanes never have to queue before landing

This case is very simple to describe. Due to the deterministic nature of the system if the first plane is not queued none of the following will be so, after the transient, the system behaves in a perfect cyclic manner with period equal to t_n, the transient last exactly $t_l + t_p$ and the average parking usage is $\frac{t_p}{t_n}$.

Proof Be each plane in the system identified by a progressive number n starting from 0. The first plane enters the system at time 0, lands by time t_l and exit the building at time $t_l + t_p$. At this point it find the runway empty or occupied. If it is occupied the stability condition grants it will not impact on the second plane, so the next one will behave in the same way because it will not experience any delay. If it is not occupied, it will not impact on a following airplane if the following conditions are respected:

- $m\ t_n + t_l \leq t_l + t_p$ the plane starts take-off when the runway is empty
- $t_l + t_p + t_o \leq (m+1)\ t_n$ the plane completes take-off before the next plane arrives

If exists an airplane index m such that both conditions are matched the first plane does not impact on any following plane and this will be true for all planes, so all events will periodically repeat every tn.

Considering that there is no interaction between plane we conclude that planes enter the parking area with a period of t_n starting from t_l, that implies average parking occupancy is exactly $\frac{t_p}{t_n}$.

4.4 Case $\mathbf{t_l + t_o = t_n}$

The system is stable, the runway is always occupied (100% utilisation), the system behaviour is cyclical with period equal $t_o\ t_n$ and the average parking occupancy is $\frac{t_p}{t_n}$.

Proof The system is stable because the stability condition is respected. After the transient, the runway utilisation is exactly 100%, it is never empty. This is due to the deterministic nature of the system and the constant interarrival time. Any interval in which the runway would result empty would add a delay in the serving of future planes and summing this delay for all the intervals (after the transient, of course) would cause the total delay to sum to infinite. The period is exactly t_n because this is the sum of $t_l + t_o$. The transient is given by $t_l + t_p + t_o$ if when the first plane is ready for the take-off the runway is empty, because from that point the runway will never be empty anymore. From this point All arriving planes will experience a delay:

- Of 0 if the first plane finds the runway occupied when it is ready to take-off.
- Given by the time passed since the last airplane completed the landing phase and the moment in which the first one is ready to take-off if the runway is found empty.

In a similar way, all the planes ready for the take-off will experience a delay:

- Of 0 if the runaway is found empty when the first one is ready for the take-off
- Given by the remaining time the current landing airplane to complete this operation.

Due to the perfect periodic behaviour of the system, after the transient phase we can conclude the average parking occupancy is given by $\frac{t_p}{t_n}$. Being the case with runway usage equal to 100% the worst case, we can assert there cannot be a combination of parameters in which there are two or more airplanes in the same queue. Furthermore, we can also assert the waiting time before landing is strictly lower than to and the waiting time before take-off is less or equal to t_l due to the priority rules.

4.5 General case

This case is more complex, let's start with an example.

Example Let $t_l = t_p = t_o = 1$, $t_n = 2.5$.

1. time=0, airplane 1 arrives and starts landing
2. time=1, airplane 1 completes landing and enters the parking area
3. time=2, airplane 1 leave the parking area and starts take-off
4. time=2.5, airplane 1 continues take-off, airplane 2 enters the system and queues for landing
5. time=3, airplane 1 leaves the system, airplane 2 starts landing
6. time=4, airplane 2 completes landing and enters the parking area
7. time=5, airplane 2 leave the parking area, airplane 3 arrives and starts landing, airplane 2 queue for take-off
8. time=6, airplane 3 completes landing and enters the parking area
9. time=7, airplane 3 leave the parking area and starts take-off
10. time=7.5, airplane 3 continues take-off, airplane 4 enters the system and queues for landing
11. time=8, airplane 3 leaves the system, airplane 4 starts landing
12. ...

The period of this system is 5, the transient is 2.5 and the average parking occupancy is $\frac{2}{5} = 0.4$. Opportunely setting parameters, we could augment transient period (t_p growths $\Leftrightarrow$ transient time growths) and change the waiting time for arriving planes in the landing queue.

General analysis Analysing all the details of the general deterministic case give us no insights we could use to handle the stochastic case but there is an interesting result: fixed-size frames of airplanes share the same behaviour and "chaoticity" does not growth with time.

Proof Let's consider the first plane, it cannot suffer any interference from following arrives until it leaves the parking area. Here there are only two possible outcomes:

1. it leaves the system without forcing arriving planes to queue, possibly waiting in the take-off queue
2. it forces arriving airplanes to postpone landing by occupying the runway to take-off

The former has been previously described; the latter will be described here. Assume each airplane is identified by a progressive natural number with the first one identified by 0. The first airplane interferes with another one only if exists $m_1 > 0$ (index of a plane) such that:

- $t_l + t_p < m_1\ t_n$
- $t_l + t_p + t_o > m_1\ t_n$

that means plane m_1 found the runway occupied by the first plane.

First important consideration: all the plane before m_1 will experience the same behaviour of 0, that means all the planes with index comprises between 0 and $m_1 - 1$ will affect in the same way all planes between m_1 and $2m_1 - 1$. Being arrives at constant rate that means all planes in a m_1 size frame get involved in the same events every t_n. Due to the fact m_1 can be theoretically any integer opportunely choosing the parameters the average parking usage can slightly change over time, but it will always be approximately $\frac{t_p}{t_n}$ considering the system structure. We can always simplify the system analysis by taking the previous inequalities "modulo" t_n in order to get $m_1 = 1$.

Be $m_2 = 2m_1$, it is important to estimate how the system will behave with respect to m_2 to verify if its "chaoticity" will increase over time. Remembering the stability condition that also poses an upper bound for $t_l + t_o$, due to the periodic nature of the system m_2 will be potentially (if ever) affected by m1 only if the following inequalities hold:

- $m_1\ t_n + t_l + t_p < m_2\ t_n$ $\quad$ m_1 starts taking-off before m_2 arrives
- $(m_1 + 1)\ t_n + t_p > m_2\ t_n$ $\quad$ and leaves after it is arrived

It is very interesting that, by substituting m_2 with $2m_1$, the inequalities become again:

- $t_l + t_p < m_1\ t_n$
- $t_l + t_p + t_o > m_1\ t_n$

That means there is no dependency on time or plane number! This result is perfectly compatible with the conclusion of the worst case (100% utilisation) analysis, so we can always expect the queue lengths to be at most one.

4.6 Summary of deterministic analysis

Here we resume all the main results of this section.

- Average parking occupancy is about $\frac{t_p}{t_n}$
- There are never two airplanes in the same queue
- Waiting time in the landing queue is always less than t_o
- Waiting time in the take-off queue is always less or equal than t_l

5 Exponential case

5.1 Preliminary analysis

To find the warm-up time of the system we plotted the throughput of planes exiting the system over the rate of planes entering the system. We know that, if no planes are created or destroyed in the system, when the system reaches steady state that ratio will equal one. We plotted this value for many different configurations of input parameters to find an upper bound for the warm-up time, that we could safely use in all subsequent scenarios without needing to recalculate it every time. In Figure 7 we can see that the parameter that plays the biggest role in the warm-up time is, as one may intuitively guess, t_p. We can see that in general after $2t_p$ the system has reached a steady state.

To select the simulation duration we ran few simulations in the debug GUI provided by OMNeT++ and saw when the standard deviations of the variables describing the system became constant. For reasonable input parameters 60 days of simulation was enough. From the debug GUI we also saw that the system will reach a steady state iff:

$$t_l + t_o < t_n$$

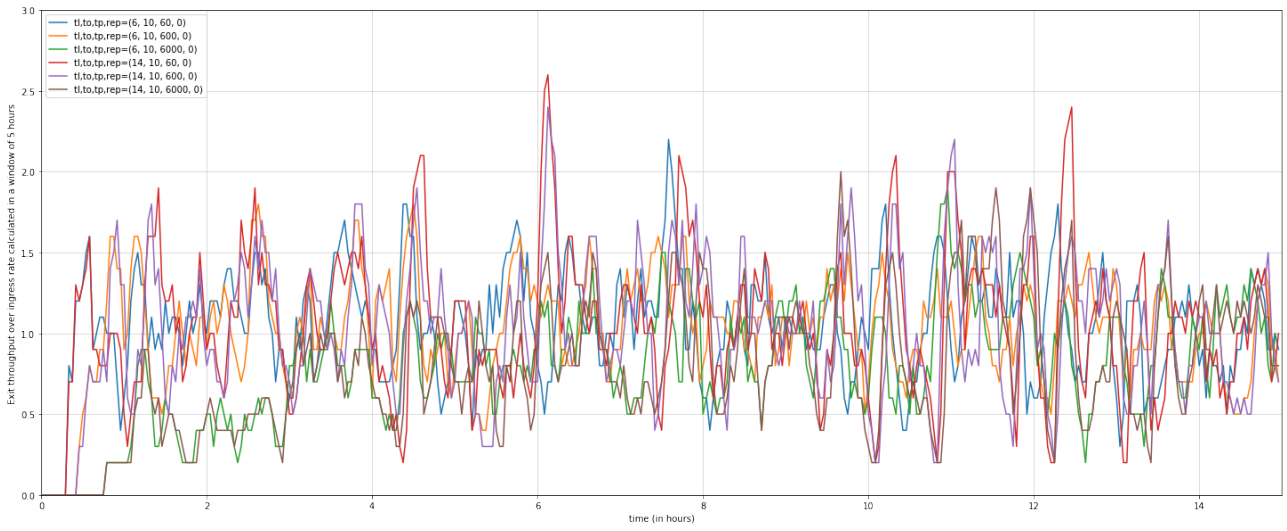
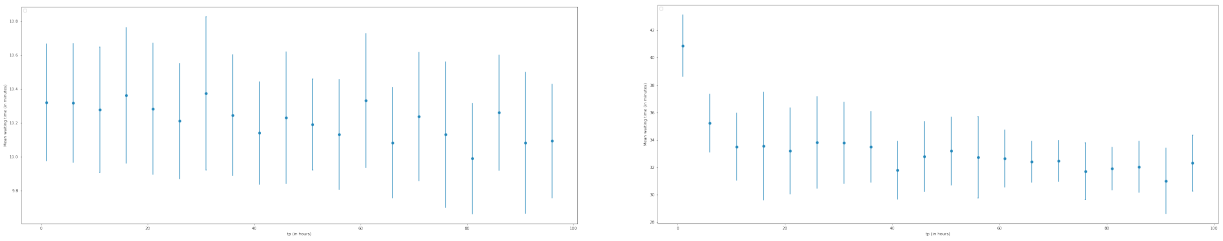


Figure 7: Exit throughput over ingress rate

5.2 Considerations on the parking time

It comes natural to think that the parking time does not have any effect on the time spent in the queues by the airplanes, since a greater parking time means that on average an airplane spends more time in the parking lot, thus only "moving to the right" the warm-up time. To check this assumption we plotted the time spend in the queue for both the landing queue and the takeoff queue, with varying values of t_p. As we can see in Figure 8 (a) the parking time has virtually no effect, and the waiting stays constant when varying t_p. In Figure 8 (b) we can see that for small values of t_p (less than 5 hours) the waiting time decreases when the parking time increases, after that the parking time has no effect in the waiting time.



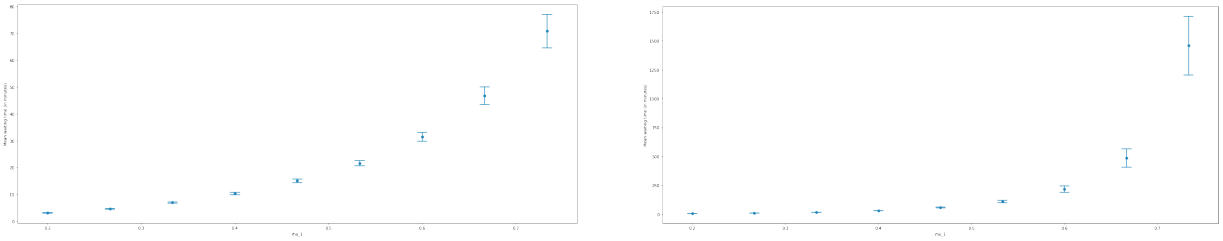
(a) Landing queue (b) Takeoff queue

Figure 8: Mean waiting time in the two queues with varying t_p with 95% CI

5.3 Waiting times at different utilization levels

Knowing that the parking time has a very marginal role in the waiting times, we will study how the waiting times are effected from different values for the utilization ρ_1 and ρ_2. Recall that we defined $\rho_1 = \frac{\lambda}{\mu_1} = \frac{t_l}{t_n}$ and $\rho_2 = \frac{\lambda}{\mu_2} = \frac{t_o}{t_n}$.

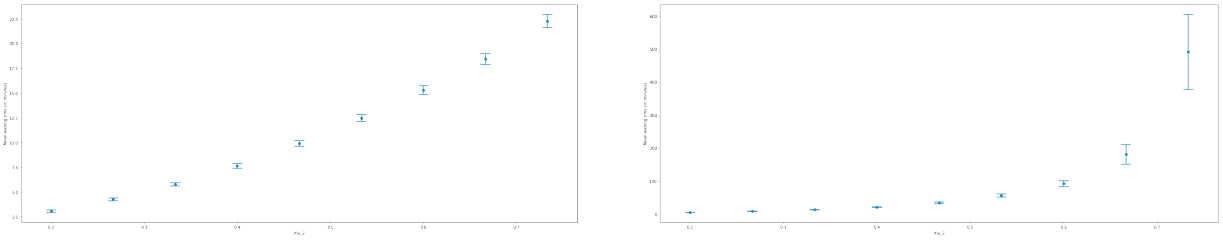
In Figure 9 we see that the waiting time for both queues are shaped like a Kleinrock function, as it was expected. Note that since $\rho_2 = 0.2$ the real utilization of the service center is $\rho_1 + \rho_2$, that's why it starts to grow more rapidly after $\rho_1 = 0.6$.



(a) Landing queue (b) Takeoff queue

Figure 9: Mean waiting time in the two queues with varying ρ_1 with 95% CI ($\rho_2 = 0.2$)

In Figure 10 we can see that the waiting time for the takeoff queue still follows a Kleinrock function, while the landing queue exhibits a linear behavior. That's because since the landing queue has a greater priority than the takeoff queue, as long as $\rho_1 < 1$ the waiting time will be finite. In other words, the landing queue will reach a steady state regardless of the value of ρ_2.



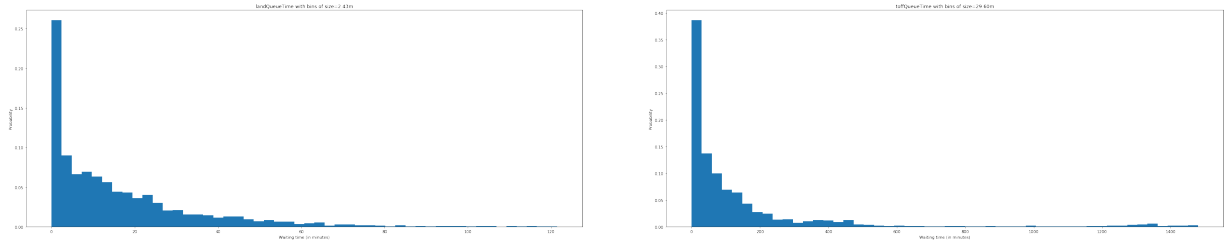
(a) Landing queue (b) Takeoff queue

Figure 10: Mean waiting time in the two queues with varying ρ_2 with 95% CI ($\rho_1 = 0.2$)

5.4 Waiting times EPDF

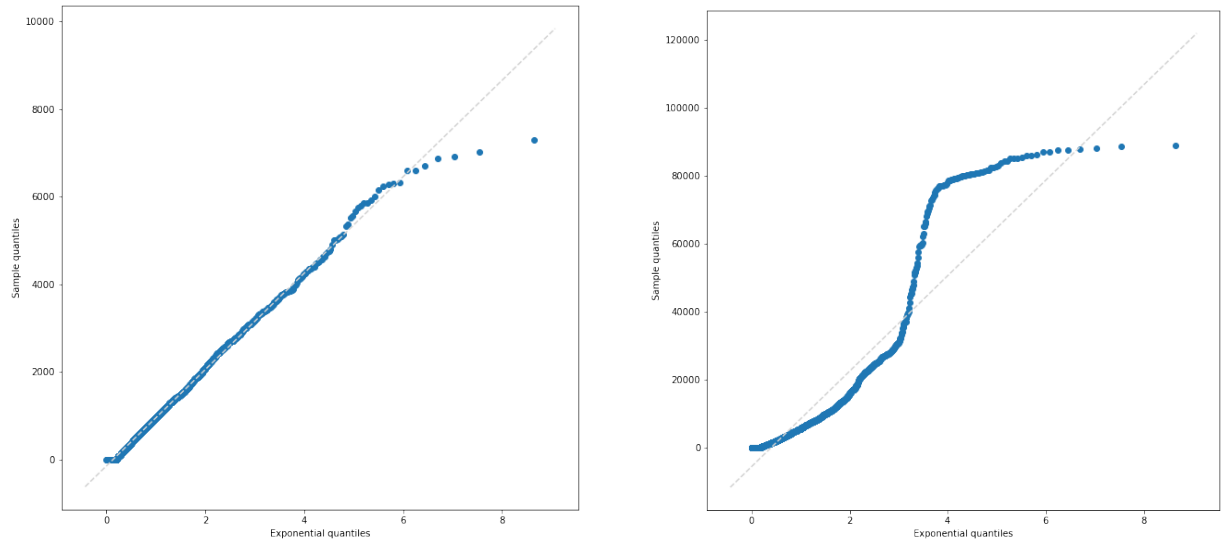
Finally, we want to study how the waiting time is distributed in the two queues. Figure 11 shows the EPDF for both queues, since all variables are exponential it makes sense that also the waiting times would be distributed somewhat similarly to an exponential. We can already see that the takeoff queue has a heavier tail than an exponential.

To conclude our study on waiting times we looked at QQ plots for the waiting time against the exponential, in Figure 12 (a) you can see that the exponential is a good fit for the landing queue. It is not however a good fit for the takeoff queue (Figure 12 (b)), this does not come as a surprise since it was clearly visible from the EPDF that the takeoff queue had a heavier distribution. We shall note that we tried other distributions for the takeoff queue, namely lognormal and pareto, but none of them was a god fit.



(a) Landing queue (b) Takeoff queue

Figure 11: Empirical PDF for waiting times with $\rho_1, \rho_2 = 0.4$



(a) Landing queue (b) Takeoff queue

Figure 12: QQ plot against exponential for waiting times with $\rho_1, \rho_2 = 0.4$

5.5 Parking occupancy study

From Little's Law we know that at the steady state the mean response time in a system is the mean number of jobs in the system over the mean arrival rate. Knowing that the time spent in the parking lot is on average (by definition) t_p we have that:

$$E[N] = \frac{t_p}{t_n}$$

where N is the number of airplanes in the parking lot. The data from the simulation confirms this result, see Figure 13.

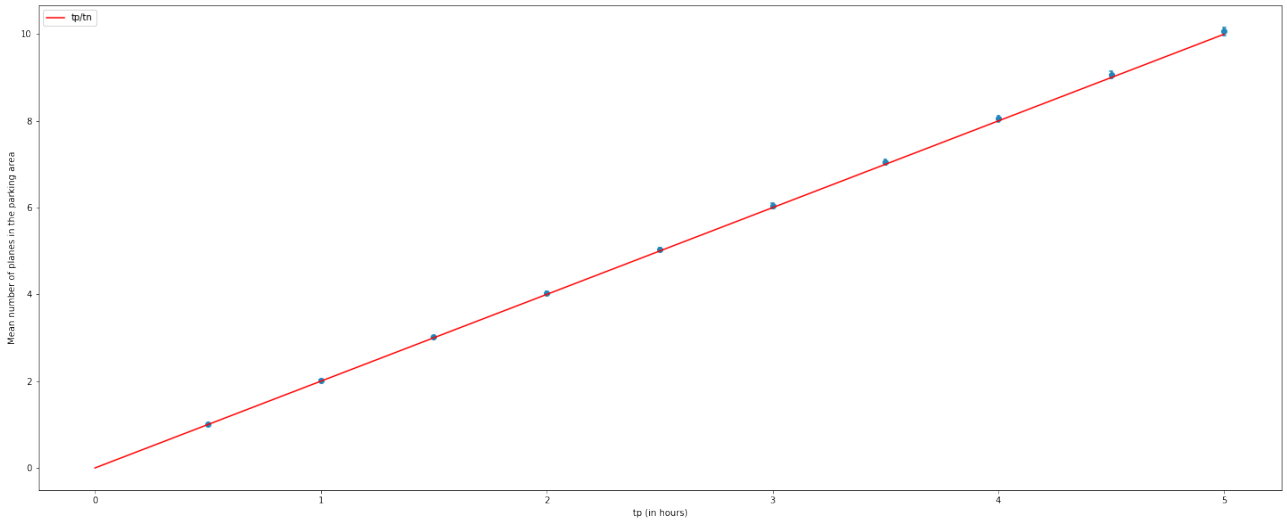


Figure 13: Number of planes in the parking lot with varying t_p with 95% CI

6 Final remarks

Some results were quite counter intuitive, so we will spend a couple words on them. The distribution of the waiting time for the takeoff queue for instance has a heavy tail, so while the mean waiting time was roughly 140 minutes, the max value was 1480, which is quite far away from the mean, and would be very catastrophic in a real world airport. The explanation for this is that in this scenario inter-arrival times and service times are exponentially distributed, thus intuition and "real world airport experience" does not help up because is based on events that are distributed more *normally*.