UNIVERSITÀ DI PISA

Performance and Evaluation of Computer Systems

Epidemic Broadcast

Francesco Berti

Iacopo Canetta

Giovanni Enrico Loni

ANNO ACCADEMICO 2023/2024

1 Introduction

This project's goal is to analize the spread within a system of information. In particular, given a floorplan with people dropped on it uniformly, a message is passed among them. Starting from a *Producer* and operating in slotted time, a message is broadcasted to all the users within a certain Radius (R). However, if an user receives more than one message at once, he is not able to decrypt it due to a *Collision* and discards all of them. If an user receives a message correctly, he will wait a random amount of time between 0 and W, and then, if he did not receive more than a maximum of messages (M) within the window, it will rebroadcast the Message, otherwise he does not; after that, the user goes idle in both the previous cases. Given this scenario, the spreading time of the message, the collisions, and the coverage will be analized in different ways and with different tools.

2 System Implementation

The system analized for this project is implemented on *OMNeT++*, a simulation framework in *C++*. This choice allows to run the simulation and retrieve all the data in a lightweight and easy way.

The simulation is composed by a network called *Floor* which includes a set of *Floor* and one *Supervisor*:

2.1 User

It models a single node able to receive messages from its neighbors and eventually relay one message. *User.NED* has the following fields:

- has two double values to act as coordinates of the plane
- has an int value *relayDelay* which indicates the time window to wait before relaying a message
- uses *direct transmission* to communicate with other nodes, this allows for better flexibility in case we want to do a simulation with non - static nodes
- collects *collisions per time slot* and *the evolution of the its coverage status*

In *user.cc* we can find the behaviour, the User class has of course been inherited by the *cSimpleModule OMNeT++*'s class. Each user keeps three counters:

- *elapsedTimeSlots* to keep track of how many time slots have passed
- *msgRcvInCurrSlot* to count the number of messages received in the current time slot
- *totCopies* to count the message's copies received in the window

The behaviour of User can be descripted as follows:

- The *initialize* method retrieves all the parameters from the network, finds all the neighbors which are stored inside the *neighbors* vector, registers the signals, and finally it either broadcasts the message, if the current user is the producer, or schedules a message to start the time slotting.
- The *broadcast* method simply loops the *neighbors* vector and send the message to all of them.
- *OMNeT++*'s handleMessage has been overridden so that, if the message received it is not a self message to simulate slotting, then *msgRcvInCurrSlot* is increased. This counter is then used inside the *handleTimeSlot* method to understand if there's been a collision or not.
- Counters *elapsedTimeSlots* and *msgRcvInCurrSlot* are used in the *handleTimeSlot* method, which advances the time slot and check whether the user can relay the message or not, it can be broken down into three sections:
  - Only one message was received in the last time slot: the message has been correctly received, the User communicates to the Supervisor that it can potentially forward the message to other users

– Multiple messages were received in the last time slot: the number of messages received in that slot are emitted to the corresponding signal
– Message has already been correctly received, waiting for W slots before relaying: once the W slots are passed, if the user has received less than M messages then the message is relayed, otherwise not. In both cases the *User* will comunicate with the *Supervisor* that it has lost the capacity of forwarding the message. The user will be idle from now on.

2.2 Supervisor

This module has the only purpose to end the simulation exactly when every user has done its job, so there are not anymore possibility to send messages. The reason for this is the User's behaviour: the User will keep on scheduling a time slot event, waiting indefinitely for another user's message, thus the only way to end the simulation is by means of the simulation time limit or by using a counter for the number of users that have the possibility to infect other users, if this counter goes to zero then the simulation can end.

The simulation is ended a few time slots after the counter goes to 0 since some events might be processed later, and so some users might still relay the message, but the broadcast time signaled by the Supervisor is the time slot at which the counter became equal to 0, thus giving a precise measurement.

2.3 Floor

Floor is the name of the network that accomodates all the previous submodules, the .NED file specifies the value for all the parameters of interests (N, R, W, M) as well as allocating users and defining their parameters.

2.4 Randomizer

The random numbers generated for each run of the simulation are the producer index and, for each user, the coordinates and dimension of the window. These were generated by means of *OMNeT++*'s default RNG using a uniform distribution inside the network .NED file and then by retrieving values inside the initialization method using *par* method.

2.5 Data retrieval

Data has been retrieved exploiting *OMNeT++*'s signals, the list of collected signals is the following:

- Overall time to broadcast the message (scalar)
- Overall number of infected users (scalar)
- Evolution through time of the number of collisions for each user (vector)
- Evolution through time of coverage status for each user (vector)

The vectors are retrieved for some in depth statistics considering each node individually. However, this process can take a lot of storage space and slow down the process; for bigger practical cases, only the scalar cases would be kept, in order to have lighter results and use all the capabilities offered by *OMNeT++*.

3 Performance Analysis

The first step for the Performance Analysis is choosing the parameters configurations. In particular, the area of the floorplan is kept constant for all the runs. The proportions between the floorplan sides, however, is modified from a standard square figure ($200 \cdot 200$) to a rectangle ($800 \cdot 50$), to check how that effected the Performance. The other parameters, so R, M, W, and N are set depending on the tests. Thanks to the *omnetpp.ini* file, setting the configurations and the repetitions is easy and clear.

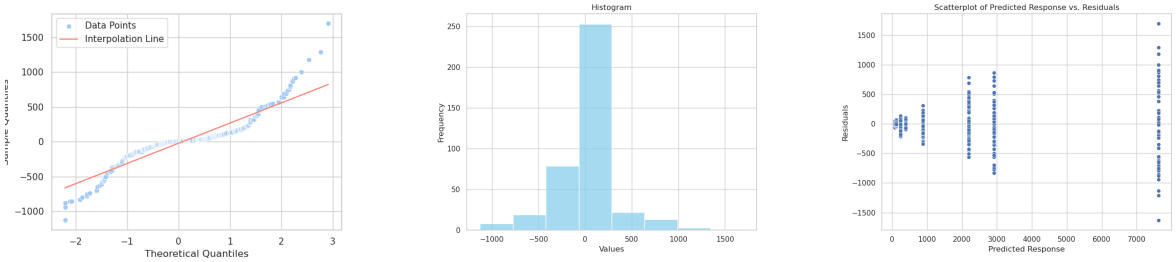
Once the data regarding the SimulationTime, the Sollisions, and the Coverage per run are retrieved, it needs to be parsed correctly and analized. A *Python* notebook is used for handling results and plotting graphs, making this process easy and quick.

3.1 Factorial $2^k r$ Analysis with fixed population's density

For a first analysis, the Factorial $2^k r$ Analysis is run on three different scenarios, defined by the population's density. The first and most important scenario is the "average" case, with a medium density of $0.02\frac{users}{m^2}$, and this is also the case for deeper analysis, due to its realistic characteristics; the other two, instead, are created to push the system to its limit, with a really low ($0.004\frac{users}{m^2}$) and really high density ($0.045\frac{users}{m^2}$). To achieve results with consistent statistical properties, every test is repeated 50 times, randomizing each time the distribution of the nodes.

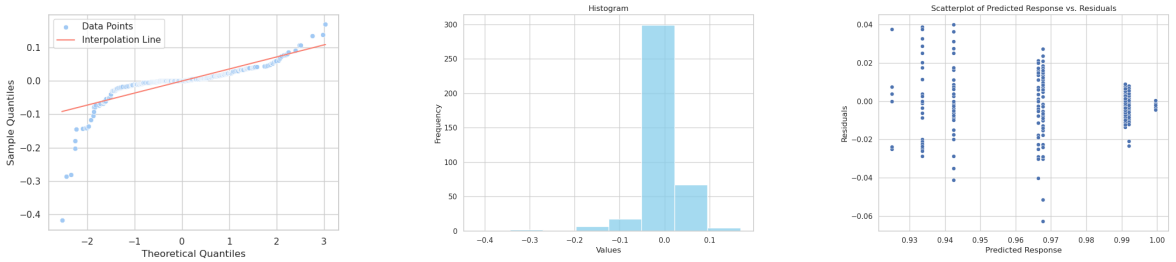
Once the results were ready, first analysis performed was the factorial $2^k r$ analysis, to show and to recognize which parameters were the ones effecting the most the tests, and in which combinations. For this purpose, a custom function has been created, and it revealed how each scenario was effected by different parameters.

The factorial analysis, to be valid, requires that the residuals (errors) to be IID Normal RVs with a null mean and a constant standard deviation. To test this hypothesis, a QQ plot is the first choice and it clearly proves, as showed below, how the mean is null; unfortunately it is not enough for proving the Normal distribution, since it shows some suspicious waves. For this very reason, all the QQ plots are followed by a more intuitive hystogram, which shows a Normal distribution. However, using the scatterplot, it is easy to see as the variance of the residuals is not finite, showing a clear trend. For this reason the linear model for the Factorial $2^k r$ Analysis is invalidated and a new one must be found.



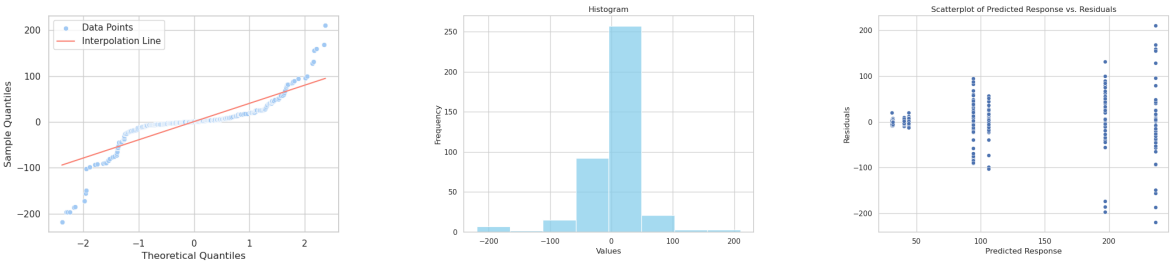
(a) QQ-PLOT NORMAL (b) HISTOGRAM (c) HISTOGRAM

Figure 1: COLLISIONS RESIDUALS DISTRIBUTION ANALYSIS



(a) QQ-PLOT NORMAL (b) HISTOGRAM (c) VARIANCE SCATTERPLOT

Figure 2: COVERAGES RESIDUALS DISTRIBUTION ANALYSIS



(a) QQ-PLOT NORMAL (b) HISTOGRAM (c) VARIANCE SCATTERPLOT

Figure 3: TIMES RESIDUALS DISTRIBUTION ANALYSIS

3.2 Factorial $2^k r$ Analysis with logarithmic model

Since the previous model is rejected, others are tried. In particular, the one resulting the best one is the logarithmic one, having $y' = log(y)$. With this new model, the hypothesis are verified and the variance is finite, making the Factorial $2^k r$ Analysis valid. The only scatterplot with a worse variance is the coverage without ouliers, which however has all value close to 99%, making it less relevant. The independence is also tested and verified with a scatterplot; the scatterplot, in fact, does not show any trend depending on the run number. The picture below, however, is zoomed in to show better the results, since in the original the values look compressed and unreadable due to outliers.

	I	**R**	**M**	**W**	**RM**	**RW**	**WM**	**RWM**	**Y**
	1	-1	-1	-1	1	1	1	-1	5.500
	1	1	-1	-1	-1	-1	1	1	7.969
	1	-1	1	-1	-1	1	-1	1	5.965
	1	1	1	-1	1	-1	-1	-1	8.934
	1	-1	-1	1	1	-1	-1	1	4.398
	1	1	-1	1	-1	1	-1	-1	6.775
	1	-1	1	1	-1	-1	1	-1	4.746
	1	1	1	1	1	1	1	1	7.682
SUMS	51.969	10.750	2.685	-4.768	1.059	-0.125	-0.175	0.058	
MEAN QI	6.496	1.344	0.336	-0.596	0.132	-0.016	-0.022	0.007	**SSE**
50 * 8 QI^2	16879.812	722.325	45.062	142.082	7.015	0.098	0.190	0.021	12.471
PERCENTAGES		**0.7773**	**0.0485**	**0.1529**	**0.0075**	**0.0001**	**0.0002**	**0.00002**	**0.0134**

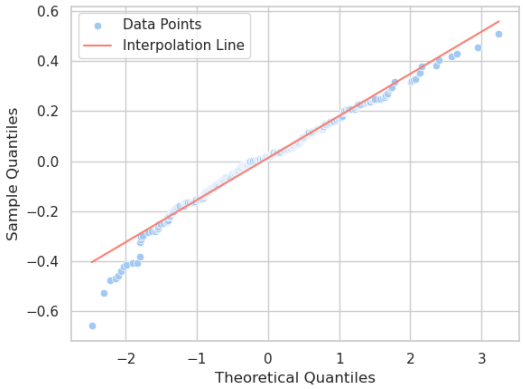
Table 1: Collisions MidDensity LOG 2KR

	I	R	M	W	RM	RW	WM	RWM	Y
	1	-1	-1	-1	1	1	1	-1	4.2449
	1	1	-1	-1	-1	-1	1	1	3.4214
	1	-1	1	-1	-1	1	-1	1	4.5594
	1	1	1	-1	1	-1	-1	-1	3.4486
	1	-1	-1	1	1	-1	-1	1	5.3689
	1	1	-1	1	-1	1	-1	-1	3.7709
	1	-1	1	1	-1	-1	1	-1	5.2552
	1	1	1	1	1	1	1	1	3.6897
SUMS	33.7588	-5.0977	0.1467	2.4105	-0.2548	-1.2292	-0.5367	0.3198	
MEAN QI	4.2199	-0.6372	0.0183	0.3013	-0.0319	-0.1536	-0.0671	0.04	**SSE**
50 * 8 QI^2	7122.87	162.41	0.1346	36.31	0.4058	9.4431	1.8001	0.6391	100.735
PERCENTAGES		**0.5207**	**0.0004**	**0.1164**	**0.0013**	**0.0303**	**0.0058**	**0.002**	**0.323**

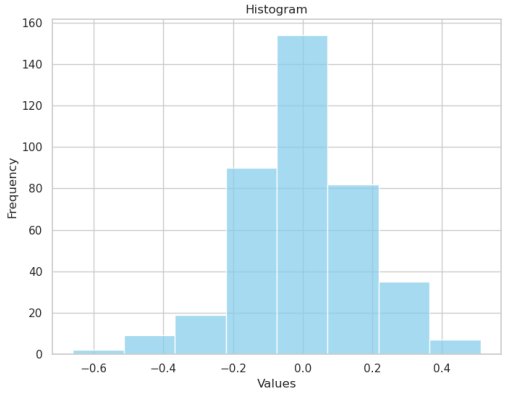
Table 2: SimulationTime MidDensity LOG 2KR

	I	R	M	W	RM	RW	WM	RWM	Y
	1	-1	-1	-1	1	1	1	-1	-0.3992
	1	1	-1	-1	-1	-1	1	1	-0.0331
	1	-1	1	-1	-1	1	-1	1	-0.0728
	1	1	1	-1	1	-1	-1	-1	-0.0091
	1	-1	-1	1	1	-1	-1	1	-0.1407
	1	1	-1	1	-1	1	-1	-1	-0.0080
	1	-1	1	1	-1	-1	1	-1	-0.0582
	1	1	1	1	1	1	1	1	-0.0005
SUMS	-0.7213	0.6202	0.4405	0.3068	-0.3774	-0.2394	-0.2603	0.2274	
MEAN QI	-0.0902	0.0775	0.0551	0.0383	-0.0472	-0.0299	-0.0325	0.0284	**SSE**
50 * 8 QI^2	3.2518	2.4037	1.2125	0.5881	0.8900	0.3583	0.4235	0.3233	7.9439
PERCENTAGES		**0.1700**	**0.0857**	**0.0416**	**0.0629**	**0.0253**	**0.0299**	**0.0229**	**0.5617**

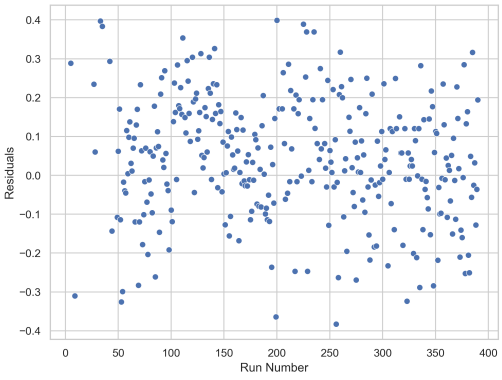
Table 3: Coverage MidDensity LOG 2KR



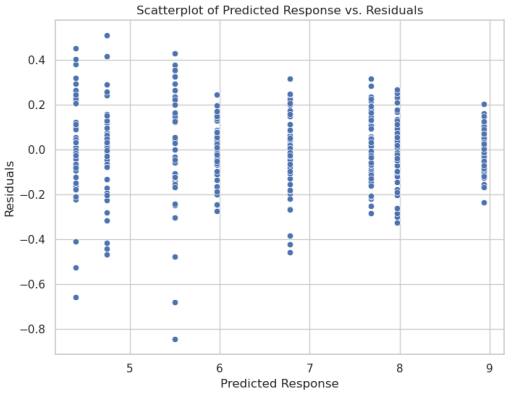
(a) QQ-PLOT NORMAL



(b) HISTOGRAM

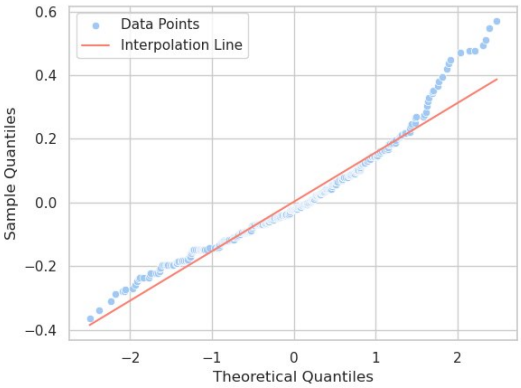


(c) INDEPENDENCE SCATTERPLOT

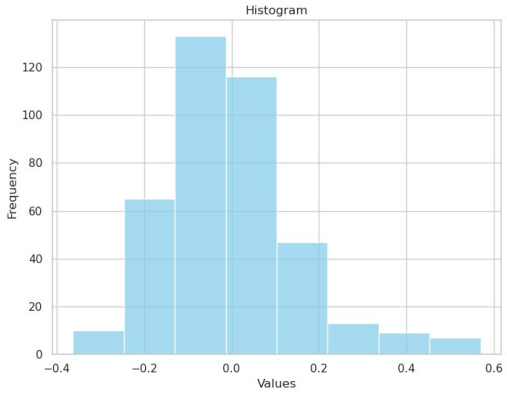


(d) VARIANCE SCATTERPLOT

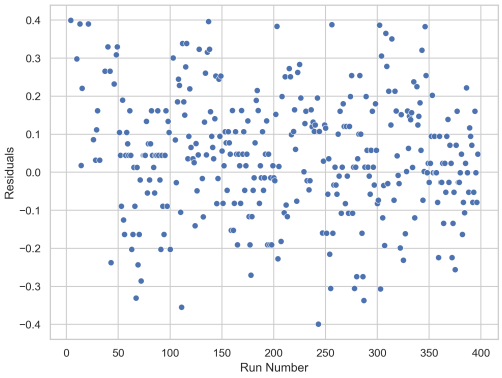
Figure 4: COLLISIONS RESIDUALS DISTRIBUTION ANALYSIS



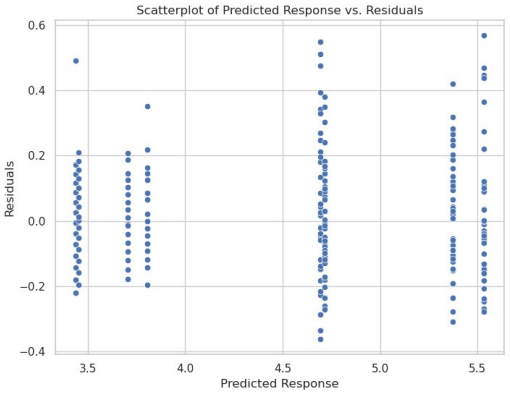
(a) QQ-PLOT NORMAL



(b) HISTOGRAM



(c) INDEPENDENCE SCATTERPLOT



(d) VARIANCE SCATTERPLOT

Figure 5: TIMES RESIDUALS DISTRIBUTION ANALYSIS

The results for the HighDensity and LowDensity are omitted to ease the reading but the results are summarized as follows:

- *MidDensity*: the most important parameter is the Radius, effecting heavily the results of all three the measured units; however the Coverage and Collisions are effected also by MaxMessage, while Collisions is instead effected also by Window; some runs, however, show very low coverage due to some unlucky placement of the starting node in the corner and with few nodes around, causing the error to increase heavily: for this very reason, these few results are discarded as outliers;
- *LowDensity*: this scenario is effected mostly by the Radius, since due to the low density of nodes, the graph of the connections results easily disconnected for low Radius, reducing Coverage, Collisions, and also the BroadcastTime, due to an early stop;
- *HighDensity*: in this last scenario the Radius greatly increases the connections between nodes, effecting heavily both the BroadcastTime, but also the number of Collisions; the Coverage, instead, depends mostly on MaxMessage and Window, which decide if the information gets relayed or not, and eventually getting it stuck early in the spreading.

Furthermore, as said before, the errors are very high due to disconnections in the graph; for this reason, other models are tried for the fit, but ibly resulting in increasing even more the error. For this reason, the behaviours above can be accepted, but with a random component due to unlucky placement of nodes. The same scenarios are also tested on the rectangular floorplan, but the shape caused the connections graph to disconnect easily, resulting in the Radius to be the most effecting parameter, and the error to be high.

3.3 Factorial $2^k r$ Analysis with varying population's density

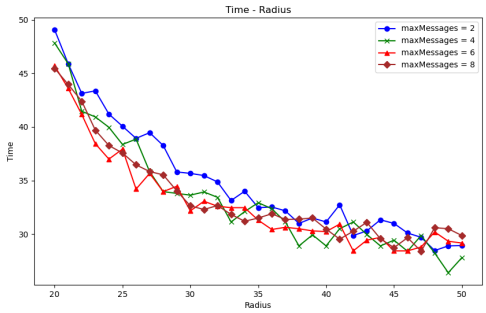
The next analysis is another Factorial $2^k r$, but only considering scenarios close to the MidDensity, adding the number of Users N to the test's parameters. As expected, however, all the measured performance depends on R and N, both individually and combined; these parameters, in fact, effect directly the connections' graph, causing mostof the information flow's direction of and eventual disconnections and partitions. For better Analysis, however, the population is be kept still for the rest of the project, for having a constant density and avoiding that changing both N and R would overlap in effects and just make the data reading confusings.

3.4 In - depth Analysis

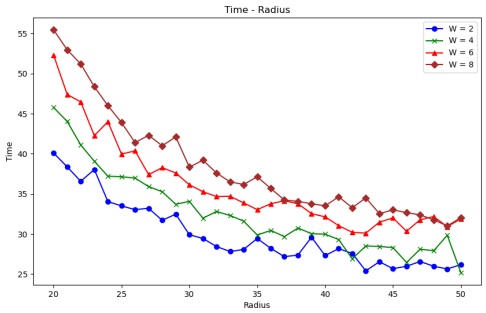
From the project assignment:

> A random user within the floorplan produces a message, which should ideally reach all the users as soon as possible

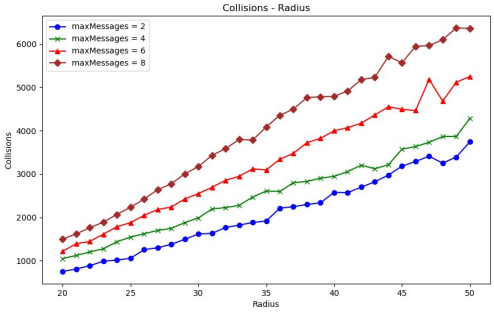
thus collisions are treated as less important compared to BroadcastTime and Coverage. The $2^k r$ Analysis highlights how the interplay between W and M is negligible, thus separated tests are conducted using the MidDens scenario and varying the Radius:



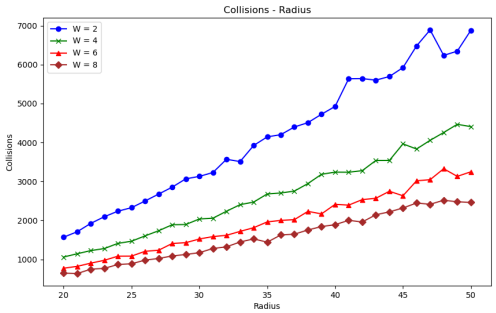
(a) TIME depending on Radius and MaxMessage



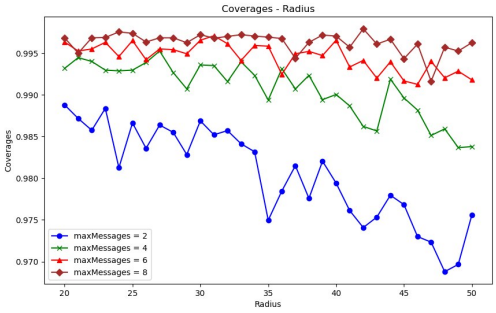
(b) TIME depending on Radius and Window



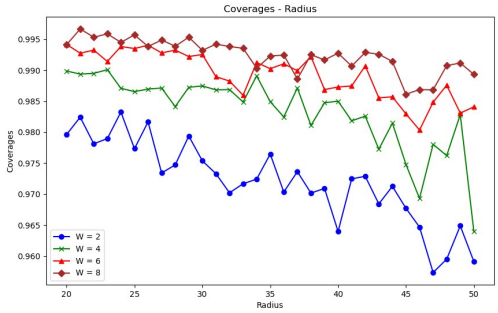
(c) COLLISIONS depending on Radius and MaxMessage



(d) COLLISIONS depending on Radius and Window



(a) COVERAGES depending on Radius and MaxMessage



(b) COVERAGES depending on Radius and Window

As shown above, it is possible to see how higher values of W reduce Collisions but increases SimulationTime and decreases Coverage. Higher values of M, instead, increase Coverage but also Collisions For the following tests parameters $W = 6, M = 5$ are chosen as they offer a fast broadcast (higher W altered BroadcastTime in negative but by a relatively small amount) without completely disregard for collisions.

3.4.1 Radius in - depth Analysis

After the Factorial $2^k r$ Analysis, it is clear how it is necessary to study better the performance's behaviour depending on the Radius. For this very reason, more tests are performed on the MidDensity scenario, testing more values of the Radius. The results, as shown below, are easy to understand.

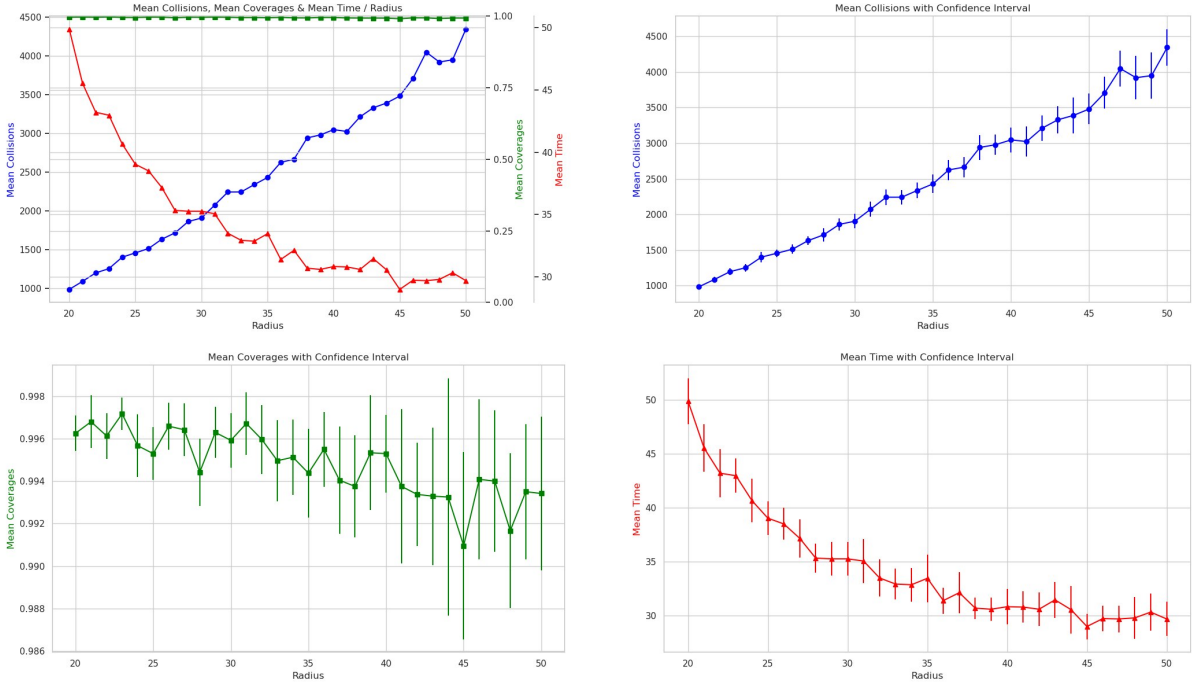


Figure 8: Large Radius Values Performance

As the radius increases, more nodes are reached at once, decreasing the SimulationTime, but also increasing the Collisions; the Coverage, instead, is fairly stable aroud 99%, even if slowly decreasing and showing higher variance increasing with R. This happens due to more Collisions happening and the Trickle Relay policy.

3.4.2 Larger Radius Values

The decreasing of the Coverage value starting around $R = 45$ shows a pattern worth studying for large value. For this reason, more tests are done incrementing R 20 by 20, to avoid creating GBs of data to analyze. These new tests show, however, interesting trends; the Coverage, in fact, starts decreasing due to an increasing number of collisions, limiting the spread of information. Increasing even more the Radius, however, makes the producer able to directly reach most of the users in the floorplan, reducing both Collisions and increasing the Coverage. The only metric showing a regular trend was the RunTime, which was always decreasing, due to the larger broadcasts and nodes reached at once. These extreme Radius values, are not studied further, since it is a less realistic simulation and a broadcast that powerful would be expensive in terms of technology and power consuption.

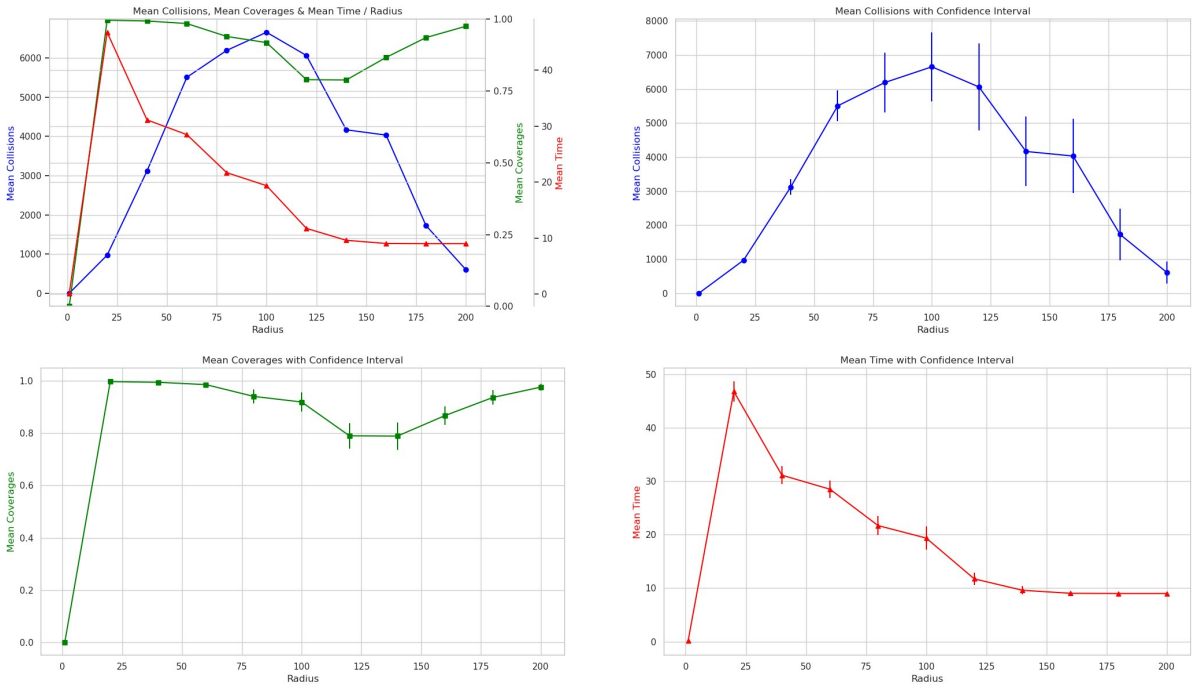
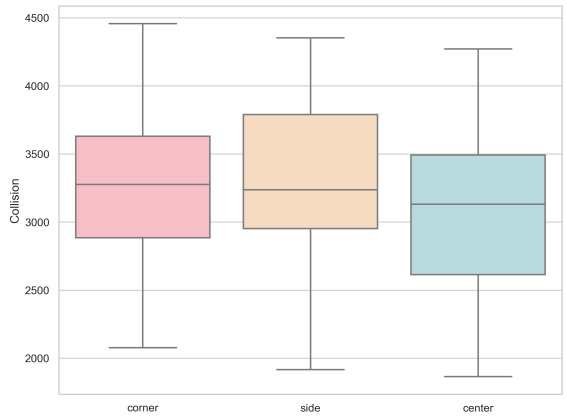


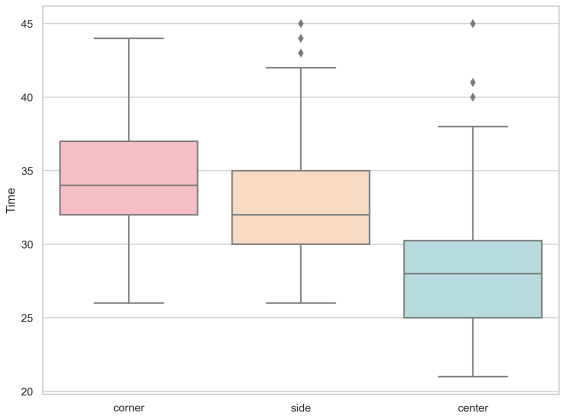
Figure 9: Large Radius Values Performance

3.5 Distance from producer analysis

The last analysis performed is studying the producer's position. For this reason the simulations uses a combination of parameters that offers good performance, in particular $R = 40$, $W = 6$, and $M = 5$ in a MidDensity scenario. $R = 40$ is not the best compromise between SimulationTime and Collisions, but, as said, before, SimulationTime has priority over the other; for this reason the value chosen is the one where the curve starts flattening. Furthermore, 3 main configurations are created, with 100 repetitions each: the first configuration keeps the producer always in a corner, the second one always on a side of the square, and the last one in the center. This allows to check how the RunTime and the collisions varied depending on that position: the collisions are not really effected, but the time, even though the Confidence intervals do not allow a perfect comparison, shows a descending trend depending on the configuration.

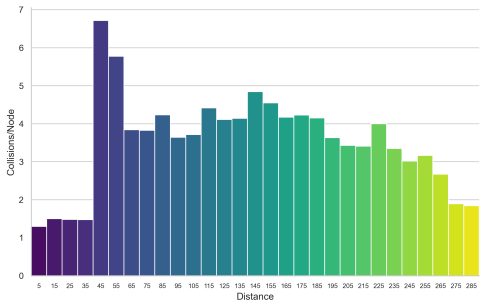


(a) Collisions depending on producer's position

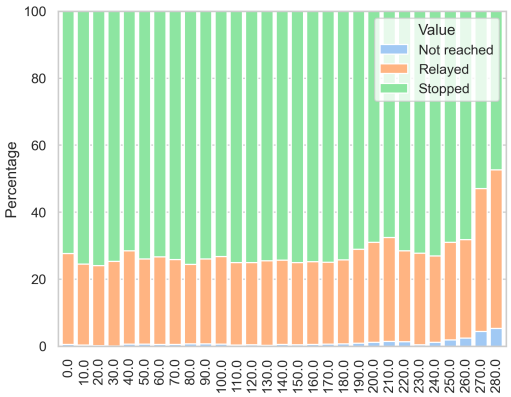


(b) RunTime depending on producer's position

The distance from the producer is also analized, and how it effects the collisions and coverage of the single nodes. As shown in the graphs below, with high distances from the producer, nodes tend to relay more the message or not receive it at all; this fact is easy to explain, since the nodes receive less messages and they have lower probabilities of getting M messages. For what concerns the collisions, they are low under the Radius range from the producer (40), and they spike with it, then they stay approximately constant with some little spikes, usually on Radius multiples.



(a) Nodes' collisions per distance



(b) Nodes' coverage per distance

4 Conclusion

This project shows an performance analysis of a system, studying both with mathematical formulas and tests, how changing a parameter can effect the whole system. This study could be in fact useful if applied on a similar real world case, and, depending on the constraints and the objectives, could find the optimal parameters. In general, if collisions are a concern, having a large Window and small MaxMessage can help reduce them, since a smaller number of users is able to relay the message without impacting coverage.