1 Comments

1.1 Validation

After the analytical modeling and the simulation, the *validation* step is needed to check how *close* the values predicted by the model are with respect to the ones measured during the simulation, to understand if the model reflects the "real" (simulated) behavior of the system.
Indeed, the theoretical analysis may prove that the performance guarantees are respected, while the simulation shows the opposite: for this reason, validation is needed to build *credibility* for the analytical model and the simulation, to make them acceptable for the domain experts that will use it.
There are different techniques that can be applied (expressed in [**?**]).
The main guidelines followed in this project were:

- run the simulation under *different settings and inputs* and check that the output are *reasonable*;
- *verify* that the estimates of the parameters given by the two methodologies are *close*, to build a certain degree of *confidence* about the correctness of the analysis.

Thus, this step was performed by *comparing* the results given for every metrics in each of the four scenarios taken into account for the three analyzed algorithms.

1.1.1 Lazy Reliable Broadcast

1. Considering a system of 6 nodes and no failures, the metrics' values are quite close, except for the average throughput and utilization of the channel (e.g. the gap between $U_{channel,simulation}$ and $U_{channel,analytical}$ is of 1.7%);
2. Adding the possibility to crash for the processes, also the node's metrics are more distant in the two analysis but the biggest difference is in the average response time R, since it is smaller with respect to the one of the first scenario for the analytical model but for the simulation we get the opposite result.
3. In a system of 30 nodes with no failures, the metrics are estimated almost in agreement by the two techniques.
4. Considering also crash failures for processes, the estimations of the response time and of the throughput are within a small gap while the utilization is profoundly overestimated by the analytical model ($U_{channel,simulation} = 3.972\%$ and $U_{channel,analytical} = 13\%$).

1.1.2 Eager Reliable Broadcast

1. In a system of 6 nodes and no failures, the analytical model slightly underestimates R (of 0.02 s) and the other parameters. The main difference is in the utilization, where $U_{channel,simulation} = 28\%$ and $U_{channel,analytical} = 18\%$.

2. Conversely, adding the crash of the processes, the analytical model overestimates the parameters, quantified as almost the double of the ones returned by the simulation.

3. Considering a system of 30 nodes with no failures, the measures of the average throughput and utilization for the channel and the node are quite close, but the response time experienced in the simulation is much higher.

4. With crash failures for processes, the response time R is similar in the two analysis but the other metrics are all far from each other: especially, they are overestimated by the analytical model.

1.1.3 Eager Probabilistic Broadcast

1. Considering a system of 6 nodes and no failures, the measurements of throughput and utilization are close, but the response time R is underestimated when computed analytically.

2. Adding the possibility to crash for the processes, the results are more distant, with the simulation that reports smaller values for each parameter.

3. In a system of 30 nodes with no failures, the metrics are closer, except for the response time R, which is almost the double for the simulation.

4. Also considering crash failures for processes, the response time is still considered to be double by the simulation with respect to the one returned by the analytical model but all the other metrics are overestimated by the latter modeling.

After the observations done in the validation phase, in real systems the *calibration* is executed, to tune the model accordingly to make it fitting better the simulated system's behavior (it was not performed during this project).

1.2 Choosing the Right Algorithm

Once validated the results, some considerations can be done relying on these outputs. In particular, based on the simulation's outputs, we can observe that the *Lazy Reliable Broadcast* algorithm is the one that performs better in every metrics almost in the majority of the scenarios, both for small and large systems and with or without crash failures for the nodes.

This result confirms what we could already intuitively state, since this protocol is minimizing the number of retrasmissions of the same messages.
The only exception is the case of a *small* system (assumed to have 6 nodes in this project) with *crashes* where, conversely, the *Eager Probabilistic Broadcast* behaves better in all the parameters.

1.3 Limitations

The reason of the *discordance* in the results returned by the simulation and by the analytical modeling needs to be searched in the fact that both the methodologies present their own *limitations*, which are quite complex to mitigate.

1.3.1 Limitations in Simulation

The simulation's outputs are mainly influenced by the *intrinsic* limitations of the simulation framework and of Python, the programming language used for its development. Indeed, despite the multi-threading implementation, the Python's interpreter runs in a *single process* and the framework deals directly with the *sockets*, opening and closing them "by hand" and increasing the *overhead*.
Another influential factor is that each channel is *bidirectional* and includes a queue for each process on its sides but it implements the receiving and the forwarding of the messages in two threads, working in both the directions. This is also why the utilization of the channel is often *underestimated* by the analytical modeling.
Moreover, another limitation is the presence of the processes all *on the same machine*, sharing its resources: this is exactly the opposite of real systems, in which they are distributed over several entities in a network.
For this reason, also the number of processes which can be simulated by the framework *cannot* be too *high*, to avoid invalidating the measurements due to the resource's contention. Besides, the values of the metrics can be affected by a huge variance which can be *mitigated* only running a *much larger* number of experiments.

1.3.2 Limitations in Analytical Modeling

On the other side, analytical modeling is not always easy to perform, becoming even almost *impossible* to carry out for representing complex behaviors and systems.
In the case of this analysis, the main complication was in understanding the arrival rate of the processes when retrasmission was needed due to crashes: indeed, the move from a single class to a multiple class open network was necessary to have a more "realistic" modeling of the nodes' load (still not so accurate).
Talking of load, it is also worth noticing that the analytical models are *not load dependent* as the real systems they aim to represent, whose performances may vary with its fluctuations.

Another aspect to consider is that the analytical results were computed assuming always the *worst case* scenario in terms of messages that a process receives.

1.3.3 Limitations of the Project

There are also other limitations in the way the project was carried out, since no techniques to increase the *validity* and the *credibility* of the results were embedded. Some of them (reported by [**?**]) are that, as a modeler, I could not benefit from a complete and accurate set of information about the real system (since there isn't one) and a formal (quantitative) *results validation* leveraging on statistical procedures was not performed, in favor of a simpler qualitative analysis.