Assignment No. 4 Report

Performance of a Packet Switching Connection

Desempenho e Dimensionamento de Redes
MIECT, DETI, UA

Gabriel Silva (85129) & Gonçalo Vítor (85119)

Simulator 1 Results

With 10 simulations and P=1000

Case	lambda	C	f	Packet Loss(%)	Average Delay(ms)	Maximum Delay(ms)	TT(Mbps)
A	100	2	100000	0±0	4,4855±0,0895	20,499±1,6715	0,6674±0,0092
B	200	2	100000	0±0	8,3462±0,356	38,613±3,7218	1,3383±0,0187
C	100	2	10000	0±0	4,5911±0,0662	20,9557±1,6502	0,6714±0,0096
D	200	2	10000	0,0398±0,05	7,9497±0,3147	35,7897±1,969	1,3406±0,0199
E	500	10	100000	0±0	0,9316±0,0264	4,2167±0,5664	3,3802±0,0617
F	1000	10	100000	0±0	1,5596±0,0756	6,7846±0,5808	6,5901±0,1386
G	500	10	10000	0±0	0,9±0,0129	3,9928±0,3367	3,361±0,0336
H	1000	10	10000	0,01±0,0164	1,6199±0,0687	7,7158±0,4764	6,6971±0,1441

With 10 simulations and P=100000

Case	lambda	C	f	Packet Loss(%)	Average Delay(ms)	Maximum Delay(ms)	TT(Mbps)
A	100	2	100000	0±0	4,5379±0,0101	31,2115±1,3121	0,6679±0,0013
B	200	2	100000	0±0	8,1677±0,0535	67,3519±5,1958	1,3355±0,0034
C	100	2	10000	0±0	4,5413±0,0085	32,6535±1,3045	0,667±0,0014
D	200	2	10000	0,0514±0,0045	8,0738±0,0384	44,7527±0,3633	1,3352±0,0033
E	500	10	100000	0±0	0,9097±0,0015	6,7701±0,4261	3,3365±0,0062
F	1000	10	100000	0±0	1,6272±0,0109	13,4899±1,1569	6,6668±0,01
G	500	10	10000	0±0	0,9109±0,0016	6,6202±0,1798	3,3462±0,0035
H	1000	10	10000	0,0567±0,0093	1,6209±0,0065	8,944±0,0601	6,6784±0,0088

With 1000 simulations and P=1000

Case	lambda (pps)	C (mbps)	f (B)	Packet Loss(%)	Average Delay(ms)	Maximum Delay(ms)	TT(Mbps)
A	100	2	100000	0±0	4,545±0,0087	20,8686±0,1715	0,6679±0,0013
B	200	2	100000	0±0	8,1524±0,045	38,1515±0,4326	1,3354±0,0026
C	100	2	10000	0±0	4,5521±0,0089	20,7916±0,1807	0,6688±0,0013
D	200	2	10000	0,0554±0,006	8,0671±0,0406	36,3677±0,2494	1,3324±0,0025
E	500	10	100000	0±0	0,9104±0,0018	4,188±0,0372	3,3437±0,0067
F	1000	10	100000	0±0	1,634±0,0095	7,6627±0,0863	6,6801±0,0133
G	500	10	10000	0±0	0,909±0,0018	4,1793±0,0378	3,3439±0,0069
H	1000	10	10000	0,0561±0,0067	1,6028±0,008	7,1947±0,051	6,6501±0,0127

Simulator 1 Analysis

b) Impact of *lambda*, C and f (with 10 simulations and P=1000)

To analyse the impact of *lambda* we can compare the results in cases A-B, C-D, E-F and G-H. In these cases, we see that when *lambda* increases, so do the Average and Maximum Delays and the Transmitted Throughput, which is expectable, given that more packets arrive every second, meaning there's more data to transmit (increase in TT), and since more data arrives, there are more packets waiting in queue at each instant (increasing Delays). Moreover, it's evident that the increase in these 3 performance parameters is proportional to the increase of *lambda* (when *lambda* doubles, so do the 3 performance parameters mentioned, or at least they close to double).

C also has a very noticeable impact: when it increases the Average and Maximum Delays decrease, and the Transmitted Throughput increases, which is, again, expectable, since we can transmit more data every second (which directly impacts TT, and makes it so packets can be transmitted from the queue faster).

The queue size, f, does not seem to have an impact on the Average and Minimum Delays and the Transmitted Throughput, but a low enough queue size, f, together with a high enough *lambda* and low enough Communication Capacity C, makes the Packet Loss increase.

c) Impact on confidence intervals (with 10 simulations and P=100000)

Comparing the simulation setting from above (with 10 simulations and P=1000) and this one (with 10 simulations and P=100000) we notice that the Packet Loss, Average Delay and Transmitted Throughput confidence values got considerably smaller. This happens because since the stopping criteria is now much bigger, each simulation takes longer, meaning that in each simulation there's more time for the performance parameters to plateau. But on the Maximum Delay, the effect was the opposite, the confidence intervals got much worse. Since the time of each simulation increases, the probability of a packet staying longer in the queue increases, and so there's a higher variation between simulations in the Maximum Delay. It's also worth noting that the values for the Maximum Delay also increased, which is also explained by the fact that now simulations take longer, allowing packets to stay in the queue longer.

d) Impact on confidence intervals (with 1000 simulations and P=1000)

With this setting, we now see an improvement in confidence intervals for every performance parameter, including the Maximum Delay (contrary to the situation in c)). The reason why the confidence intervals no longer got worse for the Maximum Delay is because simulations are not as long as in c) . The confidence intervals improve because of the increase in number of simulations. By running more simulations, we become more and more sure of what the average value for each performance parameter is, which means the confidence interval decreases. This is also easily verifiable in the way the confidence interval is calculated, since it's necessary to divide by the number of simulations.

e) M/M/1 and M/G/1 models

Average Delay theoretical values for the M/M/1 and M/G/1 queueing models

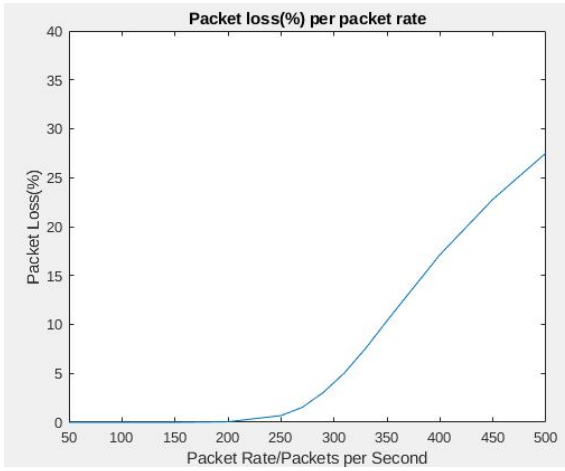
Case	Lambda (pps)	C (Mbps)	Avg. Delay M/M/1 (ms)	Avg. Delay M/G/1 (ms)
A'	100	2	5,0116	4,5449
B'	200	2	10,0465	8,1755
C'	500	10	1,0023	0,9090
D'	1000	10	2,0093	1,63

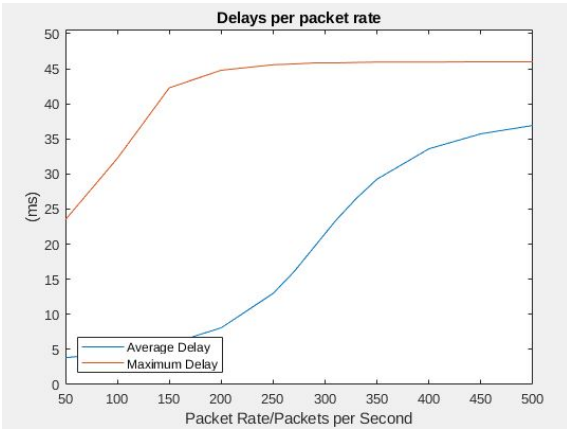
For all cases, the M/G/1 model seems to be a better approximation of the simulation results.

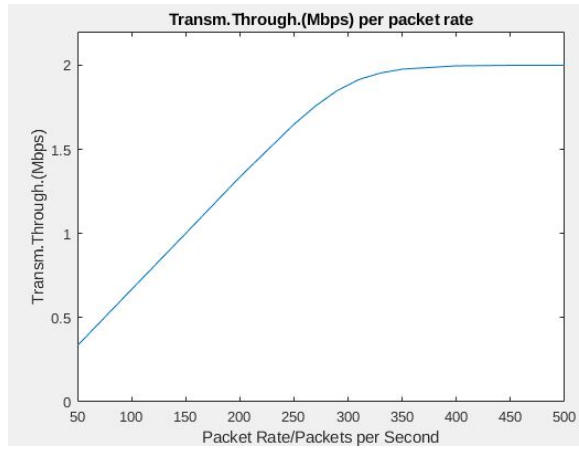
On a more thorough look, if we average the Average Delays from cases A-C and B-D we have an Average Delay of 4.5383 ms with *lambda*=100 and C=2 (A and C) and an Average Delay of 8.14795 ms with *lambda*=200 and C=2 (B and D). Model M/M/1 differs by 0,4733 ms on Case A' and 1,89855 ms on Case B'. Model M/G/1 varies 0,0066 ms and 0,02755 ms for the same cases. Doing the same for the bottom two cases, we get that the M/M/1 model has a total difference (summing the difference from all cases) of 2.8779 ms from the simulations results and M/G/1 model has a 0,04705 ms total difference.

Hence, we conclude that the model that approximates the results best (by a very large margin) is the M/G/1 model. This happens because the service time of each packet depends on the size of the packet, and since packet sizes follow a generic distribution, so does the service time for packets.

f) Plot analysis (C=2 Mbps f=10000)



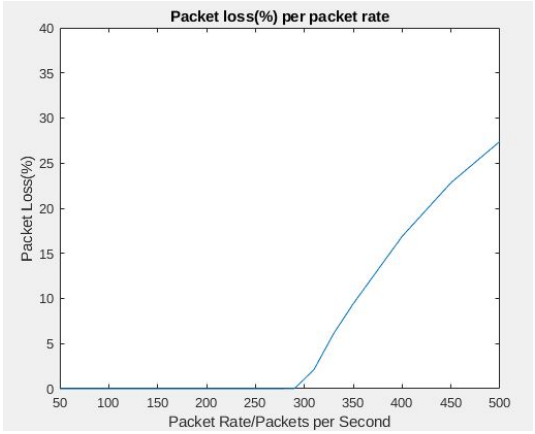


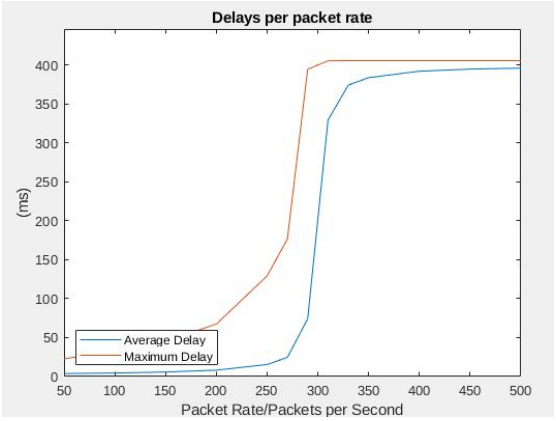


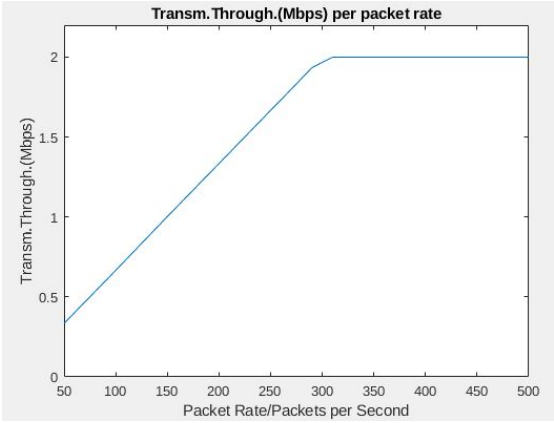
In these first three plots we can see that the delays per packet rate increase with *lambda* until they saturate (at around 45 ms for the max delay, and around 35 for the average delay). Regarding the Maximum Delay, saturating means that the system reached a *lambda* that is able to fill the queue at least once during simulation (we see that the Maximum Delay saturates at around 200 pps). For the Average Delay, saturating means that the queue is full at all times (we notice it saturates at around 450 pps).

As for the Packet Loss it only starts increasing at around 200 pps (the same as when the Maximum Delay stabilizes, this is, the same as when the queue can start to fill, even though not at all times, and so packets can be lost), which allows a 1.25 Mbps of TT (roughly 62.5% of the maximum possible, which is 2 Mbps, the same as the connection capacity). As we go past the 200 pps, the Packet Loss increases at around 5% for each 100 pps. When the packet rate gets to 400 the TT has already reached the maximum possible (2 Mbps).

g) Plot analysis (C=2 Mbps and f=100000)







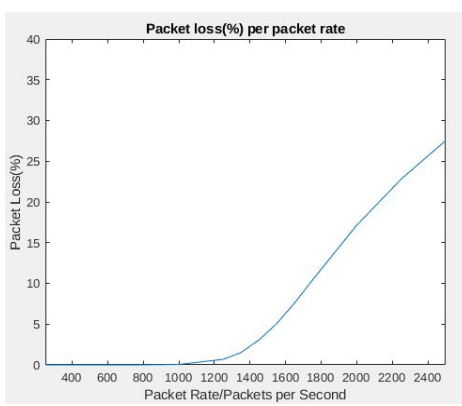
A queue ten times larger made it so that a bigger packet rate was required before starting to lose packets, which was expected: since the queue is bigger, it can hold more packets before it gets full and the system has to discard packets. In this case we see that the

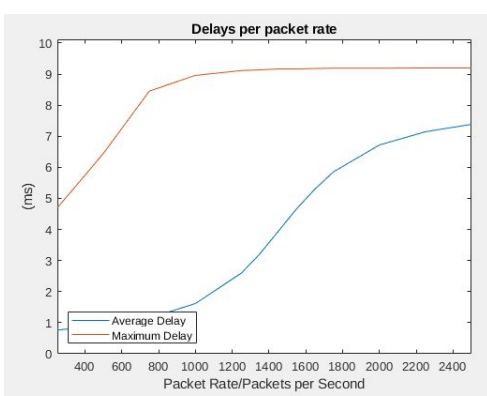
packet loss starts increasing at around 275 pps (again, this value also corresponds to when the Maximum Delay stabilizes), when in the prior situation it started increasing at 200 pps.

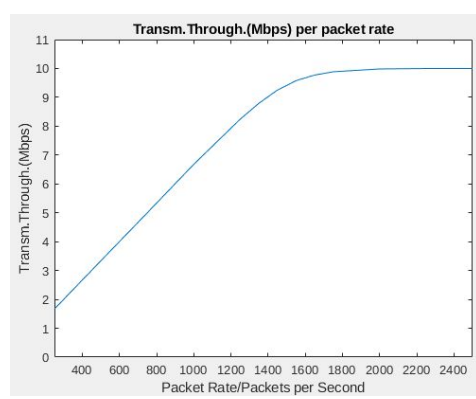
The Maximum Delay only saturates at higher packet rates (275 pps in this case), which is expectable since the queue is larger. In the previous situation a *lambda* of 200 was enough to fill the queue, but now since the queue is bigger even if a group of large packets arrive, they won't be enough to stall so that the queue fills because the queue has more space and this lambda isn't high enough, with our connection capacity, for the system to not be able to transmit the packets faster, or as fast, than they arrive. As for the Average Delay, we notice an interesting result, which is, the packet rate at which it saturated was not very affected. This is because the queue being full at all times (the condition that makes the Average Delay stabilize at its maximum) does not depend on the queue size. Regardless of queue size, as we reach a certain packet rate, our connection capacity can no longer transmit packets as they arrive, and the queue, inevitably, fills up, no matter the size of queue (given that the simulation is run for enough time). Both the Maximum Delay and the Average Delay now saturate at much higher numbers as well: a little over 400 pss for the Maximum Delay and a little under 400 pps for the Average Delay, which is just about 10 times more than before; this is because the queue is 10 times larger, meaning a packet has to wait for 10 times more packets to be transmitted before its time to be transmitted arrives.

Since packets only start to get lost at 275 pps, instead of 200, that increased the maximum throughput without packet losses possible, which now stands at around 1.75 Mbps (87.5% of the maximum, which is superior to the former 62.5%). The graph for the TT is unchanged, since, as seen before, the queue size does not affect the TT.

h) Plot analysis (C=10 Mbps and f=10000)



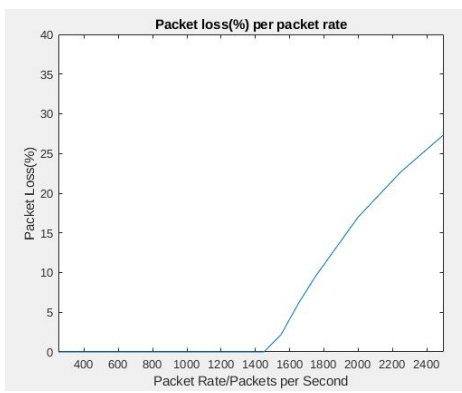


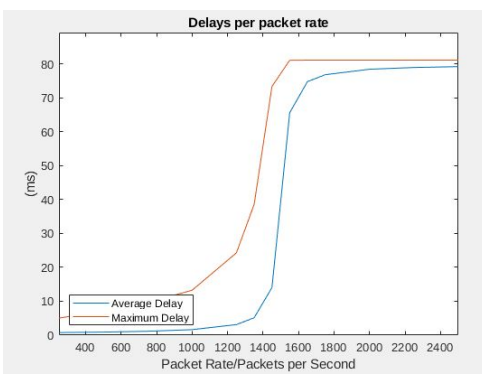


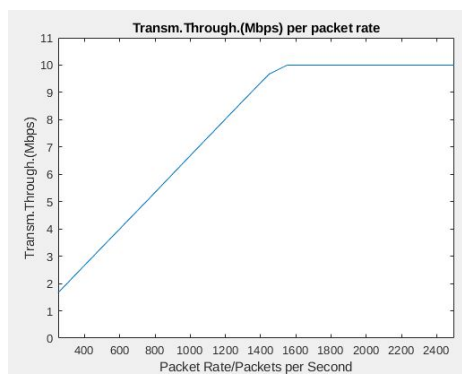
This situation is very similar to the first one (although with different scales, the plots even look the same). The main difference is that both Delays got much smaller, since the packets are processed faster in the queue, because of the higher connection capacity (with a 5 times larger connection capacity, packets are transmitted 5 times faster, and with no surprise the delays got 5 times smaller). That higher connection capacity, means that the maximum TT also gets higher (again, to the same value as the connection capacity).

Packet Loss starts to increase at around 1000 pps, which allows 65% of the maximum throughput possible of 10Mbps, very close to the first case f) , but still smaller than in the second case g), in which we had 87.5% of the maximum TT without Packet Loss. This helps to conclude that even though the queue size does not affect the TT per packet rate, a higher queue size allows to have a higher TT without losing packets, or in other words, makes it so the Packet Loss only starts increasing with higher *lambdas* (like seen in g)).

i) Plot analysis (C=10 Mbps f=100000)







Compared to h), by increasing the queue size we observe similar results that when increasing the queue size with lower capacity (f) to g)). The maximum throughput obtainable before starting to lose packets and without drastically high delays increases greatly, to around 90% of the maximum possible (the maximum being 10 Mbps), which was at around 1400 pps.

The delays also had a similar behavior to the change from f) to g) (the Maximum Delay started to peak at a higher packet rate, but the Average Delay stayed the same and both Delays are around 10 times larger, because of the queue size).

When comparing with g) we can see the same thing seen before: the delay times are around 5 times smaller than the ones in g) (again because the capacity is 5 times larger).

Simulator 2 Results

With 10 simulations and P=100000

Case	Lambda (pps)	C (Mbps)	f (B)	n	Packet Loss Data / VoIP(%)	Avg. Delay Data / VoIP (msec)	Max. Delay Data / VoIP (msec)	Transm. Through. (Mbps)
A	200	2	100000	5	0±0/0±0	11,0675±0,161/ 8,0392±0,1378	82,9867±5,4495/ 81,1144±5,1111	1,5762±0,0035
B	200	2	100000	10	0±0/0±0	22,1968±0,68/ 19,1043±0,6438	148,6677±12,8216/ 147,912±12,8602	1,8202±0,0056
C	200	2	100000	15	2,5697±0,366/ 0,193±0,0282	331,1136±10,404/ 329,622±10,3898	405,1832±0,1979/ 404,519±0,2763	1,9991±0,0004
D	200	2	10000	5	0,243±0,0234/ 0,0189±0,0042	10,3587±0,0982/ 7,4261±0,0938	44,7518±0,276/ 43,9784±0,3531	1,5698±0,0059
E	200	2	10000	10	1,1786±0,0475/ 0,0994±0,0098	15,149±0,1084/ 12,409±0,1085	45,2856±0,1798/ 44,8149±0,2778	1,7936±0,0021
F	200	2	10000	15	5,1956±0,1423/ 0,4399±0,0247	23,2548±0,2339/ 21,147±0,2386	45,3809±0,1264/ 44,9446±0,2057	1,948±0,0026
G	1000	10	100000	25	0±0/0±0	2,2155±0,018/ 1,6219±0,0159	16,7448±1,0969/ 16,3987±1,1378	7,887±0,0182
H	1000	10	100000	50	0±0/0±0	4,1546±0,1379/ 3,5545±0,1361	29,1811±2,6467/ 29,0167±2,6568	9,0662±0,0275
I	1000	10	100000	75	2,1509±0,2035/ 0,1921±0,0268	65,232±1,6536/ 64,9756±1,656	81,0562±0,0329/ 80,9451±0,038	9,9954±0,0025
J	1000	10	10000	25	0,2381±0,0211/ 0,0183±0,0036	2,0795±0,0187/ 1,5027±0,0164	8,9783±0,0377/ 8,8076±0,0985	7,8527±0,0168
K	1000	10	10000	50	1,1652±0,0343/ 0,1035±0,0075	2,9832±0,0238/ 2,4405±0,0217	9,0422±0,0325/ 8,8892±0,0624	8,9263±0,0124
L	1000	10	10000	75	5,2454±0,1279/ 0,4881±0,0248	4,6129±0,0413/ 4,1964±0,0401	9,0959±0,0353/ 9,0409±0,0348	9,7309±0,0074

Simulator 2 Analysis

k) Impact of *lambda*, C, f and n (with 10 simulations and P=100000)

From the results of simulator 2, we notice that the number of VoIP flows n, when increasing, increases the Packet Loss (both for Data and VoIP flows), because, since we have more packets incoming, the queue fills up faster, increases the Average and Maximum Delays (for both Data and VoIP flows), because more frequently, packets have to wait to be transmitted, since the queue fills faster (as said before), and it also increases the Transmitted Throughput, because there are more packets being transmitted.

The size of the queue, f, has no impact on the Transmitted Throughput, but a smaller queue increases the Packet Loss (since it's smaller, less packets can be stored in it and wait to be transmitted) and decreases both the Average and Maximum Delays, because since there are less packets in the queue, each individual packet has to wait less until it is transmitted.

As for *lambda* and C, we see that changing them had no impact on the Packet Loss. In simulation 1, we saw that, in some examples, changing just *lambda* or just C had some impact on the Packet Loss. In this simulation, the fact that when we changed *lambda* and C, but kept the ratio between them (*lambda*/C=100), and by keeping that ratio intact the Packet Loss stayed unchanged, we can say that the Packet Loss is not so much affected by *lambda* or C, but rather the ratio between them. In simulation 1, we can see the impact of changing the ratio: by decreasing the ratio (decreasing *lambda* in simulation 1) the Packet Loss decreases, and by doing the opposite the Packet Loss increases. This makes total sense: if for a given C we have a monstrously high *lamba* (increasing the *lambda*/C ratio) the Packet Loss has to increase. As seen before, increasing C increases the Transmitted Throughput, and so with no surprise the same is verified here. The Delays got smaller, and as seen in prior experiments, increasing C decreases Delays, but increasing *lambda* either increases or has no impact on the Delays (because they've reached their maximums). Since Delays got smaller in the second half of the table, instead of bigger, we can assume that C's impact on all Delays is much more significant than *lambda's*.

Simulator 3 Results

With 10 simulations and P=100000

Case	Lambda (pps)	C (Mbps)	f (B)	n	Packet Loss Data / VoIP(%)	Avg. Delay Data / VoIP (msec)	Max. Delay Data / VoIP (msec)	Transm. Through. (Mbps)
A	200	2	100000	5	0±0/0±0	11,9406±0,1263/ 2,5208±0,0077	90,1751±5,1635/ 8,3386±0,0567	1,5748±0,0034
B	200	2	100000	10	0±0/0±0	25,3423±0,5439/ 3,352±0,0112	163,9425±8,5248/ 9,9469±0,1256	1,8159±0,0036
C	200	2	100000	15	2,4714±0,249/ 0,1827±0,0179	511,326±15,110/ 4,5985±0,0094	629,9123±0,5885/ 11,5787±0,1393	1,9991±0,0003
D	200	2	10000	5	0,286±0,0283/ 0,0189±0,0045	11,3768±0,1411/ 2,5143±0,0088	49,9934±0,1951/ 8,3694±0,0665	1,5698±0,0031
E	200	2	10000	10	1,229±0,0591/ 0,1078±0,0102	18,3968±0,1905/ 3,2803±0,0111	56,718±0,3152/ 9,9798±0,1702	1,7936±0,0026
F	200	2	10000	15	5,1605±0,220/ 0,4087±0,0269	33,4376±0,3964/ 4,316±0,0173	66,0099±0,341/ 11,5847±0,1719	1,9474±0,0029
G	1000	10	100000	25	0±0/0±0	2,398±0,0407/ 0,5202±0,0023	18,749±1,2407/ 1,8484±0,024	7,8834±0,0249
H	1000	10	100000	50	0±0/0±0	5,1633±0,1818/ 0,6931±0,0036	34,1799±2,3295/ 2,2921±0,0378	9,058±0,0302
I	1000	10	100000	75	2,0702±0,267/ 0,1575±0,0188	98,7095±2,2236/ 0,9592±0,0013	126,3602±0,2278/ 2,8158±0,1186	9,9967±0,0018
J	1000	10	10000	25	0,2681±0,023/ 0,0185±0,0036	2,2729±0,0221/ 0,5178±0,0017	10,1782±0,1117/ 1,864±0,0239	7,8507±0,0128
K	1000	10	10000	50	1,1268±0,068/ 0,0978±0,0107	3,6023±0,0433/ 0,6759±0,0017	11,8719±0,1464/ 2,2517±0,0454	8,9309±0,0131
L	1000	10	10000	75	5,1386±0,133/ 0,4382±0,0272	6,5832±0,0714/ 0,8945±0,0036	13,8714±0,1107/ 2,6542±0,0339	9,7269±0,0163

Simulator 3 Analysis

m) Impact of *lambda*, C, f and n (with 10 simulations and P=100000)

From the results of simulator 3, we can see that the number of VoIP flows n, when increasing has similar effects as in simulator 2: Packet Loss (both for Data and VoIP flows) increases, because of the higher amount of packets incoming, the Transmitted Throughput also increases, and so do the Maximum and Average Delays. The difference here, caused by the priority, is that the Average and Maximum Delays for VoIP packets are much smaller, and, consequently, the delays for data packets are higher, than in simulator 2.

The queue size continues to have no impact on the Transmitted Throughput, and a smaller queue still increases the Packet Loss for both data and VoIP packets. The difference here is that a smaller queue no longer makes the delays for VoIP packets smaller (this still happens for data packets). A smaller queue decreases delays because a packet that is in the end of the queue will have to wait for less packets to be transmitted, but since the VoIP packets are always put in ahead of all data packets, a smaller queue no longer decreases delays for VoIP packets (that would only happen if there were enough VoIP packets to fill the queue).

For *lambda* and C we see the same effects as in simulator 2: increasing them, but maintaining the ratio between them, makes it so the Packet Loss stays unchanged too, all Delays decrease and the TT increases.

o) M/G/1 model with 2 priorities

The cases that can be modeled to the M/G/1 model with priorities are all, after removing the queue size from the cases (that leaves us with half the cases, displayed below).

Case	*Lamba* (pps)	C (Mbps)	n	Avg. Data Delay M/G/1 with 2 priorities (ms)	Avg. VoIP Delay M/G/1 with 2 priorities (ms)
A’	200	2	5	12,0965	2,3394
B’	200	2	10	27,0747	2,6709
C’	200	2	15	-92,2300	3,1069
D’	1000	10	25	2,4193	0,4679

E’	1000	10	50	5,4149	0,5342
F’	1000	10	75	-18,4460	0,6254

From the table above, we can see that the M/G/1 model with priorities is not a good fit to the simulation results. For some cases the model does seem to fit with the results with a queue size of 100000, but then the values calculated with the M/G/1 model become far from the real ones on cases C’ and F’. In these cases, the sum of ρ of all classes exceeds one so the model calculates negative average delay times, which is obviously impossible to happen in practice, garenteing us that the model is not a good approximation of the simulation results. Since ρ is the amount of bits received per second divided by the amount of bits the system is able to transmit each second, we know it surpasses one once the amount of bits received each second become larger than C. With several classes the sum of all ρ’s is the sum of all bits received per second divided by C. The problem here, and the reason why this sum becomes larger than one is that the VoIP *lambda* depends directly on the amount of VoIP flows,n, (and the amount of bits received per second depends on both the data and VoIP *lambdas*) and with a high n, there are more bits arriving at the system each second than the connection can transmit in a second, and so this becomes impossible to model with the M/G/1 with priorities.

Simulator 4 Results

Case	Lambda (pps)	C (Mbps)	n	r (%)	Packet Loss Data / VoIP(%)	Avg. Delay Data / VoIP (msec)	Max. Delay Data / VoIP (msec	Transm.Through. (Mbps)
A	200	2	5	25	9,0618±0,0886/ 0±0	5,8404±0,0131/ 2,1605±0,0052	17,4868±0,134/ 8,2636±0,0762	1,3884±0,0027
B	200	2	10	25	12,26±0,0904/ 0±0	6,5079±0,0232/ 2,5805±0,0099	19,3252±0,2268/ 9,8795±0,1504	1,567±0,0034
C	200	2	15	25	16,6405±0,1501 /0±0	7,3527±0,0331/ 3,156±0,011	21,7411±0,2848/ 11,0008±0,111	1,7253±0,0021
D	1000	10	25	25	9,1125±0,1032/ 0±0	1,1697±0,003/ 0,4452±0,001	3,6744±0,0332/ 1,8027±0,0266	6,9475±0,0122
E	1000	10	50	25	12,2771±0,1203 /0±0	1,304±0,0039/ 0,5342±0,0017	4,2214±0,0987/ 2,132±0,0409	7,8134±0,0135
F	1000	10	75	25	16,8711±0,1792 /0±0	1,4703±0,007/ 0,6591±0,0026	4,7217±0,1141/ 2,5086±0,046	8,6004±0,0166
G	200	2	5	50	2,3674±0,0433/	8,7889±0,0416/	28,3544±0,1212/	1,5256±0,0023

					0±0	2,4235±0,0055	8,333±0,0548	
H	200	2	10	50	4,7388±0,0997/ 0±0	11,3875±0,0522 /3,0605±0,0104	32,1912±0,2384/ 9,9625±0,1472	1,7225±0,0027
I	200	2	15	50	9,078±0,1307/ 0±0	15,2382±0,0956 /3,9108±0,0146	37,5979±0,4566/ 11,5121±0,2079	1,8693±0,0029
J	1000	10	25	50	2,4452±0,0731/ 0±0	1,7674±0,0066/ 0,5007±0,0011	5,9115±0,0645/ 1,8693±0,0238	7,6363±0,0111
K	1000	10	50	50	4,8379±0,092/ 0±0	2,2591±0,0093/ 0,6287±0,0019	6,9372±0,1216/ 2,2539±0,0571	8,5634±0,0148
L	1000	10	75	50	9,3486±0,2384/ 0±0	3,0523±0,0231/ 0,8168±0,0033	8,2548±0,1775/ 2,6827±0,0992	9,3466±0,0124
M	200	2	5	75	0,7533±0,0518/ 0±0	10,4408±0,0924 /2,4896±0,0074	39,7522±0,1915/ 8,3703±0,0751	1,5584±0,0035
N	200	2	10	75	2,271±0,0703/ 0±0	15,2416±0,1198 /3,2±0,0107	44,6737±0,1942/ 9,8615±0,1197	1,7665±0,0033
O	200	2	15	75	6,504±0,2759/ 0±0	23,8258±0,3479 /4,1655±0,025	52,5418±0,3138/ 11,3442±0,0785	1,9177±0,0048
P	1000	10	25	75	0,7564±0,0469/ 0±0	2,0888±0,0093/ 0,5134±0,0014	8,0471±0,0581/ 1,8881±0,0278	7,7885±0,016
Q	1000	10	50	75	2,3775±0,0791/ 0±0	3,0701±0,014/ 0,6642±0,0019	9,4989±0,15/ 2,2578±0,0349	8,8391±0,0101
R	1000	10	75	75	6,7939±0,2017/ 0±0	4,7684±0,0488/ 0,8733±0,0032	11,3156±0,1234/ 2,7164±0,0528	9,599±0,0128

Simulator 4 Analysis

q) Impact of *lambda*, C, n and r (with 10 simulations and P=100000)

Since for the simulator 4 the priority for VoIP still exists, the impact of the number of VoIP flows, n, is similar to the one in simulator 3: Packet Loss and Transmitted Throughput increase, because of the higher amount of packets arriving, and so do the Delays for both flows. As in simulator 3, the priority makes for very low VoIP delays.

The new parameter r, completely removes the Packet Loss for VoIP packets. Since 100 - r is the space in the queue that is for VoIP use only, for the cases simulated here, there's always space in the queue so that no VoIP packets have to be dropped. With very small r, this means that there's very little space in queue to store data packets, and so with small r's we see

high Packet Loss for the data packets. As r increases, the Packet Loss for the data packets starts to become lower and getting closer to the one in simulation 3 (simulation 3 can be considered as a simulation 4 with r=100%). Regarding the Average and Maximum Delays, smaller r's mean that the Delays are smaller, because on average data packets are larger than VoIP ones (which means it's faster to transmit VoIP packets than data ones), and so, a smaller amount of data packets in queue means a given packet reaches the front of the queue faster. R also affects the Transmission Throughput: a higher r increases the TT, because with higher values for r, we discard less data packets, meaning there's more Bytes to transfer every second. This happens because the VoIP packets can't fill their "dedicated" part of the queue for any of the r values used in simulation, and so, some space of the queue is never used. Approaching as r of 100%, the VoIP packets get smaller and smaller "dedicated" part of the queue (which becomes easier and easier to fill up) and the TT goes up to the values of simulation 3.

Again, *lambda* and C have the same impacts as in simulator 2.

The Code

The simulator 2:

```
function [PL , PL_VOIP, APD , APD_VOIP, MPD , MPD_VOIP, TT] = simulator2(lambda,C,f,n,P)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
% OUTPUT PARAMETERS:
%  PL   - packet loss (%)
%  APD  - average packet delay (milliseconds)
%  MPD  - maximum packet delay (milliseconds)
%  TT   - transmitted throughput (Mbps)

%Events:
ARRIVAL= 0;        % Arrival of a packet
DEPARTURE= 1;      % Departure of a packet
ARRIVAL_VOIP= 2;   % Arrival of VoIP packet
DEPARTURE_VOIP= 3; % Departure of VoIP packet

%State variables:
State = 0;          % 0 - connection free; 1 - connection busy
QueueOccupation= 0; % Occupation of the queue (in Bytes)
Queue= [];          % Size and arriving time instant of each packet in the queue

%Statistical Counters:
TotalPackets= 0;       % No. of packets arrived to the system
LostPackets= 0;        % No. of packets dropped due to buffer overflow
TotalPackets_VOIP= 0;       % No. of VoIP packets arrived to the system
LostPackets_VOIP= 0;        % No. of VoIP packets dropped due to buffer overflow
TransmittedPackets= 0; % No. of transmitted packets
TransmittedPackets_VOIP= 0; % No. of VoIP transmitted packets
TransmittedBytes= 0;   % Sum of the Bytes of transmitted packets
Delays= 0;             % Sum of the delays of transmitted packets
MaxDelay= 0;           % Maximum delay among all transmitted packets
Delays_VOIP= 0;             % Sum of the delays of VoIP transmitted packets
MaxDelay_VOIP= 0;           % Maximum delay among all VoIP transmitted packets

%Auxiliary variables:
% Initializing the simulation clock:
Clock= 0;

% Initializing the List of Events with the first ARRIVAL:
EventList = [ARRIVAL , Clock + exprnd(1/lambda) , GeneratePacketSize() , 0];
%n vezes
for i=1:n
    EventList = [EventList; ARRIVAL_VOIP , Clock + unifrnd(0,0.02) , randi([110,130]) , 0];
end

%Similation loop:
while TransmittedPackets+TransmittedPackets_VOIP<P    % Stopping criterium
    EventList= sortrows(EventList,2);    % Order EentList by time
    Event= EventList(1,1);               % Get first event and
    Clock= EventList(1,2);               %   and
    PacketSize= EventList(1,3);          %   associated
    ArrivalInstant= EventList(1,4);      %   parameters.
    EventList(1,:)= [];                  % Eliminate first event
    switch Event
        case ARRIVAL                     % If first event is an ARRIVAL
            TotalPackets= TotalPackets+1;
```

```
        case ARRIVAL                        % If first event is an ARRIVAL
            TotalPackets= TotalPackets+1;
            EventList = [EventList; ARRIVAL , Clock + exprnd(1/lambda) , GeneratePacketSize() , 0];
            if State==0
                State= 1;
                EventList = [EventList; DEPARTURE , Clock + 8*PacketSize/(C*10^6) , PacketSize , Clock];
            else
                if QueueOccupation + PacketSize <= f
                    Queue= [Queue;PacketSize , Clock, DEPARTURE];
                    QueueOccupation= QueueOccupation + PacketSize;
                else
                    LostPackets= LostPackets + 1;
                end
            end
        case ARRIVAL_VOIP
            TotalPackets_VOIP= TotalPackets_VOIP+1;
            EventList = [EventList; ARRIVAL_VOIP , Clock + unifrnd(0.016,0.024) , randi([110,130]) , 0];
            if State==0
                State= 1;
                EventList = [EventList; DEPARTURE_VOIP , Clock + 8*PacketSize/(C*10^6) , PacketSize , Clock];
            else
                if QueueOccupation + PacketSize <= f
                    Queue= [Queue;PacketSize , Clock, DEPARTURE_VOIP];
                    QueueOccupation= QueueOccupation + PacketSize;
                else
                    LostPackets_VOIP= LostPackets_VOIP + 1;
                end
            end
        case DEPARTURE                      % If first event is a DEPARTURE
            TransmittedBytes= TransmittedBytes + PacketSize;
            Delays= Delays + (Clock - ArrivalInstant);
            if Clock - ArrivalInstant > MaxDelay
                MaxDelay= Clock - ArrivalInstant;
            end
            TransmittedPackets= TransmittedPackets + 1;
            if QueueOccupation > 0
                EventList = [EventList; Queue(1,3) , Clock + 8*Queue(1,1)/(C*10^6) , Queue(1,1) , Queue(1,2)];
                QueueOccupation= QueueOccupation - Queue(1,1);
                Queue(1,:)= [];
            else
                State= 0;
            end
        case DEPARTURE_VOIP
            TransmittedBytes= TransmittedBytes + PacketSize;
            Delays_VOIP= Delays_VOIP + (Clock - ArrivalInstant);
            if Clock - ArrivalInstant > MaxDelay_VOIP
                MaxDelay_VOIP= Clock - ArrivalInstant;
            end
            TransmittedPackets_VOIP= TransmittedPackets_VOIP + 1;
            if QueueOccupation > 0
                EventList = [EventList; Queue(1,3) , Clock + 8*Queue(1,1)/(C*10^6) , Queue(1,1) , Queue(1,2)];
                QueueOccupation= QueueOccupation - Queue(1,1);
                Queue(1,:)= [];
            else
                State= 0;
            end
    end
end
```

```
    end
end

%Performance parameters determination:
PL= 100*LostPackets/TotalPackets;      % in %
PL_VOIP=  100*LostPackets_VOIP/TotalPackets_VOIP;      % in %
APD= 1000*Delays/TransmittedPackets;   % in milliseconds
MPD= 1000*MaxDelay;                    % in milliseconds
APD_VOIP= 1000*Delays_VOIP/TransmittedPackets_VOIP;   % in milliseconds
MPD_VOIP= 1000*MaxDelay_VOIP;                    % in milliseconds
TT= 10^(-6)*TransmittedBytes*8/Clock;  % in Mbps

end

function out= GeneratePacketSize()
    aux= rand();
    if aux <= 0.16
        out= 64;
    elseif aux >= 0.78
        out= 1518;
    else
        out = randi([65 1517]);
    end
end
```

The difference from simulator 2 to simulator 3 (in the Departures for data packets):

```
if QueueOccupation > 0
    Queue = sortrows(Queue, 3, 'descend');
    EventList = [EventList; Queue(1,3) , Clock + 8*Queue(1,1)/(C*10^6) , Queue(1,1) , Queue(1,2)];
    QueueOccupation= QueueOccupation - Queue(1,1);
    Queue(1,:)= [];
```

The difference from simulator 4 to simulator 3 (in the Arrivals for data packets):

```
if QueueOccupation + PacketSize <= f*r && QueueOccupation + PacketSize <= f
    Queue= [Queue;PacketSize , Clock, DEPARTURE];
    QueueOccupation= QueueOccupation + PacketSize;
else
    LostPackets= LostPackets + 1;
end
```

The function to calculate the M/M/1 average delay theoretical values:

```
function [avg_delay] = avg_delay_mm1(lambda, C)

B = 0.16*64 + 0.22*1518 + (1-0.22-0.16)*((65+1517)/2);
b = 8*B;

mu = C / b;

avg_delay = 1e3 * 1./(mu-lambda);

end
```

And the one to calculate the same values for the M/G/1 model:

```
function [avg_delay] = avg_delay_mg1(lambda, C)

B = 0.16*64 + 0.22*1518 + (1-0.22-0.16)*((65+1517)/2);
b = 8*B;

mu = C / b;

ES = 1./mu;
ES2 = 8^2*(0.16*64^2 + 0.22*1518^2) ./ C.^2;

%+ (1-0.22-0.16)*((65+1517)/2)^2
p = (1-0.22-0.16)/length(65:1517);

for i=65:1517
    ES2 = ES2 + 8^2*(p*i^2)./ C.^2;
end
avg_delay = ( lambda.*ES2 ./ (2*(1-lambda.*ES))  + ES )* 1e3;

end
```

The function to calculate the M/G/1 average delay theoretical values with 2 priorities:

```
function [avg_delay_data, avg_delay_voip] = avg_delay_mg1_priorities(lambda_data, lambda_voip, C)
%data
B_data = 0.16*64 + 0.22*1518 + (1-0.22-0.16)*((65+1517)/2); %average size of a packet
b_data = 8*B_data; % in bits

mu_data = C / b_data;
ES_data = 1./mu_data;

ES2_data = 8^2*(0.16*64^2 + 0.22*1518^2) ./ C.^2;

p = (1-0.22-0.16)/length(65:1517);

for i=65:1517
    ES2_data = ES2_data + 8^2*(p*i^2)./ C.^2;
end

%voip
B_voip = (110+130)/2; %average size of a packet
b_voip = 8*B_voip; % in bits

mu_voip = C/ b_voip;
ES_voip = 1./mu_voip;

p = 1/length(110:130);
ES2_voip = zeros(1,length(C));
for j=110:130
    ES2_voip = ES2_voip + 8^2*(p*j^2)./C.^2;
end

%queue avg delay
ro_data = lambda_data./mu_data;
ro_voip = lambda_voip./mu_voip;
avg_delay_queue_data = ( lambda_data.*ES2_data + lambda_voip.*ES2_voip ) ./ ( 2*(1- ro_voip).*(1- ro_voip - ro_data) );
avg_delay_queue_voip = ( lambda_data.*ES2_data + lambda_voip.*ES2_voip ) ./ ( 2*(1-ro_voip) );

%avg delays
avg_delay_data = (ES_data + avg_delay_queue_data) * 1e3;

avg_delay_voip = (ES_voip + avg_delay_queue_voip) * 1e3;

end
```