**Part 1:**

The ProjectCode1.py has been modified to include a --buffer\_size flag to specify the size of the finite buffer and the --time\_distribution flag to specify the distribution for inter-arrival time and service time. For both questions in Part 1, I used a simulation time of 1000000 iterations.

**Q1:**

I experimented with arrival rates of [0.1, 0.2, 0.4, 0.8], buffer sizes of [5, 10, 20, 40] given in packets, a mu (μ) of 2 packets per second, and Poisson distribution for inter-arrival time and service time with the following commands:

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=5 --time\_distribution=poisson

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=10 --time\_distribution=poisson

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=20 --time\_distribution=poisson

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=40 --time\_distribution=poisson

The results are presented in the tables below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Arrival Rate (λ) | Buffer Size (packets) | Total Number of Packets | Number of Dropped Packets | Utilization |
| 0.100 | 5 | 100516 | 33 | 0.201 |
| 0.200 | 5 | 200424 | 1255 | 0.398 |
| 0.400 | 5 | 400591 | 35942 | 0.729 |
| 0.800 | 5 | 799281 | 318417 | 0.962 |
| 0.100 | 10 | 100144 | 1 | 0.200 |
| 0.200 | 10 | 200479 | 25 | 0.401 |
| 0.400 | 10 | 401140 | 9722 | 0.784 |
| 0.800 | 10 | 799523 | 301425 | 0.997 |
| 0.100 | 20 | 100144 | 1 | 0.200 |
| 0.200 | 20 | 199805 | 3 | 0.398 |
| 0.400 | 20 | 400967 | 993 | 0.801 |
| 0.800 | 20 | 799964 | 300425 | 1.000 |
| 0.100 | 40 | 100144 | 1 | 0.200 |
| 0.200 | 40 | 199805 | 3 | 0.398 |
| 0.400 | 40 | 400672 | 5 | 0.801 |
| 0.800 | 40 | 799486 | 299468 | 1.000 |

**Discussion:**

As I increased the arrival rate (λ), I observed increases in total number of packets, number of dropped packets, and utilization.

When buffer size was 5, I observed a sublinear increase in utilization, e.g. when I doubled the arrival rate the utilization increased less than a factor of 2. However, as the buffer size increased from 5 to 10, 20, or 40, I observed a roughly linear increase in utilization.

I observed an exponential increase in number of dropped packets as I increased the arrival rate across all buffer sizes, whereas the total number of packets increased in a roughly-uniform manner.

The loss probability of packets increased superlinearly as the arrival rate increased across all buffer sizes, starting from a minimum of 0.001% at λ=0.100 to a maximum of 37.701% at λ=0.800. Moreover, I observed that number of dropped packets and the loss probability was substantially higher for mid-level arrival rates (e.g. λ ∈ {0.200, 0.400}) when buffer size was lower. Overall, we can conclude that buffer size is inversely proportional to the loss probability and arrival rate is proportional to the loss probability.

**Q2:**

I experimented with arrival rates of [0.1, 0.2, 0.4, 0.8, 0.9, 0.99], buffer sizes of [5, 10, 20, 40], a mu (μ) of 2 packets per second, and deterministic (constant) inter-arrival time and service time with the following commands:

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=5 --time\_distribution=constant

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=10 --time\_distribution=constant

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=20 --time\_distribution=constant

python ProjectCode1.py --arrival\_rates=0.1,0.2,0.4,0.8 --buffer\_size=40 --time\_distribution=constant

The results are presented in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Arrival Rate (λ) | Buffer Size (packets) | Total Number of Packets | Number of Dropped Packets | Utilization |
| 0.100 | 5 | 99999 | 0 | 0.200 |
| 0.200 | 5 | 199999 | 0 | 0.400 |
| 0.400 | 5 | 399999 | 0 | 0.800 |
| 0.800 | 5 | 799999 | 30000 | 1.000 |
| 0.100 | 10 | 99999 | 0 | 0.200 |
| 0.200 | 10 | 199999 | 0 | 0.400 |
| 0.400 | 10 | 399999 | 0 | 0.800 |
| 0.800 | 10 | 799999 | 30000 | 1.000 |
| 0.100 | 20 | 99999 | 0 | 0.200 |
| 0.200 | 20 | 199999 | 0 | 0.400 |
| 0.400 | 20 | 399999 | 0 | 0.800 |
| 0.800 | 20 | 799999 | 30000 | 1.000 |
| 0.100 | 40 | 99999 | 0 | 0.200 |
| 0.200 | 40 | 199999 | 0 | 0.400 |
| 0.400 | 40 | 399999 | 0 | 0.800 |
| 0.800 | 40 | 799999 | 30000 | 1.000 |

**Discussion:**

When the inter-arrival time and service time is made deterministic (constant) instead of Poisson distribution, we observe that the utilization has a perfectly linear, proportional relationship with the arrival rate independent of the buffer size. Furthermore, for the values we have experimented with, we observe that buffer size no longer has any impact on the number of dropped packets and the loss probability of packets: whereas arrival rates of 0.100, 0.200, and 0.400 yield 0 dropped packets and thus 0% loss, an arrival rate of 0.800 yields 30000 dropped packets and thus 3.75% loss probability. This is much lower compared to the probabilities reported in Q1, but nevertheless we still observe an proportional relationship between arrival rate and number of dropped packets and loss probability.

**Part 2**

The ProjectCode2.py has been modified to make the simulation look like the topology given in Figure 1 and to match the initial assumptions and parameters given in the project specification.

**Q1:**

I experimented with buffer sizes of [100, 250, 500, 1000, 10000, 120000] given in bytes, port rates of [100, 500, 1000, 2000] given in packets per second, and simulation time of 4000 iterations with the following commands:

python ProjectCode2.py --port\_rate=100 --buffer\_size=100

python ProjectCode2.py --port\_rate=100 --buffer\_size=250

python ProjectCode2.py --port\_rate=100 --buffer\_size=500

python ProjectCode2.py --port\_rate=100 --buffer\_size=1000

python ProjectCode2.py --port\_rate=100 --buffer\_size=10000

python ProjectCode2.py --port\_rate=100 --buffer\_size=120000

python ProjectCode2.py --port\_rate=500 --buffer\_size=100

python ProjectCode2.py --port\_rate=500 --buffer\_size=250

python ProjectCode2.py --port\_rate=500 --buffer\_size=500

python ProjectCode2.py --port\_rate=500 --buffer\_size=1000

python ProjectCode2.py --port\_rate=500 --buffer\_size=10000

python ProjectCode2.py --port\_rate=500 --buffer\_size=120000

python ProjectCode2.py --port\_rate=1000 --buffer\_size=100

python ProjectCode2.py --port\_rate=1000 --buffer\_size=250

python ProjectCode2.py --port\_rate=1000 --buffer\_size=500

python ProjectCode2.py --port\_rate=1000 --buffer\_size=1000

python ProjectCode2.py --port\_rate=1000 --buffer\_size=10000

python ProjectCode2.py --port\_rate=1000 --buffer\_size=120000

python ProjectCode2.py --port\_rate=2000 --buffer\_size=100

python ProjectCode2.py --port\_rate=2000 --buffer\_size=250

python ProjectCode2.py --port\_rate=2000 --buffer\_size=500

python ProjectCode2.py --port\_rate=2000 --buffer\_size=1000

python ProjectCode2.py --port\_rate=2000 --buffer\_size=10000

python ProjectCode2.py --port\_rate=2000 --buffer\_size=120000

The results are presented in the table below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Port Rate | Buffer Size (bytes) | Average Wait Source 1 to Output 3 | Average Wait Source 2 to Output 4 | Packets Sent | Packets Received | Packets Dropped |
| 100 | 100 | 0.01234 | 0.00422 | 6090 | 3383 | 2707 |
| 100 | 250 | 0.02332 | 0.00759 | 6157 | 4976 | 1181 |
| 100 | 500 | 0.02873 | 0.00958 | 6012 | 5357 | 655 |
| 100 | 1000 | 0.03110 | 0.01025 | 6122 | 5441 | 681 |
| 100 | 10000 | 0.03110 | 0.01025 | 6122 | 5441 | 681 |
| 100 | 120000 | 0.03110 | 0.01025 | 6122 | 5441 | 681 |
| 500 | 100 | 0.00251 | 0.00085 | 5944 | 3479 | 2465 |
| 500 | 250 | 0.00460 | 0.00151 | 6048 | 4872 | 1176 |
| 500 | 500 | 0.00585 | 0.00194 | 6084 | 5338 | 746 |
| 500 | 1000 | 0.00608 | 0.00198 | 6034 | 5423 | 611 |
| 500 | 10000 | 0.00608 | 0.00198 | 6034 | 5423 | 611 |
| 500 | 120000 | 0.00608 | 0.00198 | 6034 | 5423 | 611 |
| 1000 | 100 | 0.00121 | 0.00043 | 6039 | 3430 | 2609 |
| 1000 | 250 | 0.00231 | 0.00078 | 6024 | 4956 | 1068 |
| 1000 | 500 | 0.00291 | 0.00094 | 5983 | 5348 | 635 |
| 1000 | 1000 | 0.00298 | 0.00096 | 6152 | 5387 | 765 |
| 1000 | 10000 | 0.00298 | 0.00096 | 6152 | 5387 | 765 |
| 1000 | 120000 | 0.00298 | 0.00096 | 6152 | 5387 | 765 |
| 2000 | 100 | 0.00061 | 0.00021 | 6055 | 3395 | 2660 |
| 2000 | 250 | 0.00113 | 0.00038 | 6112 | 4933 | 1179 |
| 2000 | 500 | 0.00147 | 0.00049 | 6038 | 5405 | 633 |
| 2000 | 1000 | 0.00150 | 0.00050 | 6116 | 5362 | 754 |
| 2000 | 10000 | 0.00150 | 0.00050 | 6116 | 5363 | 753 |
| 2000 | 120000 | 0.00150 | 0.00050 | 6116 | 5363 | 753 |

**Discussion:**

Buffer size, given in bytes as opposed to in packets as in the previous part, was proportional to both recorded average waits (i.e. one from source 1 to output 3 and one from source 2 to output 4); increasing the buffer size across all port rates increased the average wait duration. Across all port rates, we saw a saturation in average wait times after a buffer size of 1000. Port rate was inversely proportional to both average waits: a higher port rate corresponded with a lower average wait.

Buffer size was proportional to the number of packets received and inversely proportional to the number of packets dropped: a higher buffer size corresponded more packets received, less packets dropped, and hence a lower loss probability of packets. On the other hand, I observed that port rate did not have a visible impact on packets received, packets dropped, and hence the loss probability of packets.