Question 1:

Write a Python function that receives λ as one of its arguments and that returns a single random sample drawn from an exponential random distribution. Using this function, write a short piece of Python code to generate 1000 exponential random variables with $\lambda=75$. What is the mean and variance of the 1000 random variables you generated? Do they agree with the expected value and the variance of an exponential random variable with $\lambda=75$? Your values will not be exact. Why not? Note: if you do not know what you are expecting to see, you will need to look it up.

Response:

The mean is 0.013439668966118699 and the variance is 0.0001883437130246113. The expected mean and variance of an exponential variable with $\lambda = 75$ is 0.013333333333 and 0.00017777777 respectively. I agree with the values received from my Python function because there is only around a 5% error at most between the theoretical values and simulated values that I calculated.

Question 2:

Build your simulator for an M/M/1. Explain in words how you compute the performance metrics E[n] and P_IDLE. Do not show code.

Response:

After generating arrival, departure, and observation events for my simulation using exponential random variables, I sorted them all to find the performance metrics desired. I thought of a few tricks to determine E[N] and P_{IDLE} using five variables. Three of them were defined as suggested in the lab 1 document, while two of them I took the liberty of defining. At any given observation event, I want to find the number of packets in the queue. By finding the difference between the number of arrivals and number of departures up until that event, I know how many arrivals are still waiting to be serviced. If the difference is 0 that means that the queue is empty as every packet that has arrived has been serviced which means the queue is idle. To find E[N], I just averaged the sum of the number of packets that have been in the queue at each observation event over the total number of observation events recorded. P_{IDLE} can be calculated by averaging how many times the queue has been idle when an observation event occurs over the number of observation events.

Question 3:

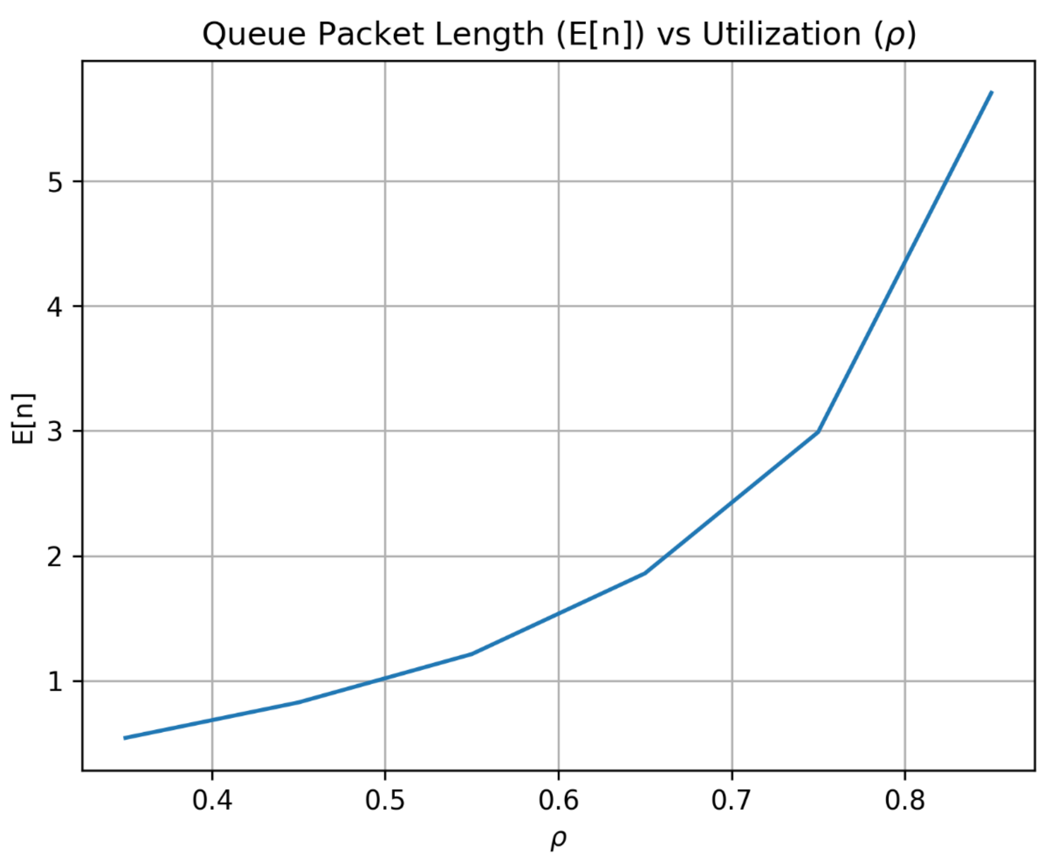
The packet length will follow an exponential distribution with an average of L = 2000 bits. Assume that C = 1Mbps. Use your simulator to obtain the following graphs. Plot them on separate figures.

1. E[N], the average number of packets in the queue as a function of ρ (for $0.25 < \rho < 0.95$, step size 0.1).
2. P_{IDLE}, the proportion of time the system is idle as a function of ρ (for $0.25 < \rho < 0.95$, step size 0.1).

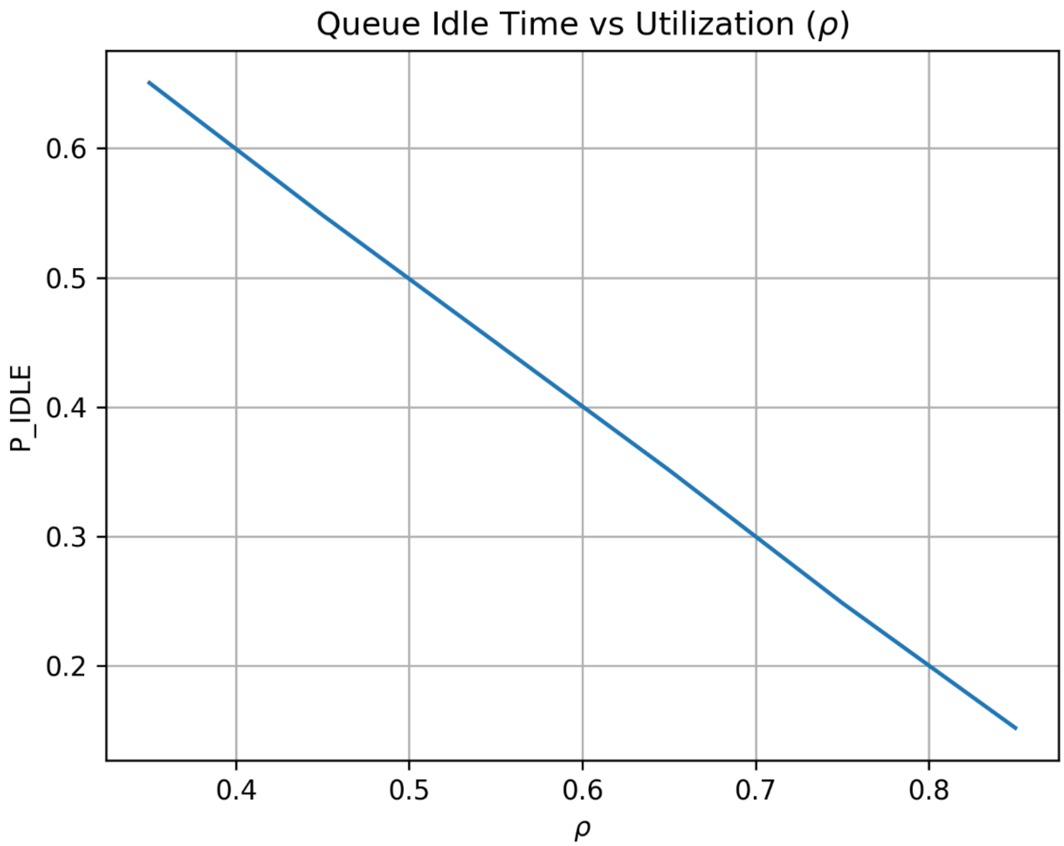
Explain the trends in both graphs.

Response:

Graph 1:



Graph 2:



For the graph of Queue Packet Length vs Utilization (Graph 1), there is a positive correlation between the queue size (number of packets in queue) and ρ. This makes sense because we calculate the arrival rate (arrival lambda) as ρ * C / L, so as ρ increases, the rate of packet arrival will increase, and since the service times are nondependent on rho, these will stay the same which means arrivals will occur at a higher rate compared to departures as rho increases. On the other hand, there is negative correlation between ρ and PIDLE, as seen in Graph 2. This can be explained by the fact that if the average number of packets in the queue increases as ρ increases, then that means the queue on average should have a non-zero number of packets more often, resulting in a decrease in queue idle time. In short, the trends from the graphs are supported mathematically by the effect that rho has on arrival rates and how those effects are absent in the calculations of service times.

Question 4:

For the same parameters, simulate for $\rho = 1.2$.What do you observe? Explain.

Response:

When utilization is set to 1.2, I found that the average number of packets in the queue (E[n]) increased greatly, and the idle time decreased to almost zero. This makes sense because the formula for lambda is proportional to ρ, so when ρ is greater than 1, the rate of arrival is much larger. In turn, with a larger rate of arrival, the service times (which are not also increasing proportionally to ρ) result in many departure events that start to get backed up as more events are generated by my simulator. Since packets start to back up, the average idle time of the queue is minimal as there will almost always be packets waiting in the queue with the increase in utilization.

Question 5:

Build a simulator for an M/M/1/K queue and briefly explain how you implemented packet dropping. Explain in words how you computed P_LOSS. Do not show code.

Response

I decided to write my M/M/1/K simulator based on my original simulator for the M/M/1 queue by allowing for a queue size to be provided. Since now departure events cannot be generated beforehand, a packet being dropped means there should not be a departure time associated. The list of events, now treated as a heapq object, is popped one by one and then managed depending on if the queue size. When the queue isn't full, an arrival event gets generated a departure event, which allows for only packets that are *not* dropped to be assigned their respective departure events. On the other hand, if the queue is full, then the arrival event is ignored to signify a packet drop. Every dropped arrival event was kept track of using a counter, which was then used to find the probability of packet loss by dividing by the total number of arrival attempts.

Question 6:

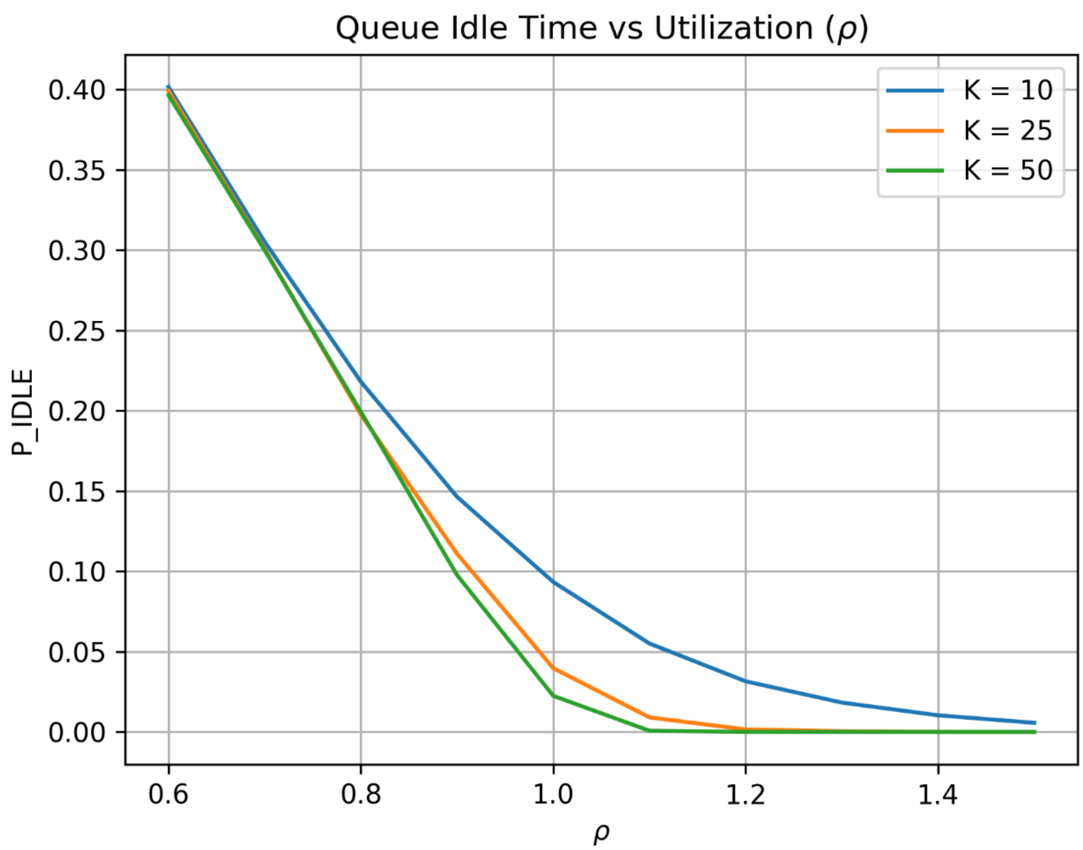
Let L=2000 bits and C=1 Mbps. Use your simulator to obtain the following graphs:

- P_{IDLE} as a function of ρ (for 0.5 < ρ < 1.5, step size 0.1), for K = 10, 25, 50 packets. Show one curve for each value of K on the same graph.
- P_{LOSS} as a function of ρ (for 0.5 < ρ < 1.5) for K = 10, 25, 50 packets. Show one curve for each value of K on the same graph.

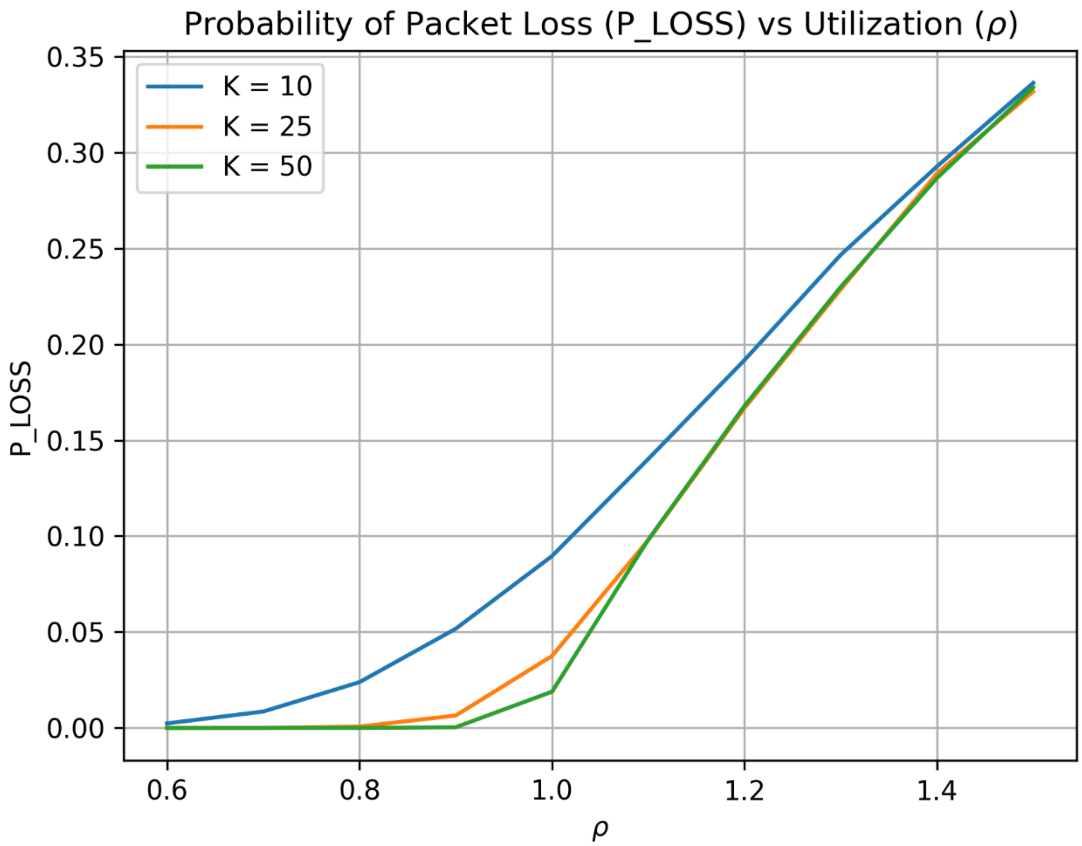
For which value of K does P_{IDLE} reach zero soonest? Why is this the case?

Response:

Graph 1:



Graph 2:



From the graphs, we see that K = 50 results in P_{IDLE} reaching zero the quickest. When looking at the graph of P_{LOSS} vs ρ (Graph 2), we can see that for larger values of K, the probability of packet loss is noticeably lower until utilization reaches 1.5. If there is a lower chance of packet loss, then there should in turn be a higher number of packets being serviced when the queue size (in this case K) is bigger. It's also interesting to note that Graph 1 looks like a reflection on the Y axis of Graph 2. In conclusion, I believe that due to more packets being serviced due to a lower probability of a packet being dropped, queues with larger capacities will reach a value of $P_{IDLE} = 0$ the quickest.