ECE 358: Computer Networks
Project 1: M/D/1 and M/D/1/K Queue Simulation

Date of Submission: October 14, 2015

Submitted By:

Student ID: Malesevic, Djordje: dmalesev@uwaterloo.ca

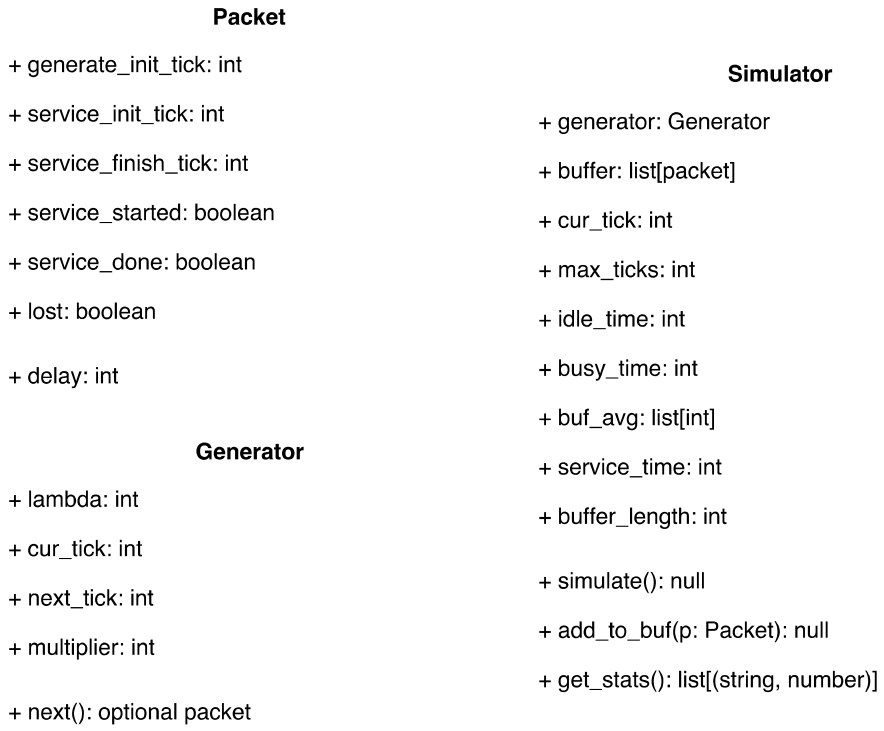
Student ID: Hkeradman, Hormoz: hkheradm@uwaterloo.ca

Marks received:

Marked by:

Table of Contents

1. Our code was designed to keep the important parts of the simulator separate from each other. As such, we created three classes as outlined in the UML diagram below:



In summary, an instance of the Simulator class would be responsible for orchestrating the simulation and most importantly hold the state of the simulation (namely, the number of ticks elapsed, and etc.). The Simulator class contains an instance of the Generator class.

A Generator is responsible for deciding if a packet should be generated at the current tick. Every time its next() function is called, it increments its ticker and if enough ticks have passed since its creation, it will return a new packet, otherwise it will return nothing; both cases are handled in the Simulator instance.

A class called Packet was created to keep statistics that need to be contained in a single packet. The most important metric is the sojourn time which is calculated by the “delay()” function for each packet.

In the main function of our program, user arguments are parsed and based on the number of data points requested, the correct number of Simulator instances are created and added to a list of simulators. Next, every Simulator’s “simulate()” function is called, followed by the construction of a list of each Simulator’s statistics which then get averaged and displayed to the user.

The details of how the Simulator instance orchestrates the simulation will be explained next.
On initialization of the Simulator object, arguments by the user are set to correct fields for later use. In addition, a Generator instance is created local to each instance of a Simulator, allowing for parallelism if required later. Current tick of the simulator is set to 0. An empty queue is created to simulate the packet queue, a global list of packets is created so we can gather statistics later, variables to keep track

of idle time and busy time are also created.

When the “simulate()” function of the Simulator is called, a ticker loop starts and it lasts for the length of the maximum number of ticks requested. In each iteration of the loop, we call the generator instance, which optionally returns a packet. If a packet was returned by the generator, the packet is passed to the “add_to_buf(p: Packet)” function which either appends the packet to the queue or reject it if the queue is full. After the potential package is taken care of, in the same tick, we process the first packet in the queue, if any. The service is done by checking if the servicing has started. If it has, we check if the service time has elapsed since the service starting time, in which case the package is marked done. If the service has not started, we set the beginning-of-service tick variable in the Packet object to the current tick and its final tick to “service_time + current_tick”. In the general case where the queue was empty, we simply do no servicing and increment only the idle time.

Numerous statistics are returned by the “get_stats()” function. Time elapsed is calculated as “cur_tick/tick_length”, packets generated is the number of packets in the global packet array/list. Packets lost are all those packets in the array that have their “lost” field set to True. Similar calculations are done for packets serviced. Average packet delay is calculated by taking the average of the “delay()” for all packets that have their “service_done” field set to true. Average number of packets in queue is calculated by appending the length of the queue at each tick in the Simulator to a list and then averaging that list.

2.

Average number of packets in the buffer/queue E[N] : 0.223
Average sojourn time (queuing delay + service time) E[T] : 2.17 ms
Proportion of the time idle Pidle
Total time = 10 s
Idle time = 8 s
Busy time = 2 s
Pidle = 80%

3.

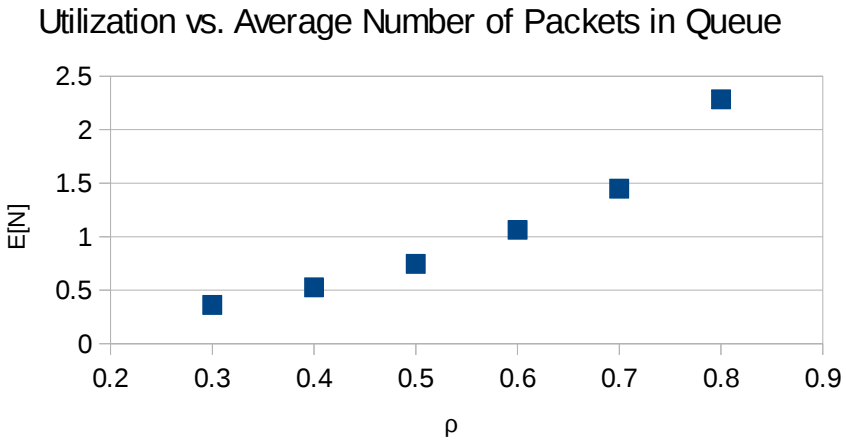


Figure 1: Utilization vs. Average Number of Packets in Queue

These results are expected because with a queue that is being utilized we should expect more packets in the queue on average. More bits per service time means that there should be more traffic and a larger build up in the buffer which is what we have observed.

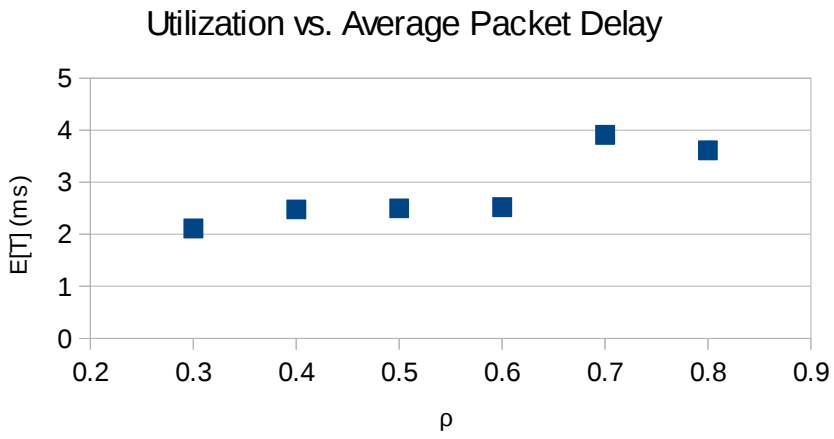


Figure 2: Utilization vs. Average Packet Delay

As we increase the utilization of the queue we observe an increase in the the average delay of a packet (sojourn time). This is expected because having a busier queues means that on average a packet must wait longer to be processed which contributes to the average sojourn time. The correlation is not as strong due to the factor of randomness.

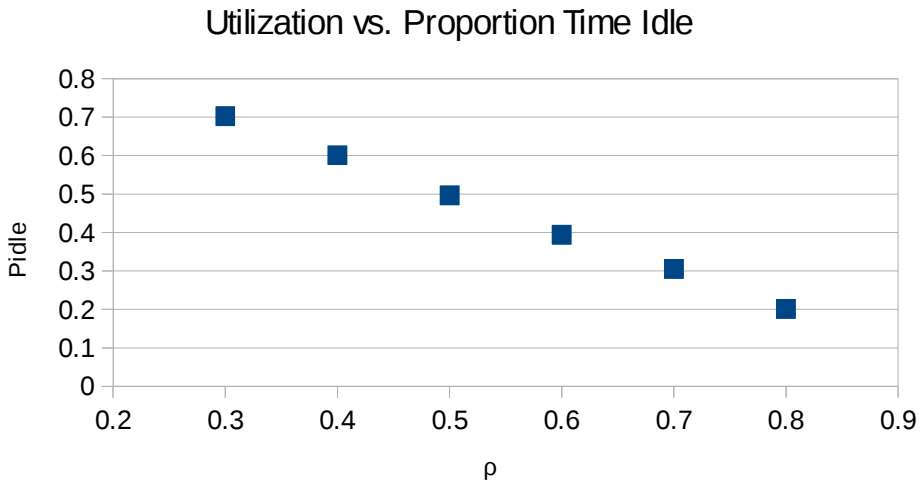


Figure 3: Utilization vs. Proportion Time Idle

A more utilized queue, one where the bits to service time is higher, would consistently have more packets being processed. This is observed in our data as the proportion of time spent idle is negatively

correlated to the utilization.

4.

The M/D/1/K queue is handled within the same class as our M/D/1 queue, meaning they both use the same simulator. The difference is that if given an inputted argument for a buffer size, a M/D/1/K queue is initialized, else a M/D/1 queue is used. This is enforced with a function that when a packet should be entered into the queue, it first checks to see if the length of the queue is less than the size of it, else the packet is lost.

5.

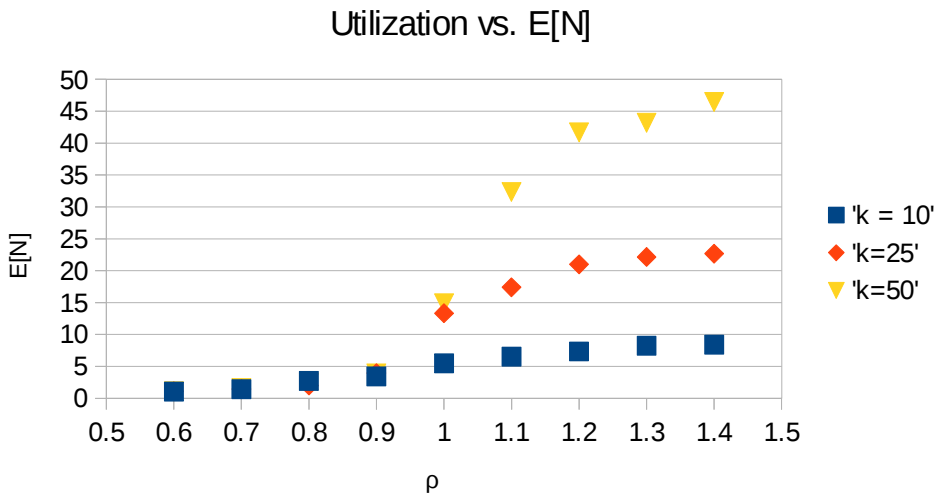


Figure 4: Utilization vs. E[N]

In increase in activity or utilization results in an increase in average number of packets in the queue. Understandably, given a larger queue there is a larger amount of possible packets to be stored in the queue.

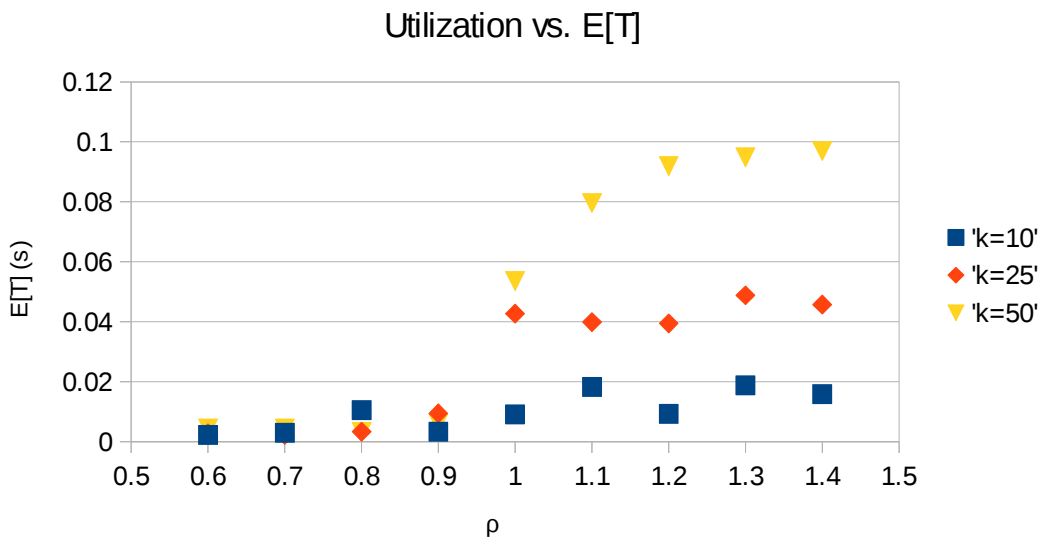


Figure 5: Utilization vs. E[T]

An increase in utilization means more traffic and longer sojourn times. With an increasing buffer size or 'k', the maximum queue delay increases making the average E[T] increasingly larger.

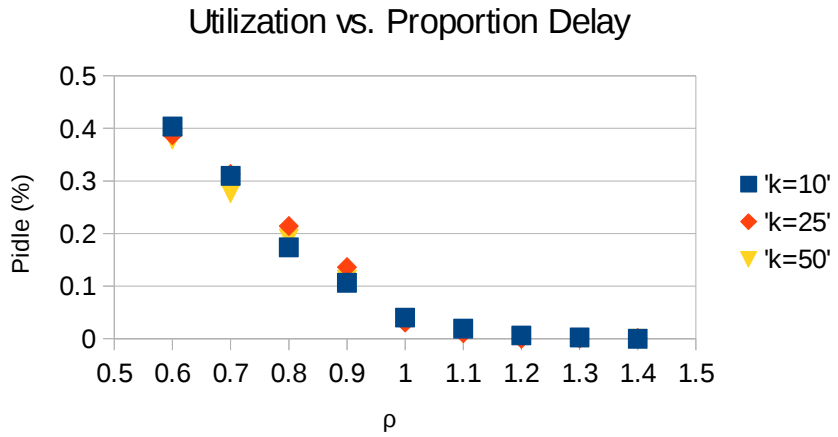


Figure 6: Utilization vs. Proportion Delay

With an increase in bits/service time the network is busier and there is less idle time. This is shown in the graph and is a relationship expressed regardless of buffer size.

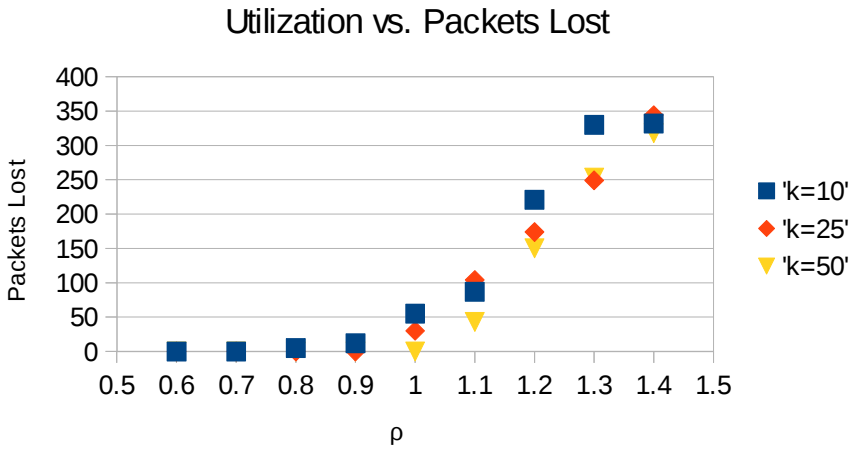


Figure 7: Utilization vs. Packets Lost

An increase in utilization means that there is more traffic and queue that fills more rapidly since it is not able to service it quick enough. A full buffer will then begin to lose packets, however, a smaller sized buffer will begin to lose packets quicker due to its inability to keep up with the arrival of new packets. This is seen in the figure above.