CS359 - Computer Network Lab

Lab 2

Tarusi Mittal 1901CS65

Architecture:

EVENTS CLASS

1. eid: it the stage ID where our packet is currently at
2. pid: It the packet id of our packet
3. currentTime: This time is the time when our packet has reached the stage with that eid

This class tells us about the various events that take place in our process:

EVENT0 = generation of packet
EVENT1 = reaching Queue time
EVENT2 = leaving queue time
EVENT3 = reaching sink time

SOURCE INFO CLASS

1. rate: It is the rate of generation of the packets by the source
2. sourceID: It is the unique source Id given to every source
3. bandSoSwi: It is the bandwidth speed between the Source and The Switch

This class contains the information about the sources i.e. We are creating the sources by this class. The rate is for questions for part 3 and 4

PACKET INFO CLASS

1. packetId: It is the unique Id of every packet
2. queueIn: The time the packet gets inside the queue
3. queueOut: The time the packet gets out of the queue
4. sourceID: The source Id of the source from where the packet was generated
5. generationTime: The time the packet was generated

This class contains the information about the packets whoch are created by sources and and going to the sink

i.e it creates the packets

SWITCH INFO CLASS

1. bandSwSin: The bandwidth between the Switch and the Sink
2. qSize: The size of the queue present in the Switch

This class handles what is going inside the switch like the queue generation and the rate at which the packets are transferred to the sink

EXPLANATION:

First we will start by seeing what is poisson distribution. It is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant mean rate and independently of the time since the last event. The poisson distribution function is given by the following formula:

$$f(k;\lambda) = \Pr(X{=}k) = \frac{\lambda^k e^{-\lambda}}{k!},$$ here k is the no of occurances

And λ is equal to the expected value of X and it is also equal to its variance. The equation can also be modified for rate where $\lambda = rt$, r=rate and t=time interval in which k events are happening.

For our purpose what we have taken from the properties of poisson distribution is:

1. The average rate at which events occur is independent of any occurrences
2. The occurrence of one event does not affect the probability that a second event will occur. That is, events occur independently.

Now coming to our coding part:

When we start our program the user is given the option either to use the default values for the variables such as simulation tine, no of sources, bandwidth etc. or the user can input the values. Once this decision is made our actual process starts.

After that we have defined a nextTime function. This function finds the next random time when the packet generation is taking place from the source to keep up with the Poisson Distribution

```
def nextTime(rateVariable):

    return -math.log(1.0 - random.uniform(0,1)) / rateVariable
```

here rateVariable is = lambda/no of sources. And for a single source its simply the rate

This function is written by taking reference from the following source: https://preshing.com/20111007/how-to-generate-random-timings-for-a-poisson-process/

Event 0:

Now our event 0 start where our packet generation takes place from the sources through the packetInfo after a certain time which depends upon our nextTime Function which again depends upon λ. All the packets that are generated are sorted into the priority queue with event id 0 according to the time that they are generated. From here to keep our process running we take one element out of our queue (with the most priority i.e., which was generated the earliest) and keep updating their event in the queue itself. Along with it we update our packet array which contains all the information about our packet with its Time. Simultaneously after every nextTime(rate) time later a new packet is generated from the source.

Event 0 -> Event 1 -> Event 2:

From here our packet enters the sink and simultaneously the event id is now changed to 1.

Now after this we have two conditions in our question:

1. When our Queue size is infinite: In this case the packets simply go from event 1 to the event 2 without experiencing any drop in their number. This is for our first 3 parts when we donot have any restriction on our queue size.

2. When our Queue size is finite: In this case, the packets can have the tendency to drop which will depend upon our lambda since we are keeping all the other factors constant. In this case we will calculate the no of packets dropped and the no of packets that actually were arriving. These calculations will take place in our Event 2 in the Switch

Event 2 -> Event 3:

In this all our packets simply leaves from the switch and goes inside our sink. Here we count the total no of packets reaching the sink as it was required to calculate average in our first 3 parts.

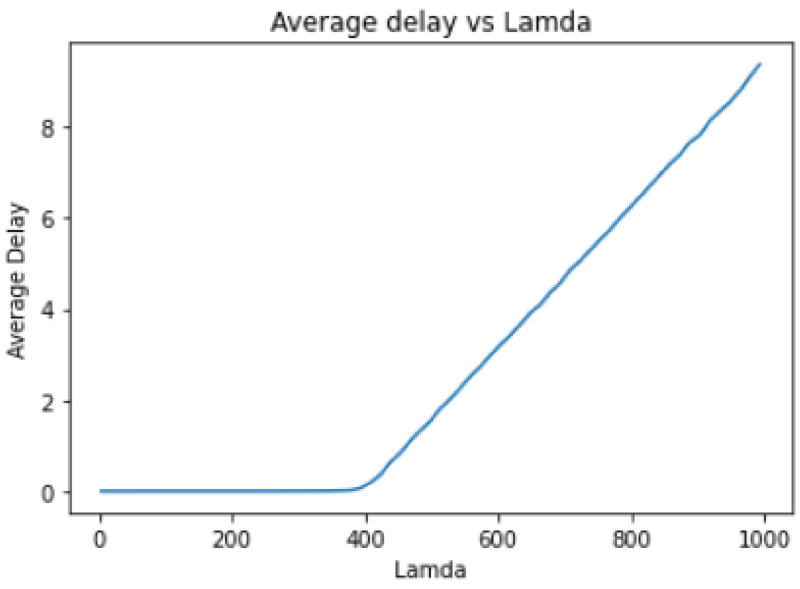
Basically in all our parts we are deciding our packet generation rate through lambda and then varying of the nature whether we need to find the average delay, average queueing size or packet loss we use that.

To ensure that we are not over randomising we have iterated every single process for a given lambda multiple number of times

<u>QUESTIONS:</u>

In this assignment rather than using a constant packet generation rate, we will use a Poisson Distribution and observe its effects. We now assume that the packet generation at each source follows a Poisson Distribution with a given rate λi which is equivalent to the fact that the generation time between two consecutive packets at each source follows an exponential distribution

1. Assuming λi to be same for each source, plot the average delay for each packet with respect to λ.



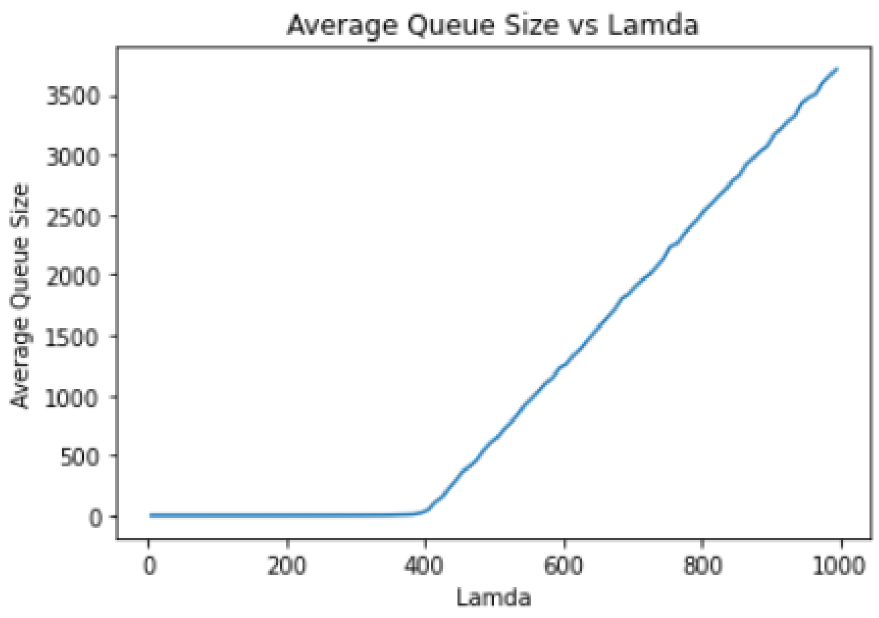
For this part in our section of **<u>Event 0 -> Event 1 -> Event 2</u>** the queue which is holding the packets inside the switch is infinite. So when every source generates its packet with **λi** the queue start filling up. And the rate by which they are going out Is constant which is mentioned in our code by bandSwSin. Now we can see that as the value of λ will increase this implies that the rateVariable in our next Time function will increase and hence that function will produce time closer to each other leading to production of more packets in a Time Period than earlier. Overall time between two packets being generated will reduce and hence the average delay will eventually increase since the queue clearing process gets slower and packets are now taking more time to reach the sink.

As we can see from the graph also initially when lambda was small the delay was almost negligible because the rate of going out from the queue was almost similar to the (rate of going in * the packet length). But with time as lambda starts increasing the queue started taking more time to clear which resulted in the packets taking more time to reach the sink.

The code implementation, output and graphs for different values for this part can be seen at:

https://colab.research.google.com/drive/19Q-XB6e9M2O6Om1VV5zsoP5S1mlcoUN-?usp=sharing

2. Assuming λi to be same for each source, plot the average queue size with respect to λ.



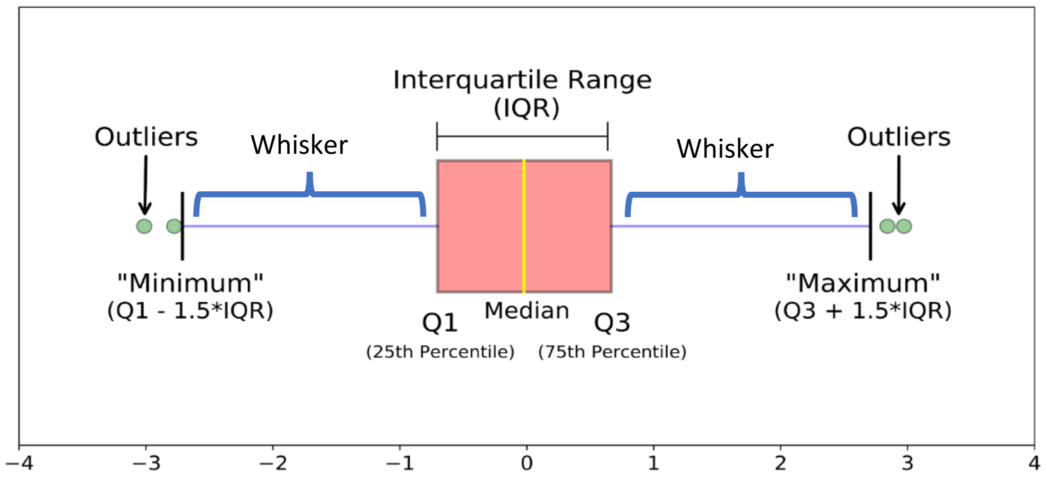
For this part in our section of **<u>Event 0 -> Event 1 -> Event 2</u>** the queue which is holding the packets inside the switch is infinite. So when every source generates its packet with **λi** the queue start filling up. And the rate by which they are going out Is constant which is mentioned in our code by bandSwSin. Now we can see that as the value of λ will increase this implies that the rateVariable in our next Time function will increase and hence that function will produce time closer to each other leading to production of more packets in a Time Period than earlier. Overall time between two packets being generated will reduce the queue clearing process gets slower which will result in increasing the queue size eventually.

As we can see from the graph also initially when lambda was small the queue size was almost negligible because the rate of going out from the queue was almost similar to the (rate of going in * the packet length). But with time as lambda starts increasing the queue started taking more time to clear and hence the queue size kept increasing.

The code implementation, output and graphs for different values for this part can be seen at:

https://colab.research.google.com/drive/1kXtNYKUGdtL00k_DlPXFG7pTIMS3jk6D?usp=sharing

For the next parts we need to plot a box graph so we will first see what exactly a box graph is:



<u>How do we make a Box-Whisker Plot:</u>

For making this graph we need to

1. Arrange the values in their ascending order and then we find the median of them and place it in on the number line.

2. Then we take the median of lower and upper quartile and place them on the number line and bound both of them with a box. This makes our interquartile range.(IQR).The difference between Q3 and Q1 is known as IQR basically

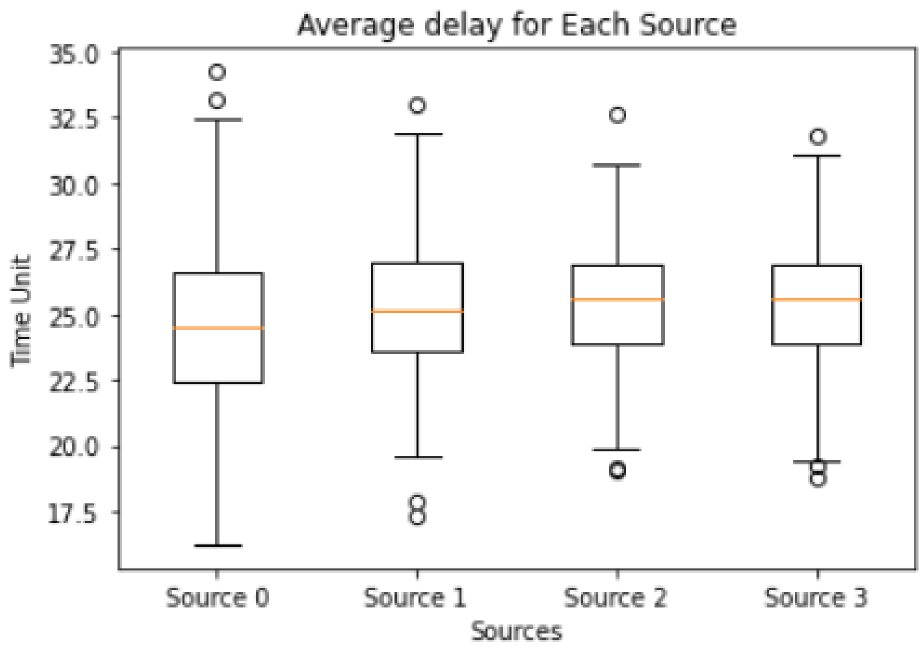
3. Then we calculate the minimum and maximum point using the formula given in the diagram above.

4. After that all the points in the range outside of Minimum and Maximum are called Outliers.

WHY BOX PLOT?

Box plots are useful as they provide a visual summary of the data enabling researchers to quickly identify mean values, the dispersion of the data set, and signs of skewness

3. Assuming unique λi values for each source, using a box plot show the average delay for each packet for each source

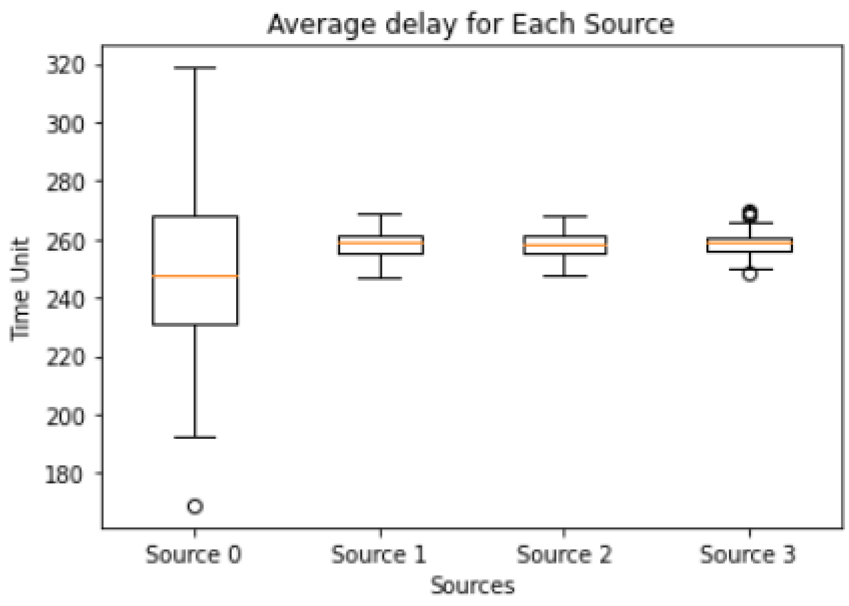


This graphs shows us the Box Plot between the average delay for each packet for each source. As all the other factors were constant the difference for different nature of graphs for the 4 sources is due to their difference of lambdas.

The values of lambdas for each source were: [0.1,0.5,4,16] respectively.

For very small lambda the median is almost somewhat in the between of lower and upper quartiles median.

As we keep on increasing the lambda the median tends to move towards the higher values suggesting a skewness.



This graph is when our lambda value are:[0.1,4,16,25].

Hence it is clear that if we increase the lambda our distribution starts getting concentrated to a little area.

The code implementation, output and graphs for different values for this part can be seen at:

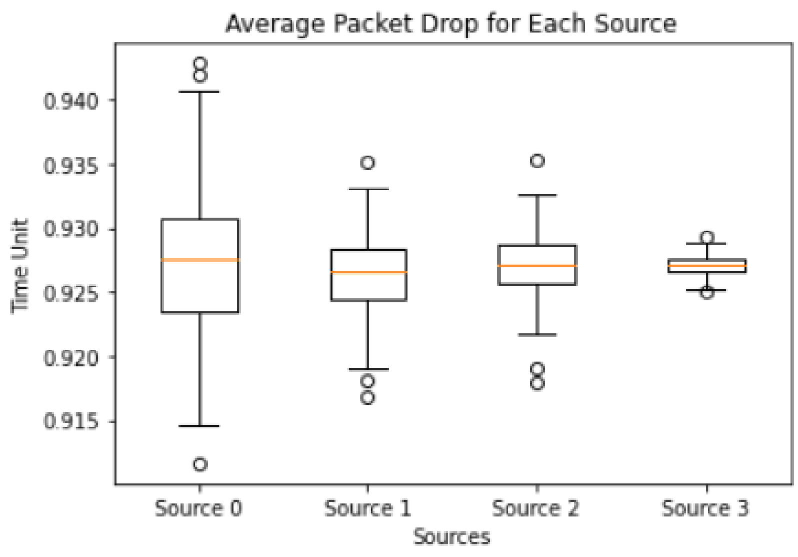
https://colab.research.google.com/drive/1yw_ZzDZIjfFG4LkKi5WFoo0NgSrhJKNv?usp=sharing

4. Assuming unique λi values for each source, using a box plot show the average packet drop for each source

For the graph below the values of Lambda were [0.9,2 ,4.5, 16].

We can see that as the value of lambda increases our packet loss rate sort of get concentrated to a small region and the total loss increases with the increase in lambda.

The skewness with respect to lambda is however not very prominent.



The code implementation, output and graphs for different values for this part can be seen at:

https://colab.research.google.com/drive/1d9UjyzrLSixlKXzl22dzdx3YQo3t7TPW?usp=sharing

END