**Computer Network Lab**

**CS359**

Tanishq Malu Lab:2 1901CS63

Architecture:

**Creates packets**

**Creates sources**

**Switch Class**

**Packet Class**

**Source Class**

**Switch\_Bs =Transmission speed in bits**

**SINK**

Queue

**SourceA**

**SourceB**

**Bs =Transmission speed in bits**

**SourceC**

**SourceD**

**Stage3: packet reaches sink**

**Stage2: packet leaving queue**

**Stage1: packet reaches queue**

**Stage0: packet generation**

**Stage Class**

Classes used:

**Packet Class**

**Attributes:**

**Pkt\_Id** = unique id of a packet

**queue\_reach\_time** = Time when packet reaches queue in switch

**queue\_leave\_time** = Time when packet leave the queue

**ReachSinkTime** = Time when packet reaches sink

**sourceId** = Id of source from which the packet was generated

**GenerationTime** = Time at which this packet was generated

**SourceClass**

**Attributes:**

**SourceID =** unique id of a source

**bs =** bandwidth speed between source and switch

**generation\_rate =** Rate of generation of packets by the source (Required in question 3 and 4 )

**Stage Class**

**Attributes:**

**StageId** = Id of the stage at which a packet is

**packetid** = Id of the packet

**currenttime** = Time at which this packet reached this stage

**Switch Class**

**Attributes:**

**QueueSize** = size of the queue in switch

**switch\_bs** = bandwidth speed between switch and sink

# The basic purpose of the stage class is to break down the whole process of

# packet transfer from source to sink via switch into smaller process.

Breaking down the whole process of packet transfer from source to sink via a switch to simulate the process.

# Let us break down the process into 4 stages

# Stage id = 0 => The time when packet is generated

# Stage id = 1 => The time when packet reaches the queue

# Stage id = 2 => The time when packet leaves the queue

# Stage id = 3 => The time when packet reaches the sink

General working with constant packet generation rate:

We set the values of different variables for the program either by default or take them as input from user.

After formation of all the source object using source class, the program initializes the generation of first packet for each source using the packet class. These packet, as is evident, are in the stage 0 and *event object* are created for each packet with current time and entered into a priority queue, which sorts the *event objects* based on their time when they were generated. (Initially, we have kept a minute time difference for formation of first packet of each source)

Now we keep popping the top element of our priority queue and pushing new event object formed along the process to simulate the process as per the stage and packet id of the *event object* mentioned in the respective elements: -

When a Packet leaves a source and heads towards the switch, our program changes the stage id of the *event object* associated with this packet, from stage 0 to stage 1 indicating this transfer of packet from source to switch. Simultaneously, a new packet will be generated from the source after (current time + 1/generation rate) time and an *event object* will be created for this packet with stage id 0 and pushed into the priority queue.

When the packet reaches the queue, if the queue size is infinite, there is no case of packet drop and it will simply transit towards the head of the queue. Here the stage id of the *event object* associated with this packet will be changed from stage 1 to stage 2 and the packet is now ready to move towards the sink. (Pushed back into the priority queue)

On the other hand, if the size of the queue is finite, there will be chances of packet drop too and separate counter will be incremented for this purpose.

The event object containing packets with stage id 2 will now move from queue to sink and its stage id will change from stage 2 to stage 3 and a new event object will be pushed back into the priority queue.

Finally, the packet will reach the sink.

We can keep account of time at which the packet was generated and packet reached the sink. Also, we can maintain a count for total number of packets reached to our sink and can use it to find average delay. Further a count of packet drop and packet arrived can be maintained to find packet drop rate.

This whole process is simulated with varying bandwidth speed between source and sink, keeping other things constant to find the relationship between utilization factor and average delay and similarly for utilization factor and packet drop rate too.

Packet generation with Poisson distribution:

 If *X* has the Poisson distribution with mean *λ*, the where, l is the mean rate of arrivals and t is a period of time. Conversely, if the number of events per unit time follows a Poisson distribution, then the amount of time between events follows the exponential distribution.

Defining T as the time of an event (a random variable), we have (by definition),

F(t) = P [T t] --------- Equation a

This is equal to,

F(t) = P [T t] = 1 - P [T > t] --------- Equation b

where,

P [ T > t] = P [ zero events occur in time 0 to t] = P [X = 0] = =

Now, we plug in our result for P [T > t] into our Equation b and we get,

F(t) = 1- --------- Equation c

which is the distribution function for the exponential distribution. Further, the expected time until the next event is the same no matter how long since the last

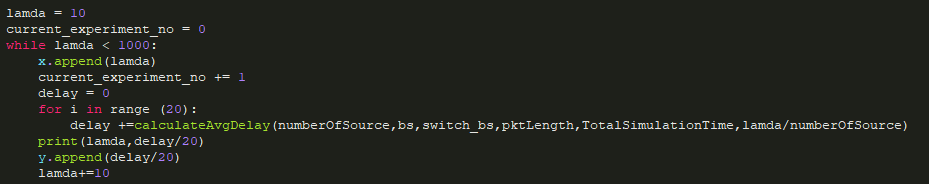
event occurred, which is termed as Memory less property of exponential distribution.

Functions used in code to generate random next time to simulate the Poisson distribution:



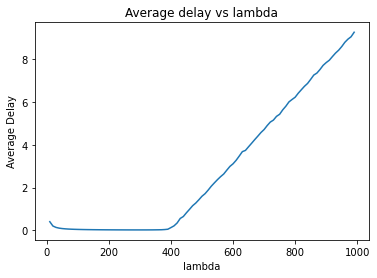


Further, for a given lambda, we will iterate a process multiple time to avoid excessive randomness by our random function and this overall helps us find better results:

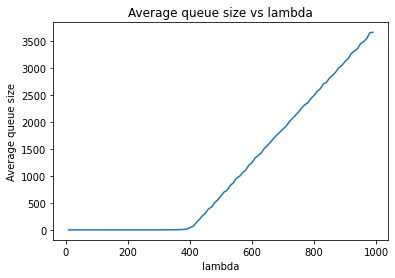


1. Assuming to be same for each source, plot the average delay for each packet with respect to

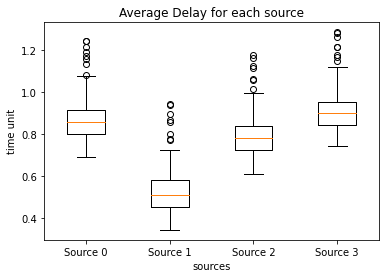
Inference: As we can observe, as lambda increases, the average delay also increases. It is quite evident rom the fact that if Poisson packet generation rate increases, the rate of generation of packets in each time interval also increases.



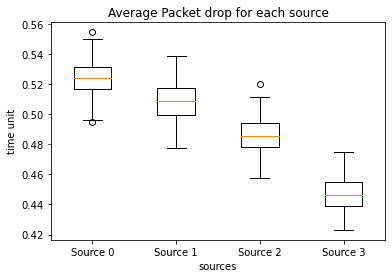
1. Assuming to be same for each source, plot the average queue size with respect to .



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



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



-------------------------------------------End Of Assignment---------------------------------------------