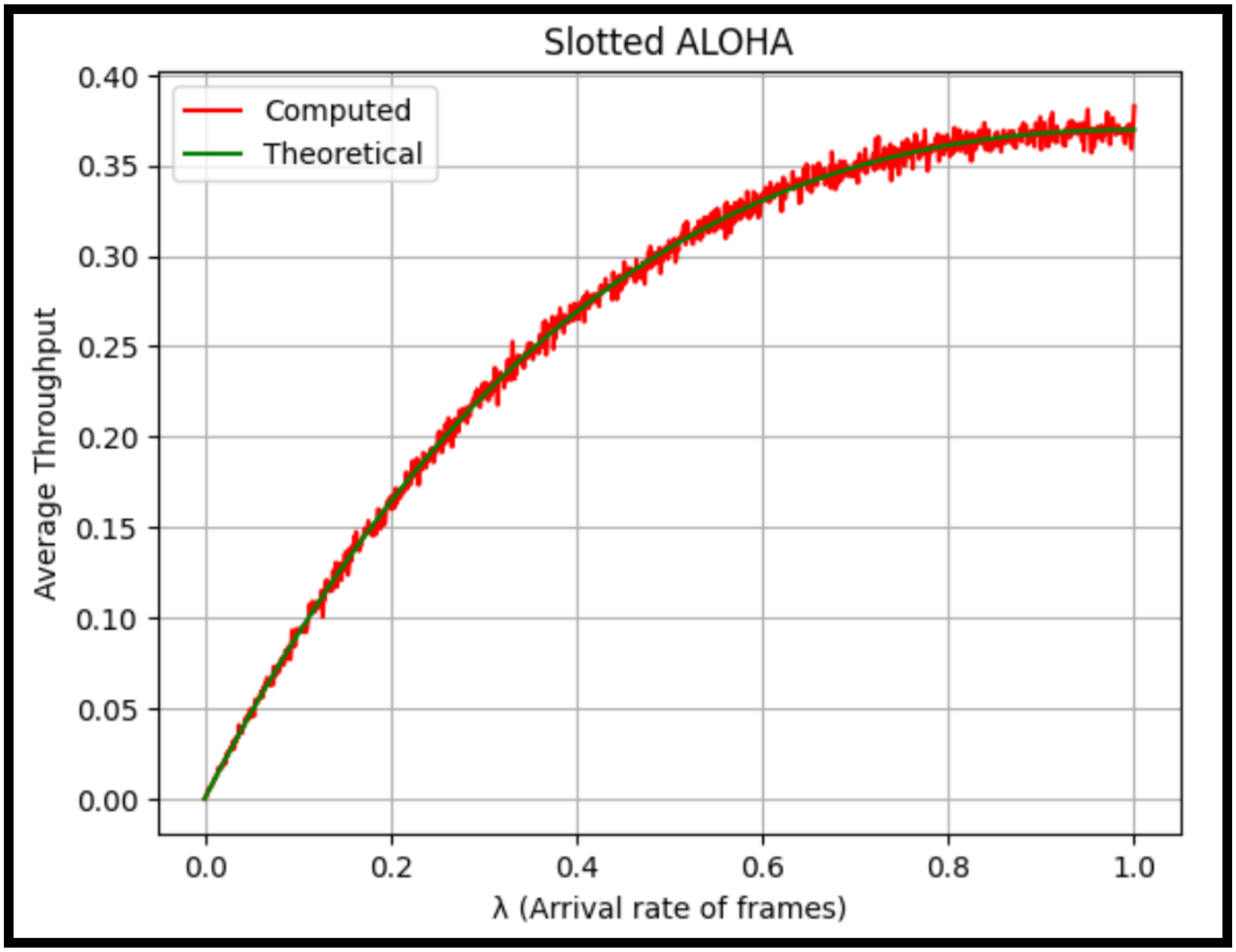
CS4150: Computer Networks Lab

Lab3

111901030
Mayank Singla

Q1. Consider a slotted system with n ≥ 1 user. In each slot, each user generates a new ethernet frame with probability λ/n, where λ ∈ [0, 1]. Assume that slot length is τ, and ethernet frame transmission time is T. All users use a common channel and have to compete with each other for an opportunity to successfully transmit their frames. For a long enough simulation run, average throughput is defined as the number of successful transmissions upon the number of slots.

(a) Use slotted ALOHA with T = τ and n = 100, and plot the average throughput as a function of λ. Plot the theoretical prediction and compare it with the simulation results. Assume that frames generated are not queued.



Above attached is the screenshot of the result I obtained with the following parameter values:

No. of users = 100
No. of slots = 10000
No. of λ values = 1000

To tackle this problem, I created a **User** class to represent each user which takes the probability of generating a new ethernet frame in each slot as input to its constructor. I defined a method **gen_frame** which generates a new ethernet frame with the above probability and returns **1** if a new frame is generated, else **0**. I created a random number in the interval **[0, 1)** using **random.random()** and if that number is less than the above probability, I will generate a new frame. This is to make sure that the frames are actually generated with the above probability.

Then, I defined all the constants for the problem. For simplicity, I chose the **SLOT_LEN** equal to **1**. I then generated uniform λ values in the given interval using **np.linspace()** and then compute the actual and the theoretical average throughput for each of the λ values and store it in a list to plot later using **matplotlib**.

For each of the λ values, I am creating the objects of the **User** class with the probability **(λ / NUM_USERS)** as mentioned in the question. Then, I am iterating over each slot to calculate the number of successful frame transmissions for each λ. I am maintaining a flag **wait_time** which will denote the time for which we have to wait before transmitting the frame i.e. time for which the channel is busy. This will be used whenever there is any successful transmission or collision, or the channel is idle. This value will be reduced by **SLOT_LEN** after every slot.

For each slot, I am first calling the **gen_frame()** method for each user because independent of whether the channel is idle or not, the user can generate a frame in each slot. Then, I am checking if **wait_time** is zero or not. If it is zero, then the channel is idle and we can try transmitting, else we will wait. *(Here, because frame transmission time**[FTT]** T is always equal to slot length τ, so the **wait_time** will always be zero)*.

Here, because the frames are not queued, the user will always try to transmit the frame if the channel is idle. Hence, to get to know which users will actually transmit the frame for the current slot, I am simply using the return value of the **gen_frame()** method because if a frame is generated and the channel is idle, then the user will try to transmit the frame.

Finally, I am simply checking the number of users who want to transmit in the current idle slot. If there is only **1** user, then we got a successful transmission and we will increase the count of successful transmissions and mark the channel as busy for **FTT** time. If there is no user, we will skip the slot by waiting for **SLOT_LEN** time. If more than **1** users try to transmit a frame, there will be a collision and the channel will be marked as busy for **FTT** time.

Finally, I am plotting the curve and the output is shown above.

The actual average throughput is calculated as **the number of successful transmissions upon the number of slots**.

The theoretical average throughput is calculated as

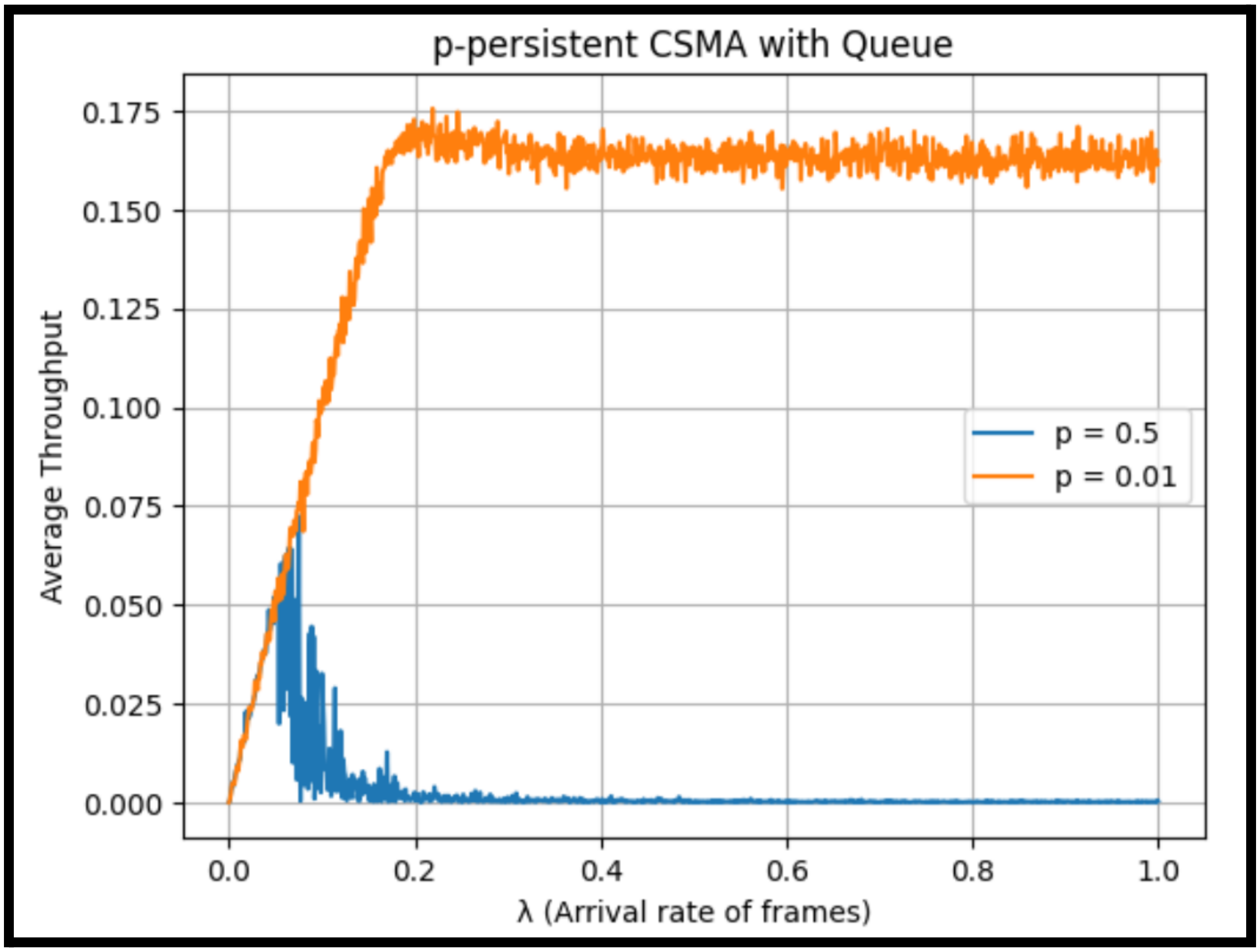
For one slot, the probability of no collision for 1 user = $(\lambda/n) * (1 - \lambda/n)^{n-1}$

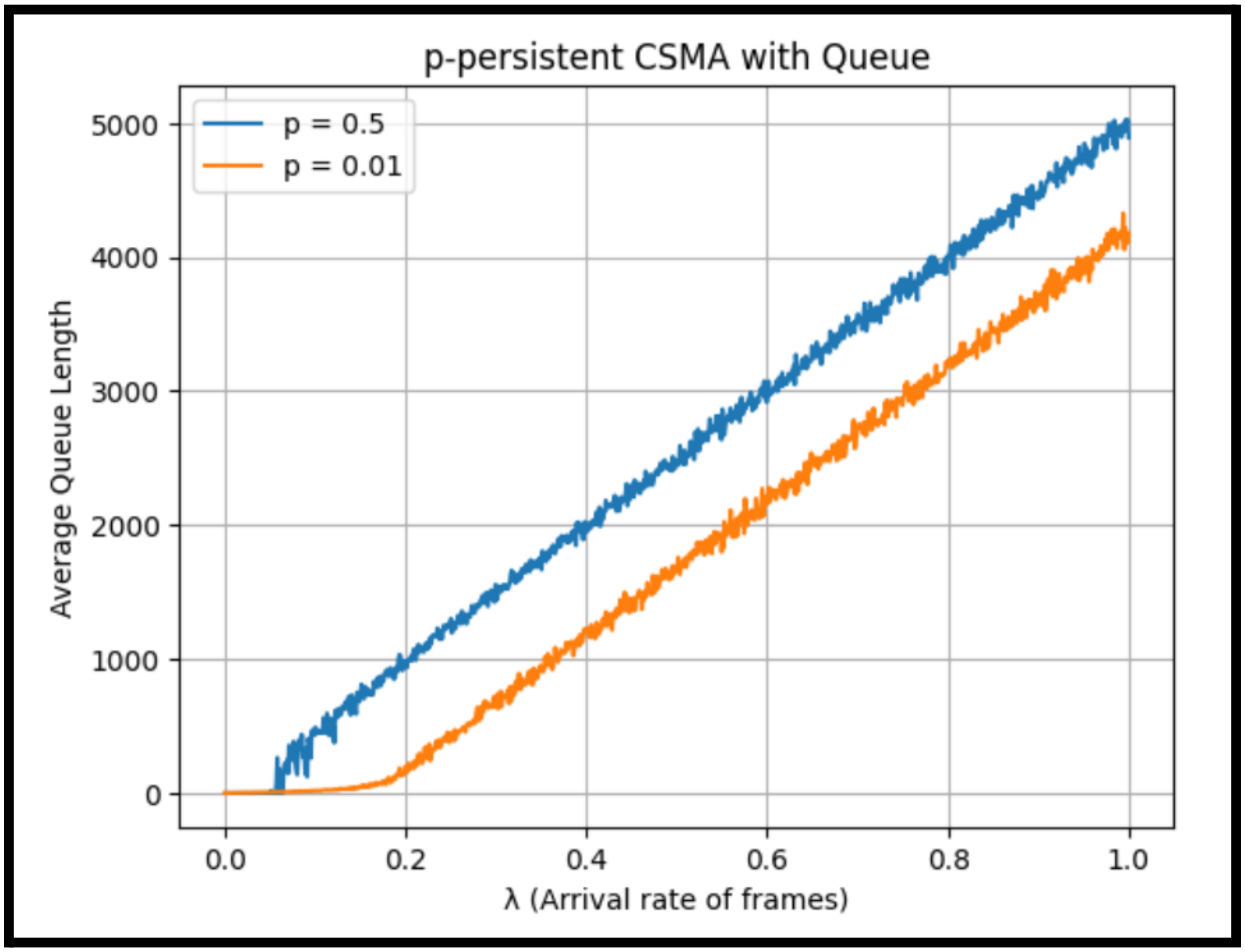
For one slot, the probability of no collision for n users, p = $n * (\lambda/n) * (1 - \lambda/n)^{n-1}$

p = $\lambda * (1 - \lambda/n)^{n-1}$

For **s** slots, the average throughput will be = **s * p / s = p**

(b) Frames generated are queued. Use p-persistent CSMA with T = 3τ and n = 100, and plot the average throughput as a function of λ for p = 0.5 and p = 0.01





Above attached are the resultant plots when using p-persistent CSMA and frames queuing with the following parameter values:

No. of users = 100
No. of slots = 10000
No. of λ values = 1000

For this problem, the modifications made to the **User** class are as follows. The constructor now also takes the additional parameter **trans_prob** which is the probability of transmitting a frame in an idle slot. The constructor is also creating an empty queue for storing frames for each user. The **gen_frame()** method is doing the same thing as in the previous question, just now it is also enqueuing value **1** (indicating frame) into the frame queue if a new ethernet frame is generated. I defined a new method **trans_fram()** which will try to transmit a frame from the queue if the queue is not empty. The logic for this method is the same as the method **gen_frame()** just it will compare the random number generated with **trans_prob** received in the constructor. I defined the method **trans_succ()** which will be called on successful transmission of the frame and it will dequeue the frame from the frame queue. Lastly, I am defining the method **queue_len()** to get the length of the frame queue of the user.

Next, I changed the constant **FTT** to **3 * SLOT_LEN** for this question. This time, I am having an extra for loop to calculate the average throughput and queue length for each given probability value.

Most of the logic for calculating the average throughput is the same as the previous question, just this time the logic for calculating the number of users who will transmit a frame is changed. Earlier, if a user generates a frame and the channel is idle, it will try to transmit that frame. Now, even if the user generates a frame and put it into the queue, it will only transmit based on the probability **p** of the p-persistent CSMA whose logic is handled by calling the **trans_frame()** method for each user.

After this, the whole logic for calculating the number of users trying to transmit a frame and waiting for the channel using the **wait_time** flag is done in the same way as in the previous question. Just, if we get a successful transmission, then I am also calling the **trans_succ()** method defined for that user who is trying to transmit the frame.

To calculate the average queue length for each λ value, I am using the below formula (as discussed in class)

Avg. Queue Length =

$$\boxed{\frac{1}{L}\sum_{K=1}^{L}\sum_{i=1}^{n} a_i[K]}$$

where,
L = no. of slots,
$\mathbf{a_i[K]}$ = no. of frames the **i**th user has in the queue when we are at the **K**th slot