EE-286: MOBILE AND WIRELESS NETWORKING

Dept. of Electrical Engineering

Project-1 Report

Advisor Prof. Jonathan Ponniah

Team members:

Darshan Shivarampura Rangaswamy, SJSU ID: 013823147

Pooja Dhuggenahalli Shivanna. SJSU ID: 013851526

Ranjitha Srinivasa, SJSU ID: 014534351

PART 1: The Geo/Geo/1 Queue.

Given a Geo/Geo/1 queue, consider the following scenarios:

i)An arrival rate of 0.2 and a service rate of 0.5

ii)An arrival rate of 0.49 and a service rate of 0.5

iii)An arrival rate of 0.6 and a service rate of 0.5

For each of the scenarios above:

A)Find and plot the empirical occupancy rate over ten million time slots (from K=0,…,9999999). On the same plot show the theoretical occupancy rate (if it exists)

B) Plot the empirical average of the queue length, $\frac{1}{K}\sum_{i=0}^{k} q[i]$ over ten million time slots (from K=0,…,9999999). On the same plot show the theoretical average queue length (if it exists).

Describe your strategy for simulation. Compare scenarios (i)-(iii) and why your plots make sense.

Answer:

[i] <u>Simulation of Geo/Geo/1 Queue:</u> An arrival rate of 0.2 and a service rate of 0.5

For Geo/Geo/1 Queue, the arrival and service processes are always Bernoulli.

Occupancy rate: Probability that queue has n packets. Fraction of time spent in a given state n.

In each time slot, we must simulate arrival and departure.

In each time slot "flip" an arrival coin, where the P(heads) =λ,P{tails) = (1 – λ). Since the arrival and service process are Bernoulli.

In this case,

Heads = packet arrives in time slot

Tails = packet does not arrive.

Theoretical occupancy rate: Determine theoretically P (being in each state), use global balance equation to find stationary distribution. We are going to compare theoretical values with empirical values at each time slot.

Empirical occupancy rate- The fraction of time spent in state **n.**

What we are doing at each state is counting the number of times(slots) that we {The System} are in each state and divide it by 10,000,000.

Example:

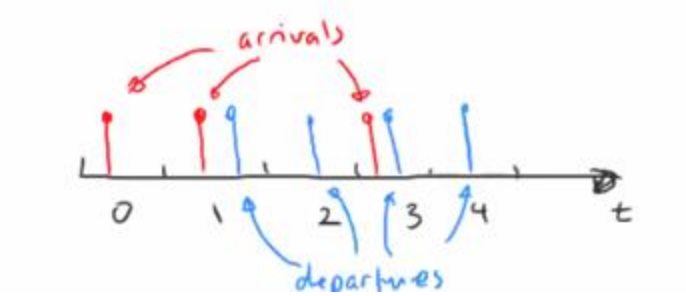
empirical occupancy rate for state 0 = (Number of time queue size = 0) / 10,000,000

empirical occupancy rate for state 1 = (Number of time queue size = 1) / 10,000,000

and this goes on…...

Need to do this for all states less than or equal to the maximum queue size reached in the experiment.

In each time slot we are simulating a Bernoulli arrival and a Bernoulli departure process.



So, we are running these 1,000,000 times and finding out the occupancy rate and plot it.

Theoretical occupancy rate: P (number of customers in queue = i)

$$= \rho^{i} (1- \rho) \ , \quad \text{where } \rho = \lambda(1 - \mu) / \mu(1 - \lambda)$$

Empirical average over queue length is given by: $\frac{1}{k}\sum_{i=0}^{k} q[i]$
for k = 0……10,000,000

Theoretical average queue length is equivalent to the Expectation or Average arrival rate.
Which is given by:
E [X] = ∑ x. p(x), here which is equal to $\frac{1}{k}\sum_{i=0}^{k} q[i]$
for k = 0……10,000,000

In each iteration, we have used:

For random Arrival: a = np.random.binomial(1, average_arrival_rate)

And for random Departure / Service: s = np.random.binomial(1,avg_service_rate)

This is used to simulate random variables in real time.

Now, q [i + 1] = max (0, q [i] + a - s)

We ran these 10 Million times and filled each value of array with the result of queue size.

This just gave us the number of packets at each time slot.

To find the occupancy rate, we found the maximum queue size over the entire routine.

Constructed an array of empirical_occupancy_rate.
Ran a For loop from 0 to runtime.

empirical_occupancy_rate [q [i]] += 1, we are incrementing the counter by 1.

Finally, Divided the elements in empirical_occupancy_rate by runtime to get the final results.

Observations:

For part (i), we have:

$\lambda = 0.2$ and $\mu = 0.5$
Here the service rate is much larger than arrival rate.
Therefore, modest queue size is observed.

For part (ii), we have:

$\lambda = 0.49$ and $\mu = 0.5$
Here, while plotting the pdf, the area under the curve must be 1 and the tail is observed to be heavier than before.

For part (iii), we have:

$\lambda = 0.6$ and $\mu = 0.5$
Since we have $\lambda > \mu$,
No stationary distribution exists for this scenario and we cannot use the formula $\rho^i (1- \rho)$, which is only for $\lambda < \mu$.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
run_time = 1000000
```

In [3]:
```
q = np.zeros(run_time, dtype=int) # time slots, Intitially queue size is set to zero for each time slot.
```

In [5]:
```
print('Number of time slots : ', len(q))
```

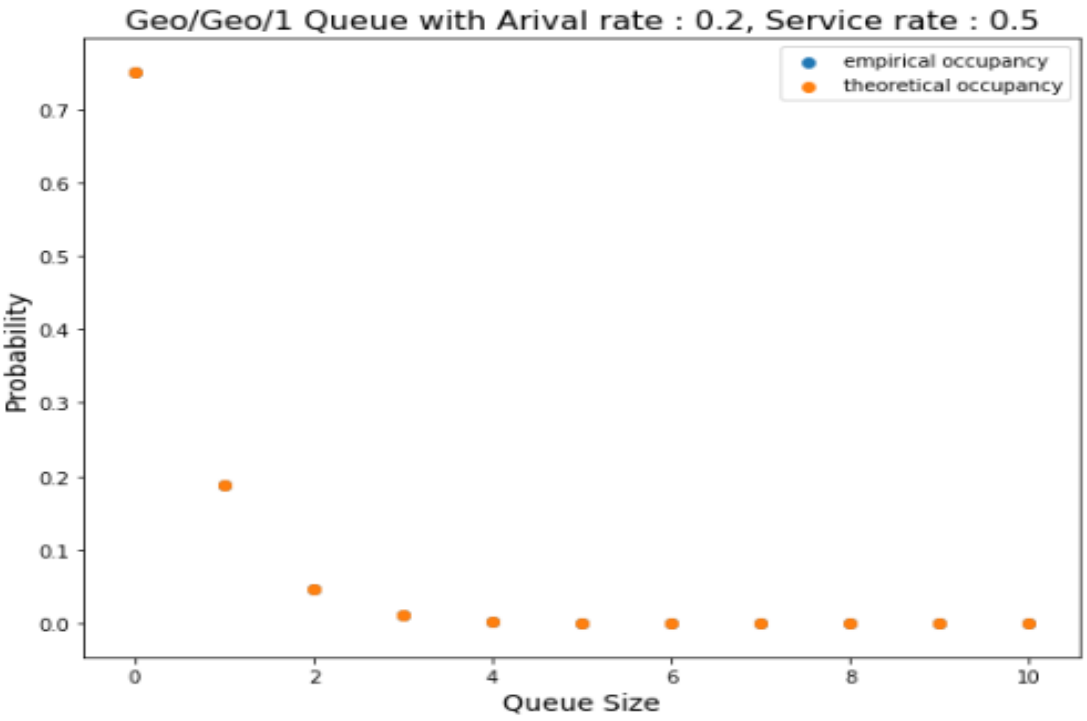
```
Number of time slots :  1000000
```

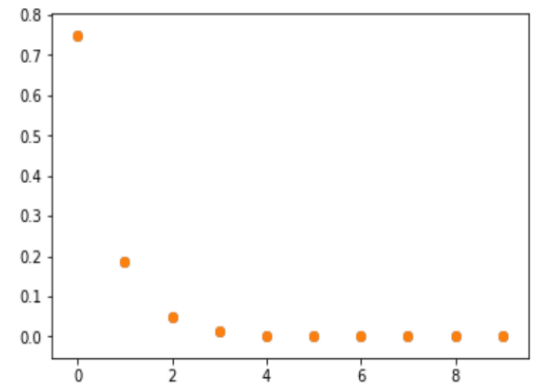
Therefore, the number of time slots = 1000000.

In [16]:

```
average_arrival_rate = 0.2
average_service_rate = 0.5

for i in range(len(q)):
    a = np.random.binomial(1, average_arrival_rate)
    s = np.random.binomial(1, average_service_rate)
    if i < len(q)-1:
        q[i+1] = max(q[i] + a - s, 0) # Calculate queue size in time slot i+1

        ## This gives number of packets in the queue at each time slot.

## Find occupancy rate

## Find maximum queue size over all time slots
max_queue_size = np.amax(q)

## Construct an array for empirical_occupancy
empirical_occupancy = np.zeros(max_queue_size+1)

for i in range(len(q)):
    empirical_occupancy[q[i]] = empirical_occupancy[q[i]]+1 ## Increment the counter by 1

empirical_occupancy = empirical_occupancy / run_time

theoretical_occupancy = np.zeros(max_queue_size + 1)

rho = (average_arrival_rate * (1 - average_service_rate)) / (average_service_rate *(1 - average_arrival_rate))

for i in range(len(theoretical_occupancy)):
    theoretical_occupancy[i] = (rho**i)*(1-rho)

t = np.arange(0,max_queue_size+1, dtype=int)

plt.figure(figsize = [9,7])
plt.scatter(t, empirical_occupancy, label = 'empirical occupancy')
plt.scatter(t, theoretical_occupancy, label = 'theoretical occupancy')
plt.title('Geo/Geo/1 Queue with Arival rate : 0.2, Service rate : 0.5', fontsize = 16)
plt.xlabel('Queue Size', fontsize = 14)
plt.ylabel('Probability', fontsize = 14)
plt.legend(loc = 'best')
plt.show()
```

OUTPUT:

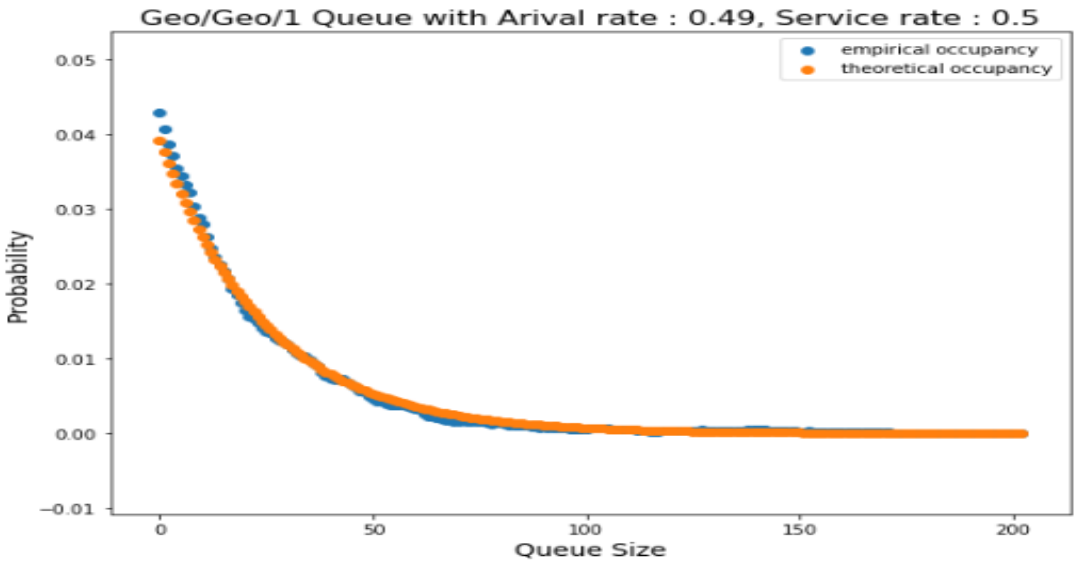


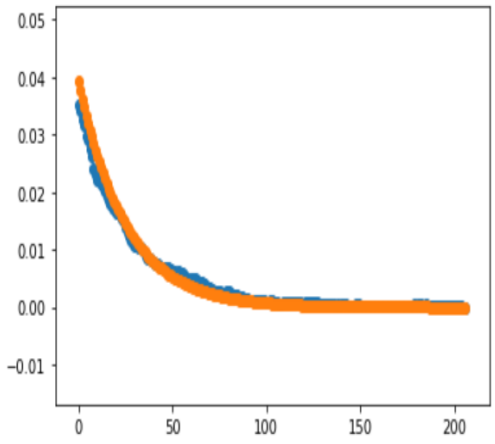


[ii] <u>Simulation of Geo/Geo/1 Queue:</u> An arrival rate of 0.49 and a service rate of 0.5

```
average_arrival_rate = 0.49
average_service_rate = 0.5

for i in range(len(q)):
    a = np.random.binomial(1, average_arrival_rate)
    s = np.random.binomial(1, average_service_rate)
    if i < len(q)-1:
        q[i+1] = max(q[i] + a - s, 0) # Calculate queue size in time slot i+1

        ## This gives number of packets in the queue at each time slot.

## Find occupancy rate

## Find maximum queue size over all time slots
max_queue_size = np.amax(q)

## Construct an array for empirical_occupancy
empirical_occupancy = np.zeros(max_queue_size+1)

for i in range(len(q)):
    empirical_occupancy[q[i]] = empirical_occupancy[q[i]]+1 ## Increment the counter by 1

empirical_occupancy = empirical_occupancy / run_time

theoretical_occupancy = np.zeros(max_queue_size + 1)

rho = (average_arrival_rate * (1 - average_service_rate)) / (average_service_rate *(1 - average_arrival_rate))

for i in range(len(theoretical_occupancy)):
    theoretical_occupancy[i] = (rho**i)*(1-rho)

t = np.arange(0,max_queue_size+1, dtype=int)

plt.figure(figsize = [9,7])
plt.scatter(t, empirical_occupancy, label = 'empirical occupancy')
plt.scatter(t, theoretical_occupancy, label = 'theoretical occupancy')
plt.title('Geo/Geo/1 Queue with Arival rate : 0.49, Service rate : 0.5', fontsize = 16)
plt.xlabel('Queue Size', fontsize = 14)
plt.ylabel('Probability', fontsize = 14)
plt.legend(loc = 'best')
plt.show()
```

OUTPUT:

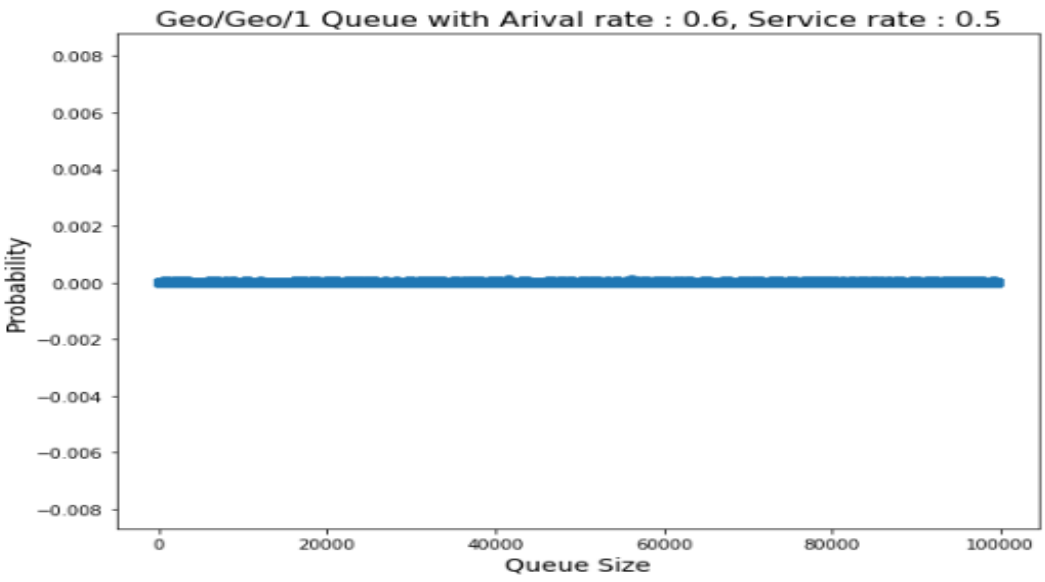


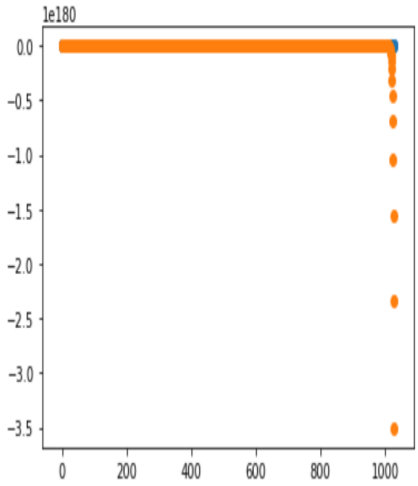


[iii] <u>Simulation of Geo/Geo/1 Queue:</u> An arrival rate of 0.6 and a service rate of 0.5

```
average_arrival_rate = 0.6
average_service_rate = 0.5

for i in range(len(q)):
    a = np.random.binomial(1, average_arrival_rate)
    s = np.random.binomial(1, average_service_rate)
    if i < len(q)-1:
        q[i+1] = max(q[i] + a - s, 0) # Calculate queue size in time slot i+1

        ## This gives number of packets in the queue at each time slot.

## Find occupancy rate

## Find maximum queue size over all time slots
max_queue_size = np.amax(q)

## Construct an array for empirical_occupancy
empirical_occupancy = np.zeros(max_queue_size+1)

for i in range(len(q)):
    empirical_occupancy[q[i]] = empirical_occupancy[q[i]]+1 ## Increment the counter by 1

empirical_occupancy = empirical_occupancy / run_time

theoretical_occupancy = np.zeros(max_queue_size + 1)

rho = (average_arrival_rate * (1 - average_service_rate)) / (average_service_rate *(1 - average_arrival_rate))

#for i in range(len(theoretical_occupancy)):
#    theoretical_occupancy[i] = (rho**i)*(1-rho)

t = np.arange(0,max_queue_size+1, dtype=int)

plt.figure(figsize = [9,7])
plt.scatter(t, empirical_occupancy, label = 'empirical occupancy')
#plt.scatter(t, theoretical_occupancy)
plt.title('Geo/Geo/1 Queue with Arival rate : 0.6, Service rate : 0.5', fontsize = 16)
plt.xlabel('Queue Size', fontsize = 14)
plt.ylabel('Probability', fontsize = 14)
plt.show()
```

OUTPUT:






```
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call last)
<ipython-input-2-4db3a04f5a4a> in <module>()
     22 (1 - average_arrival_rate))
     23 for i in range(len(theoretical_occupancy)):
---> 24     theoretical_occupancy[i] = (rho**i)*(1-rho)
     25 t = np.arange(0,max_queue_size+1,dtype=int)
     26 plt.scatter(t, empirical_occupancy)

OverflowError: (34, 'Numerical result out of range')
```

Observations and Comparison:

1)If the arrival rate is much smaller than the service rate, the queue size will be smaller. If the arrival rate is closer to the service rate, then the queue size will be larger as the high arrival rate leads to more packets in the queue.

2)Probability decreases if arrival rate closer to service rate. Probability of being in smaller states is decreasing, why is probability of being in lower state is decreasing relative to probability of being higher state

3)As probability of arrival approaches service rate, then, probability that the number of packets in lower states is more decreases

4)If arrival lower than service rate, then the probability of being in lower state is high. We also get almost similar curves for theoretical and empirical plots.

5)If arrival closer to service rate, then the probability of being in lower state is reduced and therefore the empirical curve deviates from the theoretical curve.

PART 2: The G/G/1 Queue.

The service process $s(k)$ of a G/G/1 queue has the following distribution: $P(s(k) = n) = \frac{1}{101}$ for $n = 0,\ldots,100$ and $P(s(k) = n) = 0$ for $n \geq 10$. Consider the following scenarios:

i) The arrival process $a(k)$ has the following distribution: $P(a(k) = n) = \frac{1}{45}$ for $n = 0,\ldots,44$

ii) The arrival process $a(k)$ has the following distribution: $P(a(k) = n) = \frac{1}{99}$ for $n = 0,\ldots,99$

For each of the scenarios above, plot the empirical average of the queue length, $\frac{1}{k}\sum_{i=0}^{k} q[i]$ over ten

million time slots (from $k = 0,\ldots,9999999$). On the same plot, show the upper-bound on the average
queue length of a G/G/1 queue. Describe your strategy for simulation. Compare scenarios (i)-(ii) and explain why your plots make sense.

SOLUTION:

In part (i)

We were given an uniform arrival process, where:

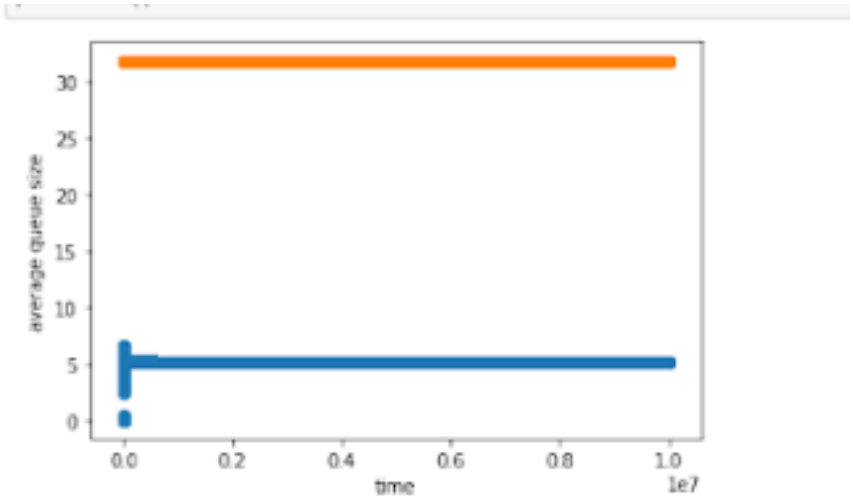
a [k] → **P** [a [k] = n] = 1 / 45 , for n = 0…….44.

and given uniform service process, where:

s [k] → **P** [s [k] = n] = 1 / 101 , for n = 0…….100.

```
import matplotlib.pyplot as plt
import numpy as np

run_time = 100000000
max_a = 44
max_s = 100
avg_a = (max_a-1)/2
avg_s = (max_s-1)/2
a_squared = (max_a-1)*(2*(max_a-1)+1)/6
s_squared = (max_s-1)*(2*(max_s-1)+1)/6
theoretical_ubound_q = np.full(run_time,(a_squared-2*avg_a*avg_s+s_squared)/(2*(avg_s-avg_a)))
q = np.zeros(run_time, dtype=int)
empirical_average_q = np.zeros(run_time)

temp1 = 0
for i in range(len(q)):
    a = np.random.randint(0,max_a)
    s = np.random.randint(0,max_s)
    if i < len(q)-1:
        q[i+1] = max(q[i] + a - s, 0)
        empirical_average_q[i+1]=((i+1)*empirical_average_q[i]+q[i+1])/(i+2)

t = np.arange(0,run_time,dtype=int)
plt.scatter(t, empirical_average_q)
plt.scatter(t, theoretical_ubound_q)
plt.xlabel("time")
plt.ylabel("average queue size")
plt.show()
```

OUTPUT:



In **scenario 1** (n= 0,….44) it can be observed that the arrival process is less than the service process. The queue is more stable. Since there is a significant difference in the arrival and service processes the upper bound is far from the average queue length.

We plotted two things:

Emperical average over queue length, which is given by:

for k = 0……10,000,000 $\frac{1}{k}\sum_{i=0}^{k} q[i]$

Theoretical average queue length, which is equivalent to the Expectation or Average arrival rate.

Which is given by:

E [X] = ∑ x. p(x), here which is equal to $1/k \sum_{i=0}^{k} i$,

for k = 0……10,000,000

Geo/Geo/1 and G/G/1 are two different things.

In Geo/Geo/1, it is known to us that the arrival and service process are Bernoulli.

But, in case of G/G/1, such condition is not known.

Here in this Problem, **we compared our Empirical result with the Theoretical Upper bound using Foster Lyapunov.**

In part (ii)

We were given an uniform arrival process, where:

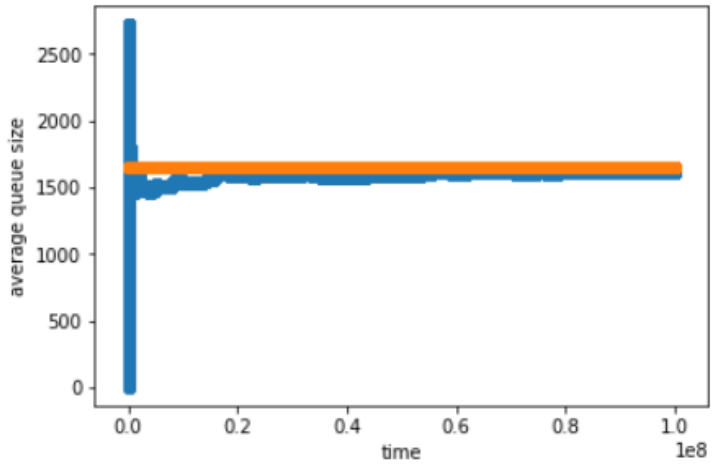
a [k] → **P** [a [k] = n] = 1 / 99 , for n = 0…….99.

and given uniform service process, where:

s [k] → **P** [s [k] = n] = 1 / 101 , for n = 0…….100.

```
import matplotlib.pyplot as plt
import numpy as np

run_time = 100000000
max_a = 99
max_s = 100
avg_a = (max_a-1)/2
avg_s = (max_s-1)/2
a_squared = (max_a-1)*(2*(max_a-1)+1)/6
s_squared = (max_s-1)*(2*(max_s-1)+1)/6
theoretical_ubound_q = np.full(run_time,(a_squared-2*avg_a*avg_s+s_squared)/(2*(avg_s-avg_a)))
q = np.zeros(run_time, dtype=int)
empirical_average_q = np.zeros(run_time)

temp1 = 0
for i in range(len(q)):
    a = np.random.randint(0,max_a)
    s = np.random.randint(0,max_s)
    if i < len(q)-1:
        q[i+1] = max(q[i] + a - s, 0)
        empirical_average_q[i+1]=((i+1)*empirical_average_q[i]+q[i+1])/(i+2)

t = np.arange(0,run_time,dtype=int)
plt.scatter(t, empirical_average_q)
plt.scatter(t, theoretical_ubound_q)
plt.xlabel("time")
plt.ylabel("average queue size")
plt.show()
```

OUTPUT:



In **scenario 2** (n =0 ,…99) it can be observed that the arrival process is close to the service process. The queue has some variations before it reaches to a steady state. Since the arrival process and service process are close the upper bound tends to be more accurate. As expected, it is evident from the graph that the average queue length converges with the upper bound.

The main difference between (i) and (ii) is the Avg (a [k]) and Avg (s [k]) in **(ii)** is very close.

And since they are very close, the **Theoretical Upper bound becomes very accurate**.

The average queue length converges in upper bound as shown in figure.