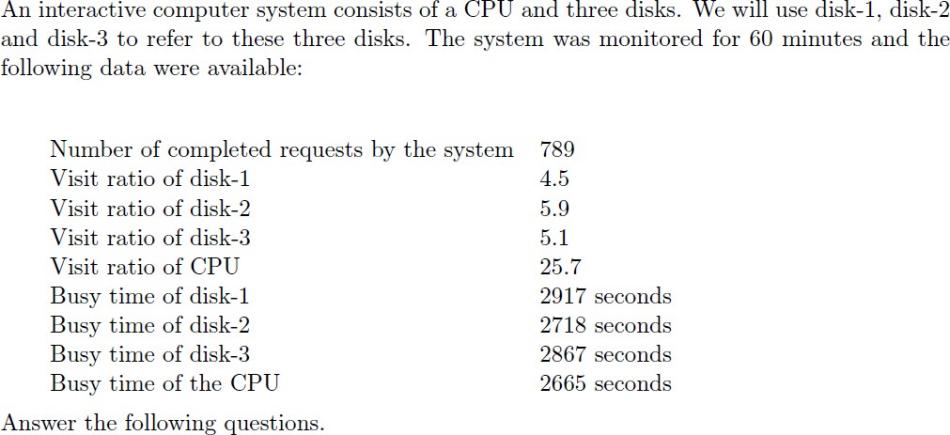
**COMP9334**

Question 1 (3 Marks):



### *Determine the service demands of disk-1, disk-2, disk-3 and the CPU.*

To get the Service demand of Disk-1, Disk-2, disk-3, and the CPU, we need to use the Service Demand Law:

𝑫(𝒋) =

𝑼(𝒊)

𝑿(𝟎)

The U(i) refer to the Utilization of devices, which mean that we need to calculate the Utilization of U(Disk-1), U(Disk-2), U(Disk-3) and U(CPU) and the Throughput of the System. Meanwhile we will convert the Monitor time from minute to second to match with the provided Busy time.

𝑈(𝐷𝑖𝑠𝑘1) =

𝐵(𝐷𝑖𝑠𝑘1)

=

𝑇

2917

60 ∗ 60

≈ 0.810 𝑈(𝐷𝑖𝑠𝑘2) =

𝐵(𝐷𝑖𝑠𝑘2)

=

𝑇

2718

60 ∗ 60

= 0.755

𝑈(𝐷𝑖𝑠𝑘3) =

𝐵(𝐷𝑖𝑠𝑘3)

=

𝑇

2867

60 ∗ 60

≈ 0.796 𝑈(𝐶𝑃𝑈) =

𝐵(𝐶𝑃𝑈)

=

𝑇

2665

60 ∗ 60

≈ 0.740

After calculate the Utilization of each devices, we need to get the Throughput of the System X(0).

𝑋(0) =

𝐶(0)

=

𝑇

789

60 ∗ 60

≈ 0.219 (𝐶𝑜𝑚𝑝𝑙𝑒𝑐𝑡𝑖𝑜𝑛/𝑆)

Since we retrieve the X(0) and the Utilization of each devices, so we can calculate the Service Demand of each device of the System.

𝐷(𝐷𝑖𝑠𝑘1) =

𝑈(𝐷𝑖𝑠𝑘1)

=

𝑋(0)

0.810

0.219

= 3.698 𝐷(𝐷𝑖𝑠𝑘2) =

𝑈(𝐷𝑖𝑠𝑘2)

=

𝑋(0)

0.755

0.219

= 3.447

𝐷(𝐷𝑖𝑠𝑘3) =

𝑈(𝐷𝑖𝑠𝑘3)

=

𝑋(0)

0.796

0.219

= 3.635 𝐷(𝐶𝑃𝑈) =

𝑈(𝐶𝑃𝑈)

=

𝑋(0)

0.740

0.219

= 3.379

### *Use bottleneck analysis to determine the asymptotic bound on the system throughput when* there are 4 interactive users, and the think time is 20 seconds.

Bottleneck Analysis:

1

𝑋(0) ≤ min [

𝑁

, ∑𝐾 ]

𝑚𝑎𝑥 𝐷𝑖

𝑖=1 𝐷𝑖

The first throughput bond will be limited by the Maximum Service demand of a device within the System. The service demand from highest to lowest: *D(Disk1) > D(Disk3) > D(CPU) > Disk (Disk2).* So, the first throughput bound value will be:

1

=

𝑀𝑎𝑥 𝐷𝑖

1

3.698

= 0.27042 (jobs/s)

The Second bond, N is the number of Interactive Users and sums the service demand of all devices. Additionally, the throughput bound will also affect by the Thinking time:

𝐾

∑

𝑖=𝑖

𝑁

=

𝐷𝑖 + 𝑇ℎ𝑖𝑛𝑘𝑖𝑛𝑔 𝑇𝑖𝑚𝑒

𝑁

𝐷(𝐷𝑖𝑠𝑘1) + 𝐷(𝐷𝑖𝑠𝑘2) + 𝐷(𝐷𝑖𝑠𝑘3) + 𝐷(𝐶𝑃𝑈) + 𝑇ℎ𝑖𝑛𝑘𝑖𝑛𝑔 𝑇𝑖𝑚𝑒

4

= =

3.698 + 3.447 + 3.635 + 3.379 + 20

4

34.1159

= 0.11725 (𝑗𝑜𝑏𝑠/𝑠)

Thus, by using the Bottleneck Analysis to get the Asymptotic bound:

1

𝑋(0) ≤ min [

𝑁

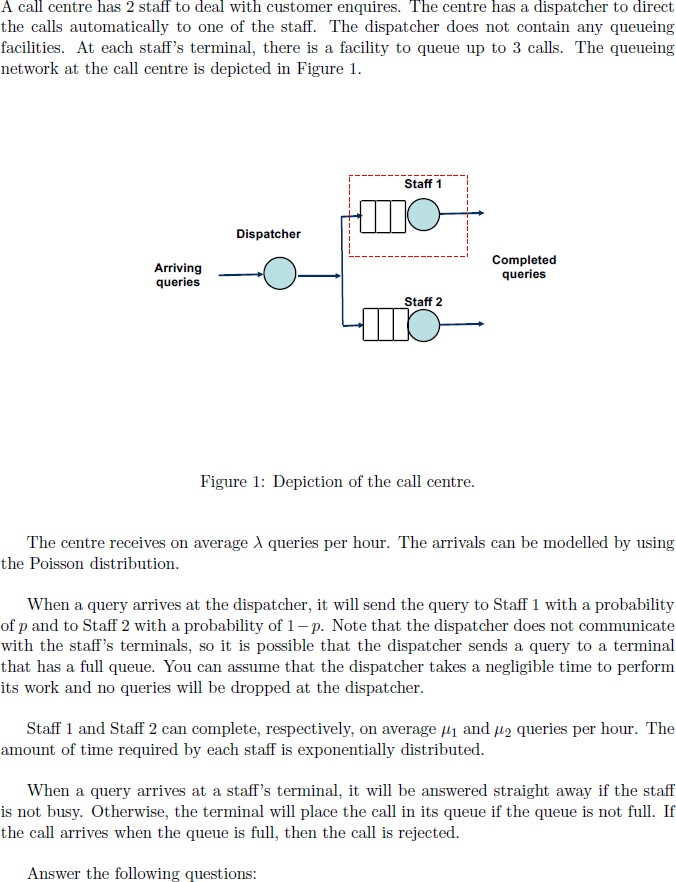
, ∑𝐾

] = 𝑚𝑖𝑛[0.27042, 0.11725] = 0.11725

𝑚𝑎𝑥 𝐷𝑖

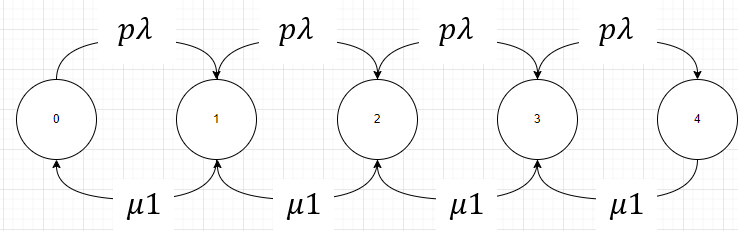
𝑖=1 𝐷𝑖

## Question 2 (7 Marks)



### *Formulate a continuous-time Markov chain for the part of the call centre consisting of Staff* 1 and their three waiting slots

The continuous-time Markov Chain for the part of the Call centre Consisting of Staff one described as Below:



Brief explanation of the term:

* + - 𝑝: The probability of Call that assign to Staff 1 from Dispatcher.
    - 𝜆: The Centre receives on average queries per hour.
    - 𝜇1: Staff 1 complete average queries per hour.
    - 𝑝𝜆: 𝑝 \* 𝜆 in result can calculate the queries that assign to Staff 1.

## Define the States:

* + - State 0: Staff 1 is Idle and waiting for calls.
    - State 1: Staff 1 receives a call and serve the query right away.
    - State 2: Staff 1 is serving a call, one query in waiting slots.
    - State 3: Staff 1 is serving a call, two queries in waiting slots.
    - State 4: Staff 1 is serving a call, three queries in waiting slots and the slots are full. A further query is assigned to Staff 1 will be rejected.

### *Write down the balance equations for the continuous-time Markov chain that you have* formulated.

Brief Explanation of the Terms:

* + - 𝑃i: Probability in State i.

Balance Equation List:

- 𝑝𝜆𝑃0 = 𝜇1𝑃1

- 𝑝𝜆𝑃1 = 𝜇1𝑃2

- 𝑝𝜆𝑃2 = 𝜇1𝑃3

- 𝑝𝜆𝑃3 = 𝜇1𝑃4

- 𝑝𝜆𝑃0 + 𝜇1𝑃2 = (𝜇 1 + 𝑝𝜆) 𝑃1

- 𝑝𝜆𝑃1 + 𝜇1𝑃3 = (𝜇 1 + 𝑝𝜆) 𝑃2

- 𝑝𝜆𝑃2 + 𝜇1𝑃4 = (𝜇 1 + 𝑝𝜆) 𝑃3

* 1. ***Derive the expressions for the steady state probabilities of the continuous-time Markov chain that you have formulated.***

# P0 + P1 + P2 + P3 + P4 = 1

## Steady State for P0:

|  |  |  |  |
| --- | --- | --- | --- |
| 𝑝𝜆𝑃0 = 𝜇1𝑃1  𝑝𝜆  => 𝑃1 = 𝑃0  𝜇1 | 𝑝𝜆𝑃1 = 𝜇1𝑃2  𝑝𝜆  => 𝑃2 = 𝑃1  𝜇1  𝑝𝜆 2  => 𝑃2 = ( ) ∗ 𝑃0  𝜇1 | 𝑝𝜆𝑃2 = 𝜇1𝑃3  𝑝𝜆  => 𝑃3 = 𝑃2  𝜇1  𝑝𝜆 3  => 𝑃3 = ( ) ∗ 𝑃0  𝜇1 | 𝑝𝜆𝑃3 = 𝜇1𝑃4  𝑝𝜆  => 𝑃4 = 𝑃3  𝜇1  𝑝𝜆 4  => 𝑃2 = ( ) ∗ 𝑃0  𝜇1 |

𝑃0

𝑝𝜆 \* 𝑃

𝜇1

+

0

𝑝𝜆 2

( )

+

𝜇1

∗ 𝑃0 +

𝑝𝜆 3

( )

𝜇1

∗ 𝑃0 +

𝑝𝜆 4

( )

𝜇1

∗ 𝑃0 = 1

𝑃 (1 + 𝑝𝜆

0

𝑝𝜆 2

𝑝𝜆 3

𝑝𝜆

4

# ) = 1

+ ( )

𝜇1 𝜇1

𝑃0 =

+ ( )

𝜇1

1

2

(

+ ( )

𝜇1

3 4

) (

)

𝑝𝜆

1+

𝑝𝜆

+

) + (

𝑝𝜆

𝑝𝜆

+

𝜇1

𝜇1

𝜇1

𝜇1

## Steady State for P1:

|  |  |  |  |
| --- | --- | --- | --- |
| 𝑝𝜆𝑃0 = 𝜇1𝑃1  => 𝑃 = 𝜇1 𝑃  0 𝑝𝜆 1 | 𝑝𝜆𝑃1 = 𝜇1𝑃2  => 𝑃 = 𝑝𝜆 ∗ 𝑃  2 𝜇1 1 | 𝑝𝜆𝑃2 = 𝜇1𝑃3  => 𝑃 = 𝑝𝜆 𝑃  3 𝜇1 2  𝑝𝜆 2  => 𝑃3 = ( ) ∗ 𝑃1  𝜇1 | 𝑝𝜆𝑃3 = 𝜇1𝑃4  => 𝑃 = 𝑝𝜆 𝑃  4 𝜇1 3  𝑝𝜆 3  => 𝑃2 = ( ) ∗ 𝑃1  𝜇1 |

𝜇1 ∗ 𝑃

1

𝑝𝜆

+ 𝑃1

𝑝𝜆

+ 𝑃1

+

𝜇1

𝑝𝜆 2

( )

𝜇1

𝑝𝜆 3

1 ( )

∗ 𝑃 +

𝜇1

∗ 𝑃1 = 1

𝑃1

𝜇1 𝑝𝜆

( + 1 +

𝑝𝜆 2

+ ( )

+ (𝑝𝜆

3

) ) = 1

𝑝𝜆

𝜇1

1∗ 𝑝𝜆

𝜇1

𝜇1

𝑃1 = 𝜇1

𝜇1

(

𝑝𝜆

+1+

𝑝𝜆

+ (

𝜇1

𝑝𝜆 2

) + (

𝜇1

𝑝𝜆 3

) )∗

𝜇1

𝑝𝜆

𝜇1

𝑝𝜆

𝑃1 = 𝜇1

𝑝𝜆

1+ +(

𝜇1

𝑝𝜆 2

) + (

𝜇1

𝑝𝜆 3

) + (

𝜇1

𝑝𝜆 4

)

𝜇1

## Steady State for P2:

|  |  |  |  |
| --- | --- | --- | --- |
| 𝑝𝜆𝑃0 = 𝜇1𝑃1  => 𝑃 = 𝜇1 𝑃  0 𝑝𝜆 1  => 𝑃 = 𝜇1)2 𝑃  0 ( 2  𝑝𝜆 | 𝑝𝜆𝑃1 = 𝜇1𝑃2  => 𝑃 = 𝜇1 ∗ 𝑃  1 𝑝𝜆 2 | 𝑝𝜆𝑃2 = 𝜇1𝑃3  => 𝑃 = 𝑝𝜆 𝑃  3 𝜇1 2 | 𝑝𝜆𝑃3 = 𝜇1𝑃4  => 𝑃 = 𝑝𝜆 𝑃  4 𝜇1 3  𝑝𝜆 2  => 𝑃2 = ( ) ∗ 𝑃2  𝜇1 |

𝜇1

2

( )

𝑝𝜆

𝑃2 +

𝜇1

𝑝𝜆

∗ 𝑃2 + 𝑃2 +

𝑝𝜆

𝜇1

𝑝𝜆 2

2 ( )

+

𝑃

𝜇1

∗ 𝑃2 = 1

𝑃 ( 𝜇1 2

2 ( )

𝑝𝜆

+ 𝜇1

𝑝𝜆

+ 1 +

1∗ (𝑝𝜆 2

)

𝑝𝜆

𝜇1

𝑝𝜆 2

( )

+

𝜇1

) = 1

𝑃2 = 𝜇1

((𝜇1 2

) +

+1+

+ (

) )∗(

)

𝑝𝜆

𝜇1

𝑝𝜆

𝑝𝜆

𝜇1

(𝑝𝜆 2

)

𝑝𝜆 2

𝜇1

𝑝𝜆 2

𝜇1

𝑃2 = 𝜇1

1 + 𝑝𝜆 + (𝑝𝜆 2 𝑝𝜆 3 𝑝𝜆 4

𝜇1

𝜇1

𝜇1

𝜇1

) + (

) + ( )

## Steady State for P3:

|  |  |  |  |
| --- | --- | --- | --- |
| 𝑝𝜆𝑃0 = 𝜇1𝑃1  => 𝑃 = 𝜇1 𝑃  0 𝑝𝜆 1  => 𝑃 = 𝜇1)3 𝑃  0 ( 3  𝑝𝜆 | 𝑝𝜆𝑃1 = 𝜇1𝑃2  𝜇1  => 𝑃1 = ( )2 ∗ 𝑃3  𝑝𝜆 | 𝑝𝜆𝑃2 = 𝜇1𝑃3  => 𝑃 = 𝜇1 ∗ 𝑃  2 𝑝𝜆 3 | 𝑝𝜆𝑃3 = 𝜇1𝑃4  => 𝑃 = 𝑝𝜆 𝑃  4 𝜇1 3  𝑝𝜆 1  => 𝑃4 = ( ) ∗ 𝑃3  𝜇1 |

𝜇1

3

( )

𝑝𝜆

𝑃 + 𝜇1)2

𝑝𝜆

3 (

∗ 𝑃3 +

𝜇1

𝑝𝜆

∗ 𝑃 𝑃 𝑝𝜆)1

𝜇1

+ +

3 3 (

∗ 𝑃3 = 1

𝑃 ( 𝜇1 3

𝜇1 2

𝜇1

𝑝𝜆 1

3 ( )

𝑝𝜆

+ ( )

𝑝𝜆

+

𝑝𝜆

+ 1 + (

𝜇1

) ) = 1

1∗ (𝑝𝜆 3

)

𝑃3 = 𝜇1

𝜇1

(( 3

2

1

3

) + (

𝑝𝜆

𝜇1

) +

𝑝𝜆

𝜇1

+1+ (

𝑝𝜆

𝑝𝜆

) ))∗ (

𝜇1

𝑝𝜆

)

𝜇1

(𝑝𝜆 3

)

𝑃3 = 𝜇1

1 + 𝑝𝜆 𝑝𝜆 2 𝑝𝜆 3 𝑝𝜆 4

𝜇1 + ( ) + ( ) + ( ) )

𝜇1

𝜇1

𝜇1

## Steady State for P4:

|  |  |  |  |
| --- | --- | --- | --- |
| 𝑝𝜆𝑃0 = 𝜇1𝑃1  => 𝑃 = 𝜇1 𝑃  0 𝑝𝜆 1  => 𝑃 = 𝜇1)4 𝑃  0 ( 4  𝑝𝜆 | 𝑝𝜆𝑃1 = 𝜇1𝑃2  => 𝑃 = 𝜇1)3 ∗ 𝑃  1 ( 4  𝑝𝜆 | 𝑝𝜆𝑃2 = 𝜇1𝑃3  => 𝑃 = 𝜇1)2 ∗ 𝑃  2 ( 4  𝑝𝜆 | 𝑝𝜆𝑃3 = 𝜇1𝑃4  => 𝑃 = 𝜇1 𝑃  3 𝑝𝜆 4  => 𝑃 = 𝜇1 ∗ 𝑃  3 𝑝𝜆 4 |

𝜇1

4

( )

𝑝𝜆

𝑃 + 𝜇1)3

𝑝𝜆

4 (

∗ 𝑃 + 𝜇1)2

𝑝𝜆

4 (

1∗ (𝑝𝜆 4

)

∗ 𝑃4 +

𝜇1

𝑝𝜆

∗ 𝑃4 + 𝑃4 = 1

𝑃4 = 𝜇1

𝜇1

(( 4

3

2

4

) + (

𝑝𝜆

𝜇1

) + (

𝑝𝜆

𝜇1

) +

𝑝𝜆

𝜇1

𝑝𝜆

+ 1)∗ (

𝑝𝜆

)

𝜇1

(𝑝𝜆 4

)

𝑃4 = 𝜇1

1 + 𝑝𝜆 + (𝑝𝜆 2 𝑝𝜆 3 𝑝𝜆 4

𝜇1

𝜇1

𝜇1

𝜇1

) + (

) + ( )

### *Assuming that* 𝒑 *= 0.4,* 𝝀 *= 5.7 and* 𝝁 *1 = 6.1. Determine the probability that a query that is* dispatched to Staff 1 will be rejected.

Query from Dispatched to Staff 1 get rejected mean that Staff 1 must be in State 4 as other state will have Waiting slot for query.

𝑝𝜆 4

( ) (

𝑃 = 𝜇1 =

0.4∗5.7 4

6.1 )

≈ 0.0123

4 𝑝𝜆 𝑝𝜆 2

𝑝𝜆 3 𝑝𝜆 4

0.4∗5.7

0.4∗5.7 2

0.4∗5.7 3

0.4∗5.7 4

1 + 𝜇1 + (

𝜇1

𝜇1

𝜇1

) + ( ) + ( )

1+(

6.1 )+ (

6.1 ) +(

6.1 ) + (

6.1 )

* 1. ***Assuming that*** 𝒑 ***= 0.4,*** 𝝀 ***= 5.7,*** 𝝁 ***1 = 6.1 and*** 𝝁 ***2 = 6.5, determine the mean waiting time of the queries that have not been rejected by the call centre. Note that Part (d) considers only queries that have been dispatched to Sta\_ 1 but Part (e) considers the whole call centre.***

## Staff 1 (Steady State Calculation):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P0 =** | 1+ | 𝑝𝜆  𝜇1 | 1  2 3 4 ≈ 0.63083  𝑝𝜆 𝑝𝜆 𝑝𝜆  +( ) + ( ) + ( )  𝜇1 𝜇1 𝜇1 | | | | **P1 =** | 1+ | 𝑝𝜆  𝜇1  2 3 4 ≈ 0.23578  𝑝𝜆 𝑝𝜆 𝑝𝜆 𝑝𝜆  +( ) + ( ) + ( )  𝜇1 𝜇1 𝜇1 𝜇1 |
| **P2 =** | 1+ | 𝑝𝜆  𝜇1 | 𝑝𝜆 2  ( )  𝜇1  2 3 4 ≈ 0.08812  𝑝𝜆 𝑝𝜆 𝑝𝜆  +( ) + ( ) + ( )  𝜇1 𝜇1 𝜇1 | | | | **P3 =** | 1+ | 𝑝𝜆 3  ( )  𝜇1  2 3 4 ≈ 0.03294  𝑝𝜆 𝑝𝜆 𝑝𝜆 𝑝𝜆  +( ) + ( ) + ( )  𝜇1 𝜇1 𝜇1 𝜇1 |
|  |  |  | **P4 =** | 1+ | 𝑝𝜆  𝜇1 | 𝑝𝜆 4  ( )  𝜇1  2 3 4 ≈ 0.01231  𝑝𝜆 𝑝𝜆 𝑝𝜆  +( ) + ( ) + ( )  𝜇1 𝜇1 𝜇1 | | | |

Since any request that arrives at stage 4 will be rejected, so we exclude it for the mean number of jobs calculation.

3

N1 = ∑

𝑘=0

𝑘𝑃k **=** 0 ∗ 0.63083 + 1 ∗ 0.23578 + 2 ∗ 0.08812 + 3 ∗ 0.03294 ≈ 0.51084

## According to Little’s Law: Mean number of Job = Throughput x Response Time

Response Time (Staff 1) = N1/ 𝑝𝜆 = 0.51084/ (0.4 \* 5.7) = 0.22405

## Mean waiting time = Mean Response Time – Mean Service Time

Waiting time (Staff 1) = 0.22405 – 1/6.5= 0.07020

## Staff 2 (Steady State Calculation):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P0 =** | 1+ | (1−𝑝)𝜆  𝜇2 | 1  (1−𝑝)𝜆 2 (1−𝑝)𝜆  +( ) + (  𝜇2 𝜇2 | 3  ) + ( | 4 ≈ 0.49375  (1−𝑝)𝜆  )  𝜇2 | | | (1−𝑝)𝜆  𝜇2  **P1 =** 2 3 4 ≈ 0.25979  (1−𝑝)𝜆 (1−𝑝)𝜆 (1−𝑝)𝜆 (1−𝑝)𝜆  1+ +( ) + ( ) + ( )  𝜇2 𝜇2 𝜇2 𝜇2 |
| **P2 =** | 1+ | (1−𝑝)𝜆  𝜇2 | (1−𝑝)𝜆 2  ( )  𝜇2  (1−𝑝)𝜆 2 (1−𝑝)𝜆  +( ) + (  𝜇2 𝜇2 | 3  ) + ( | 4 ≈ 0.13669  (1−𝑝)𝜆  )  𝜇2 | | | (1−𝑝)𝜆 3  ( )  𝜇2  **P3 =** 2 3 4 ≈ 0.07192  (1−𝑝)𝜆 (1−𝑝)𝜆 (1−𝑝)𝜆 (1−𝑝)𝜆  1+ +( ) + ( ) + ( )  𝜇2 𝜇2 𝜇2 𝜇2 |
|  |  |  |  | **P4 =** | 1+ | (1−𝑝)𝜆  𝜇2 | (1−𝑝)𝜆 4  ( )  𝜇2  2 3 4 ≈ 0.03784  (1−𝑝)𝜆 (1−𝑝)𝜆 (1−𝑝)𝜆  +( ) + ( ) + ( )  𝜇2 𝜇2 𝜇2 | |

3

N1 = ∑

𝑘=0

𝑘𝑃k **=** 0 ∗ 0.49375 + 1 ∗ 0.25979 + 2 ∗ 0.13669 + 3 ∗ 0.07192 ≈ 0.74893

Response Time (Staff 2) = N1/ (1 − 𝑝)𝜆 = 0. 74893/ (0.6 \* 5.7) = 0.21898 Waiting time (Staff 2) = 0.21898 – 1/6.1 = 0.06011

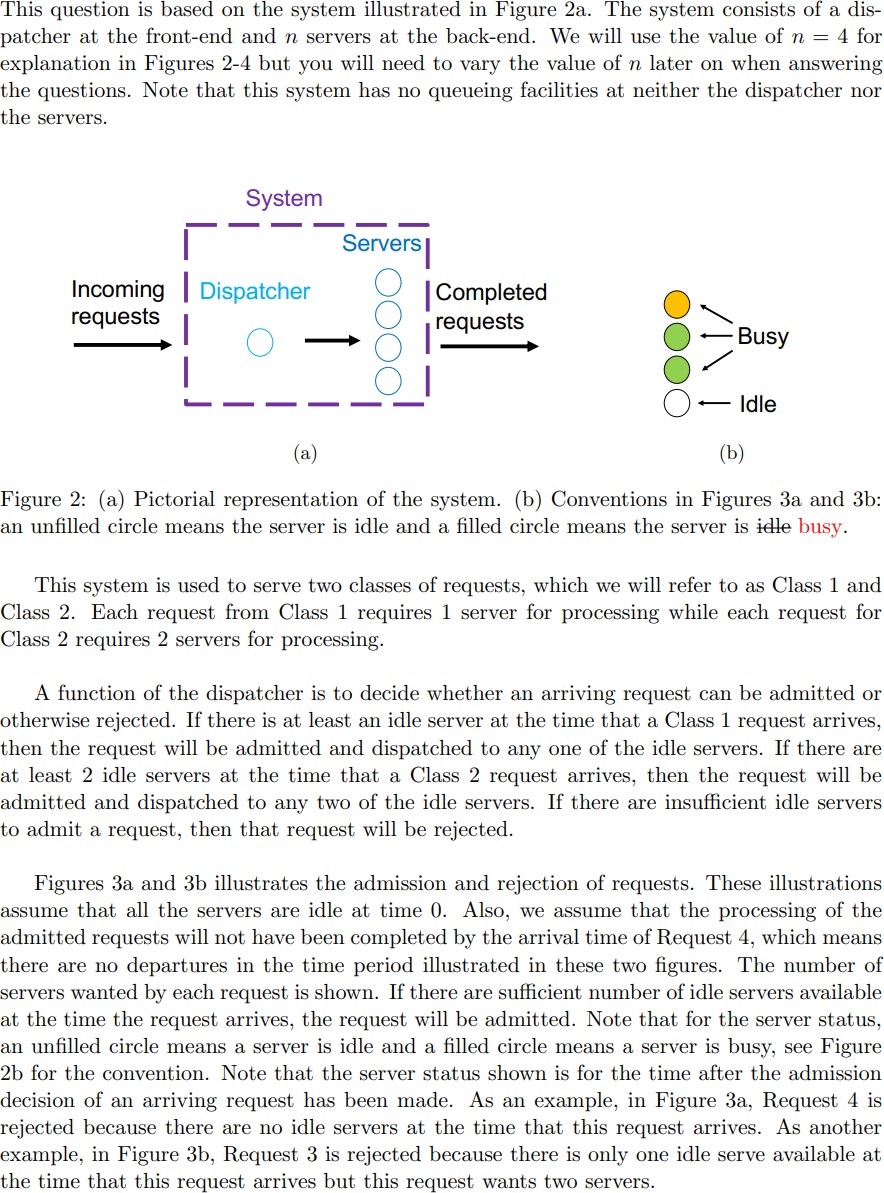
Mean Waiting time (Whole Call Centre & No Reject Call)

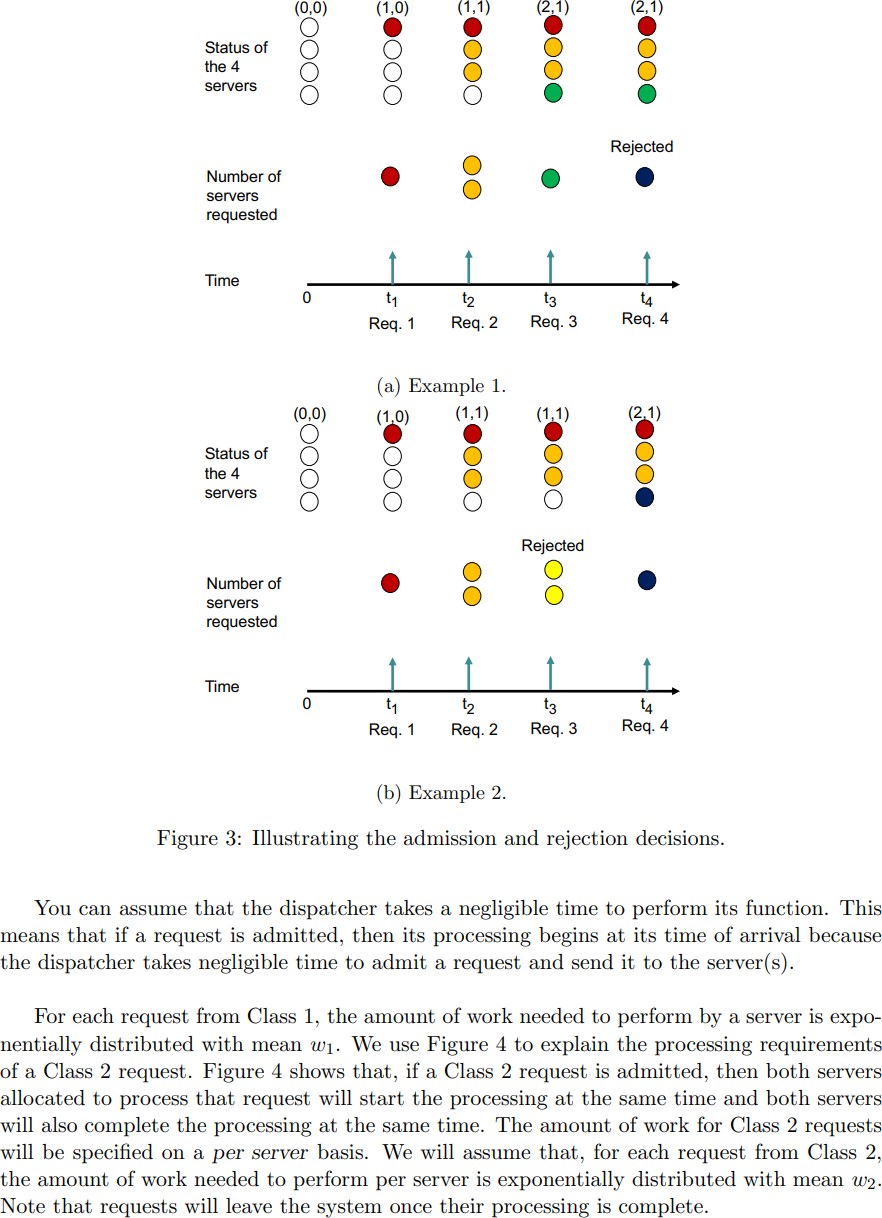
= Mean Waiting time (Staff 1) \* 𝑝 + Mean Waiting time (Staff 2) \* (1 − 𝑝)

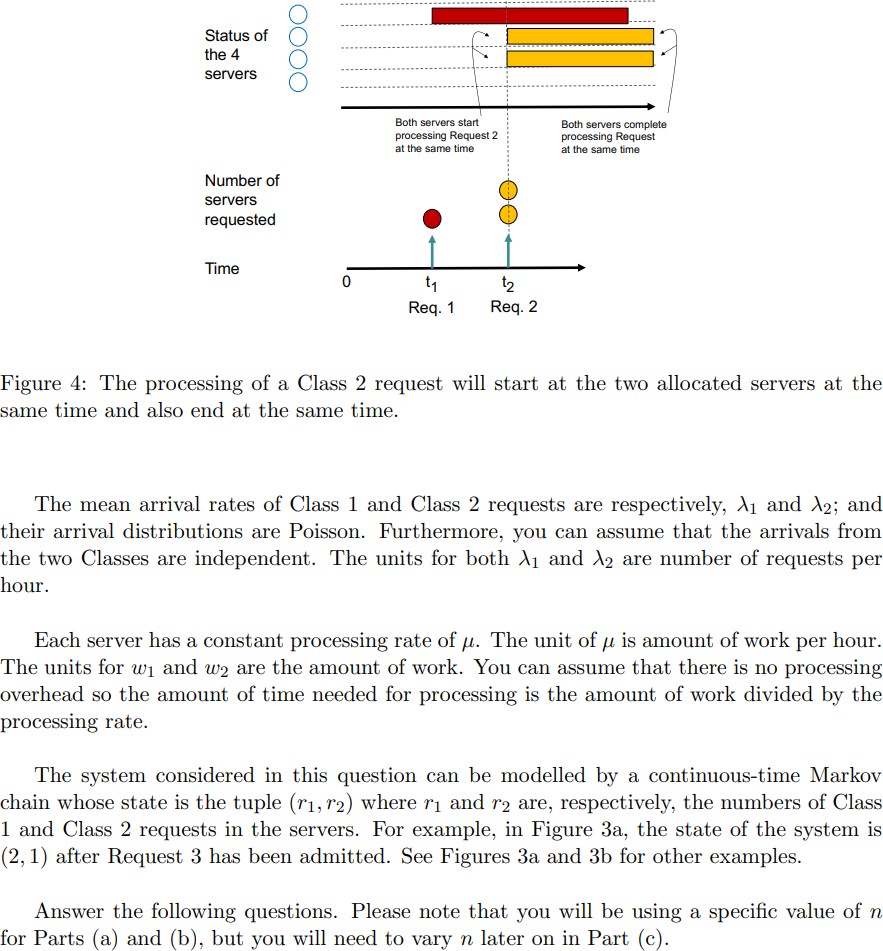
= 0.07020\*0.4 + 0.06011 \* 0.6

= 0.064146 (hour)

## Question 3:







**The Following Question uses a Python Program for calculation.**

### *a) Assuming that* 𝒏 *= 4, formulate a continuous-time Markov chain for the system using the* state definition given earlier. You can answer this question by drawing a state transition diagram with all the states and transitions. You can express the transition rates in terms of

𝝀***1,*** 𝝀***2,*** 𝒘***1,*** 𝒘***2 and*** µ*.*

w1: Workload of Class 1 query. w2: Workload of Class 2 query.

µ: The Constant processing rate of a server (Hourly).

𝑤1 & 𝑤2 ∶ The time that used to Complete the workload.

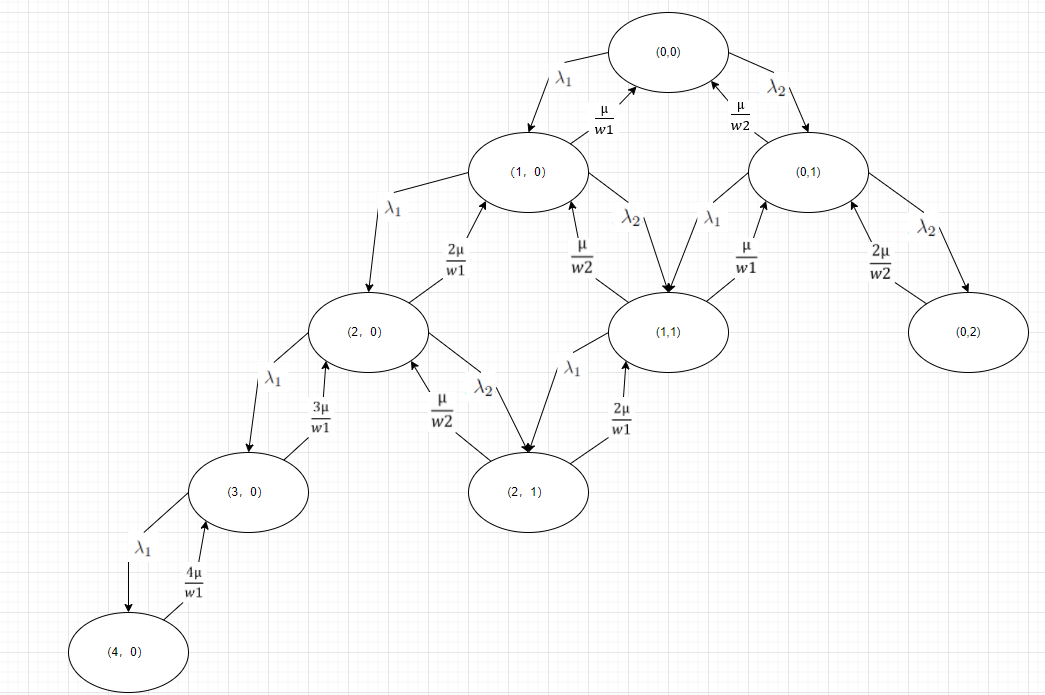
µ µ

µ & µ ∶ 1 (ℎ𝑟) = µ is the processing rate work workload per hour.

𝑤1

𝑤2 𝑤 𝑤

µ



* There are 4 Servers, which means this model is M/M/m.
* We treat Class 1 Request as a one-person team and Class 2 Request as a Two-people team. So, when a Request finishes, we return a probability of a Class team finish.
* When State (2,1) to (1,1). There are two Class 1 Requests, and we need to consider the probability of either one of the Class 1 requests finishing. So, we need to µ/w1 \* 2.

### *b) Assuming that n = 4,* 𝝀 *1 = 2.7,* 𝝀 *2 = 1.5,* 𝒘*1 = 10.4,* 𝒘*2 = 15.3 and* µ *= 70. Answer the* following questions.

1. What are the steady state probabilities of the states for the continuous-time Markov chain?

(𝜆1 + 𝜆2) ∗ 𝑷(𝟎, 𝟎) −

µ

𝑤1

∗ 𝑷(𝟏, 𝟎) −

µ

𝑤2

∗ 𝑷(𝟎, 𝟏) + 0 ∗ 𝑷(𝟐, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟏) + 0 ∗ 𝑷(𝟎, 𝟐) + 0 ∗ 𝑷(𝟑, 𝟎) + 0 ∗ 𝑷(𝟐, 𝟏) + 0 ∗ 𝑷(𝟒, 𝟎) = 𝟎

−𝜆1 ∗ 𝑷(𝟎, 𝟎) + (

µ

𝑤1

+ 𝜆1 + 𝜆2) ∗ 𝑷(𝟏, 𝟎) + 0 ∗ 𝑷(𝟎, 𝟏) −

2µ

𝑤1

∗ 𝑷(𝟐, 𝟎) −

µ

𝑤2

∗ 𝑷(𝟏, 𝟏) + 0 ∗ 𝑷(𝟎, 𝟐) + 0 ∗ 𝑷(𝟑, 𝟎) + 0 ∗ 𝑷(𝟐, 𝟏) + 0 ∗ 𝑷(𝟒, 𝟎) = 𝟎

µ

−𝜆2 ∗ 𝑷(𝟎, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟎) + (

µ

+ 𝜆1 + 𝜆2) ∗ 𝑷(𝟎, 𝟏) + 0 ∗ 𝑷(𝟐, 𝟎) −

2µ

∗ 𝑷(𝟏, 𝟏) −

∗ 𝑷(𝟎, 𝟐) + 0 ∗ 𝑷(𝟑, 𝟎) + 0 ∗ 𝑷(𝟐, 𝟏) + 0 ∗ 𝑷(𝟒, 𝟎) = 𝟎

𝑤2

𝑤1

𝑤2

2µ

0 ∗ 𝑷(𝟎, 𝟎) − 𝜆1 ∗ 𝑷(𝟏, 𝟎) + 0 ∗ 𝑷(𝟎, 𝟏) + (

3µ

+ 𝜆1 + 𝜆2) ∗ 𝑷(𝟐, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟏) + 0 ∗ 𝑷(𝟎, 𝟐) − (

µ

) ∗ 𝑷(𝟑, 𝟎) − (

) ∗ 𝑷(𝟐, 𝟏) + 0 ∗ 𝑷(𝟒, 𝟎) = 𝟎

𝑤1

𝑤1

𝑤2

µ

0 ∗ 𝑷(𝟎, 𝟎) − 𝜆2 ∗ 𝑷(𝟏, 𝟎) − 𝜆1 ∗ 𝑷(𝟎, 𝟏) + 0 ∗ 𝑷(𝟐, 𝟎) + (

µ 2µ

+ + 𝜆1) ∗ 𝑷(𝟏, 𝟏) + 0 ∗ 𝑷(𝟎, 𝟐) + 0 ∗ 𝑷(𝟑, 𝟎) − (

) ∗ 𝑷(𝟐, 𝟏) + 0 ∗ 𝑷(𝟒, 𝟎) = 𝟎

𝑤2

𝑤1

𝑤1

0 ∗ 𝑷(𝟎, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟎) − 𝜆2 ∗ 𝑷(𝟎, 𝟏) + 0 ∗ 𝑷(𝟐, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟏) + (

2µ

𝑤2

) ∗ 𝑷(𝟎, 𝟐) + 0 ∗ 𝑷(𝟑, 𝟎) + 0 ∗ 𝑷(𝟐, 𝟏) + 0 ∗ 𝑷(𝟒, 𝟎) = 𝟎

0 ∗ 𝑷(𝟎, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟎) + 0 ∗ 𝑷(𝟎, 𝟏) − 𝜆1 ∗ 𝑷(𝟐, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟏) + 0 ∗ 𝑷(𝟎, 𝟐) + (𝜆1 +

3µ 4µ

) ∗ 𝑷(𝟑, 𝟎) + 0 ∗ 𝑷(𝟐, 𝟏) − (

) ∗ 𝑷(𝟒, 𝟎) = 𝟎

𝑤1 𝑤1

µ

0 ∗ 𝑷(𝟎, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟎) + 0 ∗ 𝑷(𝟎, 𝟏) − 𝜆2 ∗ 𝑷(𝟐, 𝟎) − 𝜆1 ∗ 𝑷(𝟏, 𝟏) + 0 ∗ 𝑷(𝟎, 𝟐) + 0 ∗ 𝑷(𝟑, 𝟎) + (

2µ

+ ) ∗ 𝑷(𝟐, 𝟏) + 0 ∗ 𝑷(𝟒, 𝟎) = 𝟎

𝑤2 𝑤1

4µ

0 ∗ 𝑷(𝟎, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟎) + 0 ∗ 𝑷(𝟎, 𝟏) + 0 ∗ 𝑷(𝟐, 𝟎) + 0 ∗ 𝑷(𝟏, 𝟏) + 0 ∗ 𝑷(𝟎, 𝟐) − 𝜆1 ∗ 𝑷(𝟑, 𝟎) + 0 ∗ 𝑷(𝟐, 𝟏) + (

) ∗ 𝑷(𝟒, 𝟎) = 𝟎

𝑤1

|  |  |
| --- | --- |
| P (0,0): 0.4918992967968496  P (1,0): 0.19732188934365058  P (0,1): 0.16127269802125266  P (2,0): 0.03957713323406925  P (1,1): 0.06469339086338233 | P (0,2): 0.026437202997055195  P (3,0): 0.005292028101012529  P (2,1): 0.012975645824598293  P (4,0): 0.0005307148181300137 |

Please Check “Question3.py” and look for Q3\_b\_i

### *Determine the probability that an arriving Class 1 request will be rejected.*

This Situation only happens when all servers are occupied.

P [Class 1 will be rejected]: = P(4,0) + P(2,1) + P(0,2)

= 0.039943563639783505

Please Check “Question3.py” and look for Q3\_b\_ii

### *Determine the probability that an arriving Class 2 request will be rejected.*

This Situation only happens when less than 1 (Include 1) server is busy.

P [Class 2 will be rejected]: = P(1,1) + P(0,2) + P(3,0) + P(2,1) + P(4,0)

= 0.10992898260417834

Please Check “Question3.py” and look for Q3\_b\_iii

## Determine the probability that an arriving request will be rejected. Note that the hint in Question 2 is applicable.

In order to calculate the Arriving request will be rejected. I can calculate probability of Class 1 income Request (λ 1) of an overall income Request (λ 1 + λ 2). The we can use the it calculate the probability of Class 1 arriving request will be rejected, same method apply to Class 2.

P [Class 1 Arriving Request rate] =

λ 1 =

λ 1 + λ 2

2.7

2.7+1.5

= 0.642857

P [Arriving Request of Class 1 will be rejected] =

2.7

2.7+1.5

∗ (P(4,0) + P(2,1) + P(0,2))

P [Arriving Request of Class 2 will be rejected] = (1 − 2.7

2.7+1.5

) ∗ (P(1,1) + P(0,2) + P(3,0) + P(2,1) + P(4,0))

P [Arriving Request will be rejected] =

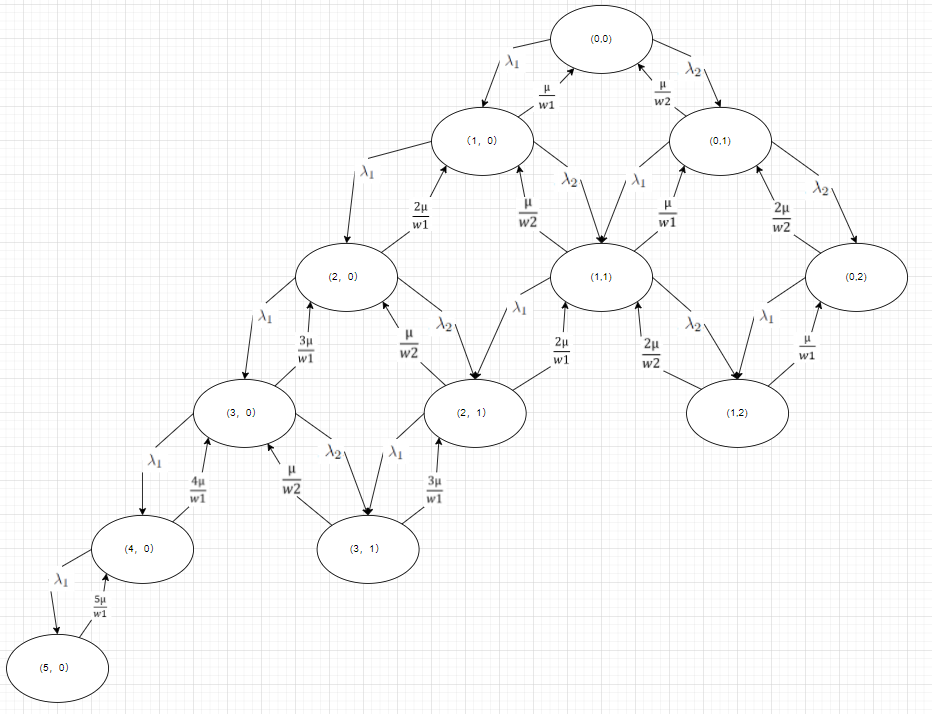
P [Arriving Request of Class 1 will be rejected] + P [Arriving Request of Class 2 will be rejected]

= 0.06493835612706737

Please Check “Question3.py” and look for Q3\_b\_iv

## Assuming that 𝝀 1 = 2,7, 𝝀 2 = 1,5, w1 = 10.4, w2 = 15.3 and µ = 70. What is the smallest value of n that can reduce the probability of rejecting an arriving request to a level lower than 0.05?

The overall request is being rejected is 0.0649 when there are 4 servers which are slightly higher than 0.05. We increase the number of servers to 5.



|  |  |
| --- | --- |
| P (0,0): 0.48588275502151895  P (1,0): 0.19490839658577508  P (0,1): 0.1593001318249121  P (2,0): 0.03909305554377544  P (1,1): 0.06390211002347906  P (0,2): 0.026113843038440866 | P (3,0): 0.005227299998424833  P (2,1): 0.012816937496137826  P (1,2): 0.010475381607420349  P (4,0): 0.0005242235141277194  P (3,1): 0.0017138076423407625  P (5,0): 4.20577036477519e-05 |

P (Class 1 will be Rejected) = 0.012231246953408863 P (Class 2 will be Rejected) = 0.051686251002115276

P [Class 1 Income Request] =

λ 1 =

λ 1 + λ 2

2.7

2.7+1.5

= 0.642857

P [Arriving Request of Class 1 will be rejected] = P (Class 1 will be Rejected) \* P [Class 1 Income Request]

P [Arriving Request of Class 2 will be rejected] = P (Class 2 will be Rejected) \* (1 - P [Class 1 Income Request]) P [Arriving Request will be rejected]

= P [Arriving Request of Class 1 will be rejected] + P [Arriving Request of Class 2 will be rejected]

= 0.026322319827946868

So, we can conclude that when number of Server is 5, the probability of request will be rejected less than 0.05.

Please Check “Question3.py” and look for Q3\_c