|  |  |
| --- | --- |
|  |  |
| Performance Evaluation Mini-Project |  |
|  |  |
|  | Multi-server Model call center Simulation |
|  | Mohamed Saber – 4245Mazen El-Lithy - 5167Karim El-Sayad - 6023 |

**Requirements:**

1. calls arrive according to a Poisson process with a mean rate of 220 call/hr.
   1. Poisson process 🡪 exponential inter-arrival time.
   2. Inter-Arrival time in minutes = =

**Graphical user interface, text, application, email

Description automatically generated**

1. The mean service capacity of the call center is 230 call/hr.
   1. Service Time in minutes () = =
   2. Using Simulink server block with exponential service time

Graphical user interface, text, application

Description automatically generated

1. Attributes :
   1. ClassA [1 , 2]
   2. Impatient [1 , 2]
   3. StartTime = 0
   4. Exit = 2
2. Use prioritized routing, where the customers are divided into two classes. The higher priority class comprises 20% of the arriving customers

% Pattern: Arbitrary discrete distribution

% V: Value vector

% P: Probability vector

V = [1 2];

P = [0.2 0.8];

entity.ClassA = randsample(V, 1, true, P);

**Question 1:**

**Diagram

Description automatically generated**

1. 10% of the arriving customers are impatient.

% Pattern: Arbitrary discrete distribution

% V: Value vector

% P: Probability vector

V = [1 2];

P = [0.1 0.9];

entity.Impatient = randsample(V, 1, true, P);

1. who abandon the system if their waiting time exceeds 3 minutes?
   * 1. we need to set a timestamp to the entities to check when the generation time was. A function called getCurrentTime() was created using digital clock block to read the current simulation time.

**Diagram

Description automatically generated**

entity. StartTime = getCurrentTime();

* + 1. Once the Entity enters the server if the difference between the current time and the entity time is greater than 3 mins then the Exit of the entity will be equal to 1 which will be routed to entity terminator 1 and if the condition is false, it will be routed to entity terminator 2

**Graphical user interface, text, application

Description automatically generated**

1. **Graphical user interface, text, application

   Description automatically generatedThe queue used is priority queue with infinite capacity and the priority source is ClassA if ClassA is 1 it will be out first.**

**Graphical user interface, text, application

Description automatically generated**

Output:

Simulation time = 100,000 unit time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirements** | **M=1** | **M=2** | **M=3** | **M=4** |
|  | 18.81 | 0.28 | 0.04 | 0.01 |
|  | 5.77 mins | 0.499 mins | 0.3827 mins | 0.3427 mins |
| **Abandonment rate** | 5.22% | 0% | 0% | 0% |

**Question 2:**

**Diagram

Description automatically generated**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirements** | **M=1** | **M=2** | **M=3** | **M=4** |
|  | 18.85 | 0.28 | 0.03845 | 0.005662 |
|  | 5.61 mins | 0.498 mins | 0.3824 mins | 0.3425 mins |

**Question 3:**

Using an M/M/∞ model, to estimate the number of agents (m) required to keep the probability that a customer must wait does not exceed 0.05. So <=0.05

The function needed to be solved first to obtain the optimum number of servers required to reduce the blocking probability to 0.05. The function below was used and the optimal number of servers that should be used is 4.

Diagram

Description automatically generated

%-------------------------------------------------------------------------%

% SOLVING BLOCKING PROBABILITY EQUATION %

%-------------------------------------------------------------------------%

clc;

%Declare variables

syms numberOfServers x S

Lambda = 220;

Mu = 230;

%Efficiency or utilization

Rhohat = Lambda/Mu;

fprintf("-----------------------------------\n");

fprintf("| Servers | Blocking Probability |\n");

fprintf("-----------------------------------\n");

%For loop to increment number of servers till the Ec(numberOfServers,Rhohat) < 0.05

for numberOfServers = 1:inf

Rho = Rhohat / numberOfServers;

Numerator = (Rhohat.^numberOfServers)/factorial(numberOfServers);

Sigma = symsum(Rhohat.^x/factorial(x),x,0,numberOfServers-1);

Denominator = Numerator + (1-Rho)\*Sigma;

finalValue = Numerator / Denominator;

fprintf("| %d | %.2f |\n",numberOfServers,finalValue);

fprintf("-----------------------------------\n");

if finalValue < 0.05

break

end

end

Table

Description automatically generated