Typical Solution for Project No. 3 Part I

ECE 642
Dr. Bijan Jabbari

I Project description: Simulation of a Finite M/M/1 Queue

In this project we simulate a finite M/M/1 queueing system using MATLAB. Consider the problem in Project 2 and assume the queue to have a maximum capacity of 20 packets ($N = 20$). All other system parameters are as in that Project. That is Poisson arrival and exponential service time, the link with a capacity of 155 Mbps, the length of packets are exponentially distributed with mean packet size of 2325 bits. We want to simulate the system for the desired utilizations.

II Project requirements:

Simulate the system for the desired utilizations of $\rho = 0.3, 0.50, 0.9, 0.95$ and obtain Probability of blocking, expected delay, and expected waiting time vs. throughput. In the same graph plot the analytical results.

III Introduction

The Finite M/M/1 queue has a single server and the queue has a maximum capacity of N packets. For this project N=20 packets. The Finite M/M/1 queue is simulated for $\rho = [0.3, 0.5, 0.9, 0.95]$. The capacity of the link is specified as 155 Mbps and the length of packets are exponentially distributed with mean packet size 2325 bits. This gives the service rate, $\mu = 155000000/2325$. Hence, we can calculate the arrival rate λ, which is equal to $\mu * \rho$. The inter-arrival and service times are exponentially distributed.

The random number generator function, rand, is used to generate uniform random numbers and with transformation to provide the exponentially distributed inter-arrival and service times:

$$inter - arrival = (-1/\lambda) * log(rand); \tag{1}$$

$$service - time = (-1/\mu) * log(rand); \tag{2}$$

The queue in Finite M/M/1 is limited to 20 packets and so all arrivals after queue size has reached 20 packets will be blocked i.e.those arrivals are not served and are dropped right away. To make it seed-independent, the simulation should be repeated on multiple seeds.

IV Analytical results:

- Blocking probability vs. normalized throughput

  For $\rho = [0.3, 0.5, 0.9, 0.95]$, the analytical values of Blocking Probability is given by:

  $$P_B = P_N = \rho^N * \frac{1-\rho}{1-\rho^{N+1}}$$

  where $N = 20$.

- Expected waiting time vs. normalized throughput

  The Expected Waiting time is given by:

$$E[W] = E[T] - \frac{1}{\mu}$$

- Expected system delay vs. normalized throughput

  The Expected delay is given by:

$$E[T] = \frac{E[N]}{\lambda * (1 - P_B)}$$

The Expected queue size, $E[N]$ is given by:

$$E[N] = \sum_{n=0}^{N} n.P_n$$

So, we can get $E[T]$ like following:

$$E[T] = \frac{1}{\lambda * (1 - P_B)} (\sum_{n=0}^{N} n.P_n)$$

where,

$$P_n = \rho^n * \frac{1 - \rho}{1 - \rho^{N+1}}$$

.

V Programm list

```
function [Pb,E_wait,E_delay] =
delaythroughput(rho,seed1,seed2,NumberOfSamples);

BufferSize = 20;
E_service = 2325/(155*10^3); % average service time
E_interarr = E_service/rho; % average interarrival time

%generate two vector of random numbers with uniformly distribution
rand('state',seed1); u1 = rand(NumberOfSamples,1);
rand('state',seed2); u2 = rand(NumberOfSamples,1);

%obtain inter-arrival time random numbers and service time random numbers
x = -E_service*log(u1);
tau = -E_interarr*log(u2);

%Initialization
t(1) = tau(1);
s(1) = t(1)+x(1);
w(1) = 0;
T(1) = x(1);
```

```
new_iterator = 2; %the iterator for vector of depart only for non_blocked customers
check_iterator = 1; %the iterator for check the current depart_time
q=0; %current buffer size
start=2; %start point for every check round
block=0;

for k=2:NumberOfSamples
    t(k) = t(k-1) + tau(k);
end;

while (start < NumberOfSamples-1)
    for i=start:NumberOfSamples
        if(s(check_iterator)>t(i))
            if(q < BufferSize)
                s(new_iterator) = s(new_iterator-1) + x(i);
                w(new_iterator) = s(new_iterator) - t(i);
                T(new_iterator) = w(new_iterator) + x(i);
                new_iterator = new_iterator+1;
                q = q+1;
            else
                block = block+1;
            end;
            if(i==NumberOfSamples)
                start = NumberOfSamples;
                break;
            end;
        else
            if(q==0) %the system will be empty after current departure
                s(new_iterator) = t(i) + x(i);
                w(new_iterator) = 0;
                T(new_iterator) = x(i);
                new_iterator = new_iterator+1;
                start = i + 1;
            else
                q=q-1;
                start = i;
            end;
            check_iterator = check_iterator+1;
            break;
        end;
    end;
end;

Pb = block/NumberOfSamples;
E_wait = sum(w)/(new_iterator-1);
E_delay = sum(T)/(new_iterator-1);
```

```
%the main file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Project 3%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%In this project we simulate a finite M/M/1 queueing system using MATLAB.
%Consider the problem in Project 2 and assume the queue to have a maximum
%capacity of 20 packets (N=20). All other system parameters are as in that
%Project. That is Poisson arrival and exponential service time. Simulate the
%system for the desired utilization of rho=0.3,0.5,0.9.0.95 and obtain Pb,
%expected delay, and expected waiting time vs. throughput. In the same graph
%plot the analytical results.

NumberOfSamples = 50000;

% for rho=0.3
[Pb1,E_wait1,E_delay1] = delaythroughput(0.3,9,10,NumberOfSamples);
[Pb2,E_wait2,E_delay2] = delaythroughput(0.3,19,20,NumberOfSamples);
[Pb3,E_wait3,E_delay3] = delaythroughput(0.3,29,30,NumberOfSamples);
[Pb4,E_wait4,E_delay4] = delaythroughput(0.3,44,45,NumberOfSamples);
[Pb5,E_wait5,E_delay5] = delaythroughput(0.3,56,57,NumberOfSamples);

% for rho =0.5
[Pb6,E_wait6,E_delay6] = delaythroughput(0.5,9,10,NumberOfSamples);
[Pb7,E_wait7,E_delay7] = delaythroughput(0.5,19,20,NumberOfSamples);
[Pb8,E_wait8,E_delay8] = delaythroughput(0.5,29,30,NumberOfSamples);
[Pb9,E_wait9,E_delay9] = delaythroughput(0.5,44,45,NumberOfSamples);
[Pb10,E_wait10,E_delay10] = delaythroughput(0.5,56,57,NumberOfSamples);


% for rho=0.9
[Pb11,E_wait11,E_delay11] = delaythroughput(0.9,9,10,NumberOfSamples);
[Pb12,E_wait12,E_delay12] = delaythroughput(0.9,19,20,NumberOfSamples);
[Pb13,E_wait13,E_delay13] = delaythroughput(0.9,29,30,NumberOfSamples);
[Pb14,E_wait14,E_delay14] = delaythroughput(0.9,44,45,NumberOfSamples);
[Pb15,E_wait15,E_delay15] = delaythroughput(0.9,56,57,NumberOfSamples);

%for rho=0.95
[Pb16,E_wait16,E_delay16] = delaythroughput(0.95,9,10,NumberOfSamples);
[Pb17,E_wait17,E_delay17] = delaythroughput(0.95,19,20,NumberOfSamples);
[Pb18,E_wait18,E_delay18] = delaythroughput(0.95,29,30,NumberOfSamples);
[Pb19,E_wait19,E_delay19] = delaythroughput(0.95,44,45,NumberOfSamples);
[Pb20,E_wait20,E_delay20] = delaythroughput(0.95,56,57,NumberOfSamples);


%simulation results
%get average blocking probability
ave_pb(1) = (Pb1+Pb2+Pb3+Pb4+Pb5)/5;
ave_pb(2) = (Pb6+Pb7+Pb8+Pb9+Pb10)/5;
ave_pb(3) = (Pb11+Pb12+Pb13+Pb14+Pb15)/5;
ave_pb(4) = (Pb16+Pb17+Pb18+Pb19+Pb20)/5;

%get average waiting time
ave_wait(1) = (E_wait1+E_wait2+E_wait3+E_wait4+E_wait5)/5;
ave_wait(2) = (E_wait6+E_wait7+E_wait8+E_wait9+E_wait10)/5;
ave_wait(3) = (E_wait11+E_wait12+E_wait13+E_wait14+E_wait15)/5;
ave_wait(4) = (E_wait16+E_wait17+E_wait18+E_wait19+E_wait20)/5;

%get average system delay
ave_delay(1) = (E_delay1+E_delay2+E_delay3+E_delay4+E_delay5)/5;
ave_delay(2) = (E_delay6+E_delay7+E_delay8+E_delay9+E_delay10)/5;
ave_delay(3) = (E_delay11+E_delay12+E_delay13+E_delay14+E_delay15)/5;
ave_delay(4) = (E_delay16+E_delay17+E_delay18+E_delay19+E_delay20)/5;

%get throughput
x(1) = 0.3*(1-ave_pb(1));
x(2) = 0.5*(1-ave_pb(2));
x(3) = 0.9*(1-ave_pb(3));
x(4) = 0.95*(1-ave_pb(4));

%analytical results
%normalized throughput = rho*(1-Pb)
ro=linspace(0,.99,100); throughput = ro.*(1-ro.^20)./(1-ro.^21);
pb = ((ro.^(20)).*(1-ro))./(1-ro.^21);
%average system delay and average waiting time
lambda = ro.*155*1000/2325; E_t = ro;
for n=2:20
    E_t = E_t + n.*ro.^n;
end;
E_t = E_t.*(1-ro)./(1-ro.^21)./(lambda.*(1-pb));
E_w = E_t - 2325/(155*10^3);

%plot the graph
%plot blocking probability vs. throughput
figure(1); plot(x,ave_pb,'ro:',throughput,pb,'r');
xlabel('Normalized throughput');
ylabel('Average blocking probability');
legend('Simulated-result','analysis-result');
title('Blocking probability vs. normalized throughput');

%plot waiting time vs. throughput
figure(2); plot(x,ave_wait,'ro:',throughput,E_w,'r');
xlabel('Normalized throughput');
ylabel('Average waiting time (msec)');
legend('Simulated-result','analysis-result');
title('Average waiting time vs. normalized throughput');
```

```
%plot system delay vs. throughput
figure(3); plot(x,ave_delay,'ro:',throughput,E_t,'r');
xlabel('Normalized throughput');
ylabel('Average delay time (msec)');
legend('Simulated-result','analysis-result');
title('Average system delay vs. normalized throughput');
```

VI Simulation results:

The simulated values are plotted on the same graph as the analytical values. The simulated values are denoted by $'o'$ and the analytical values are denoted by a solid line.

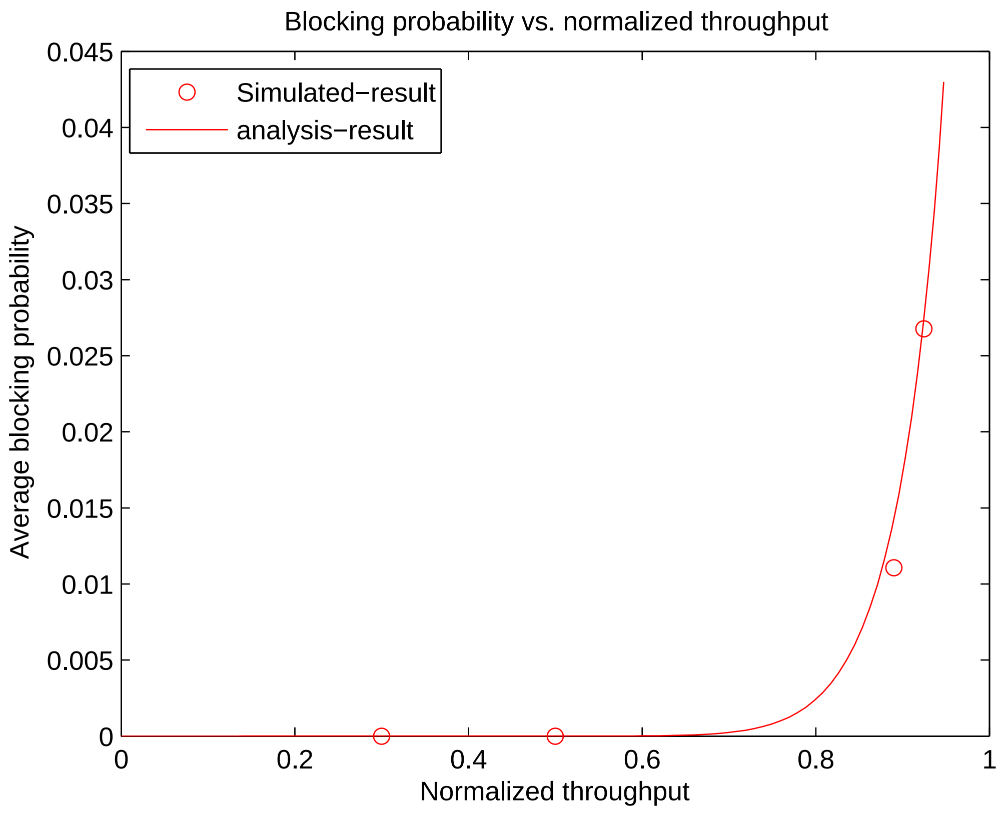

Figure 1: Blocking probability and normalized throughput for a Finite M/M/1 model $NumberOfSamples = 50000$

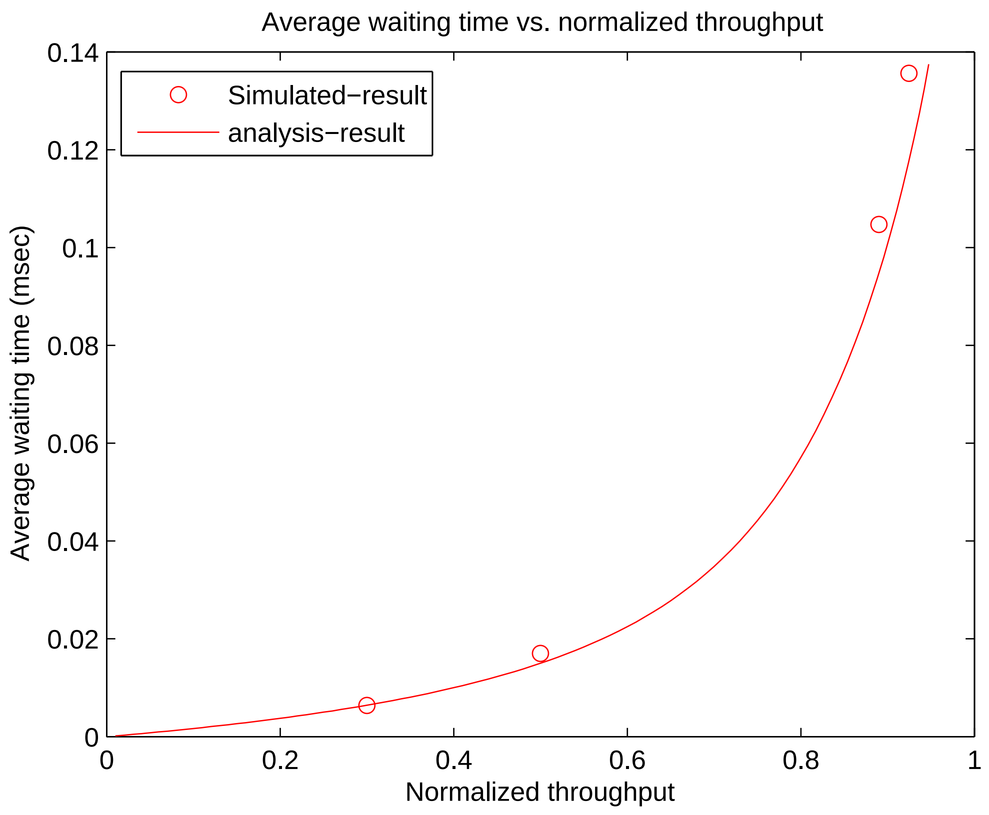

Figure 2: Expected waiting time and normalized throughput for a Finite M/M/1 model $NumberOfSamples = 50000$

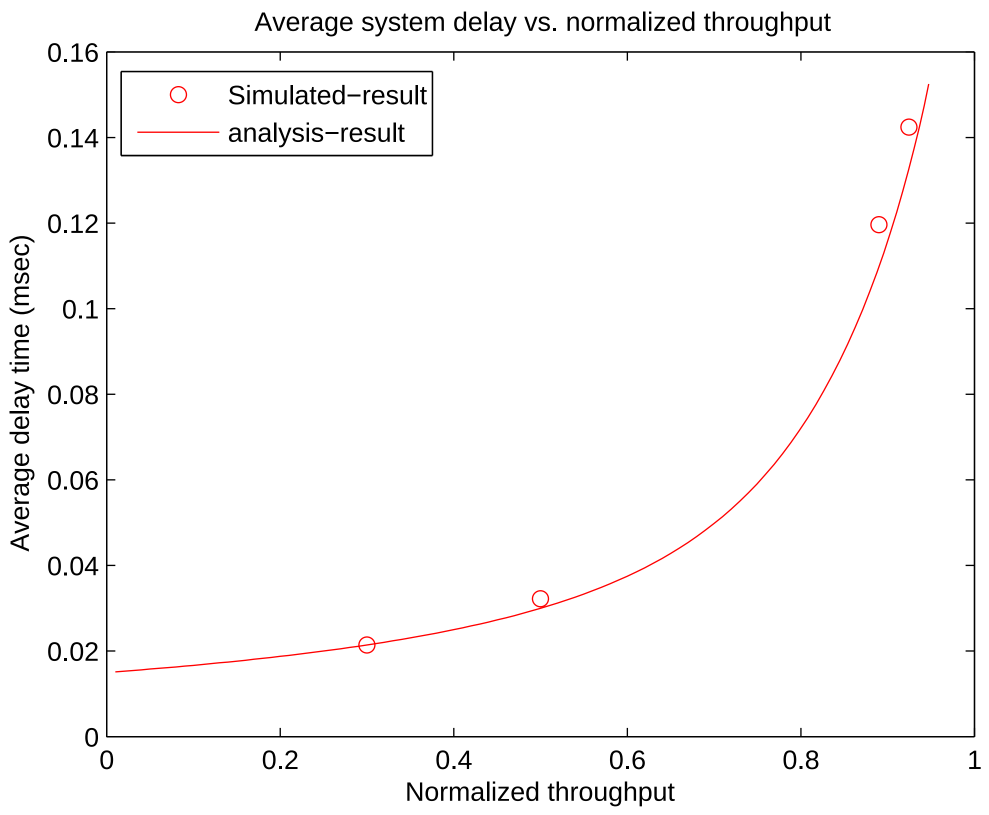

Figure 3: Expected system delay and normalized throughput for a Finite M/M/1 model $NumberOfSamples = 50000$