# Queuing Theory Exercise Series 3

Alexandros Kyriakakis (03112163)

June 2020

## 1 Network with Alternatives

### (1)

In order to form an M/M/1 queue using this type of connections, it should

- The average incoming flow $\lambda$ must be a Poisson flow
- The size of packages must be exponential distributed

So, we know that $\lambda_1 = \alpha\lambda$ and $\lambda_2 = (1-a)\lambda$ as a random disruption of a Poisson flow with average rate $\lambda$, hence $\mu_i = \frac{C_i}{E(L)}$ so,

$$\mu_1 = \frac{15 \cdot 10^6 bps}{128 \cdot 8 bits} = 14648.43Hz \ and \ \mu_2 = \frac{12 \cdot 10^6 bps}{128 \cdot 8 bits} = 11718.75Hz$$

$$\rho_1 = \frac{\lambda_1}{\mu_1} = \alpha\frac{\lambda}{\mu_1} = 0.682a < 1 \ and \ \rho_2 = \frac{\lambda_2}{\mu_2} = (1-\alpha)\frac{\lambda}{\mu_2} = 0.682(1-a) < 1$$

### (2)

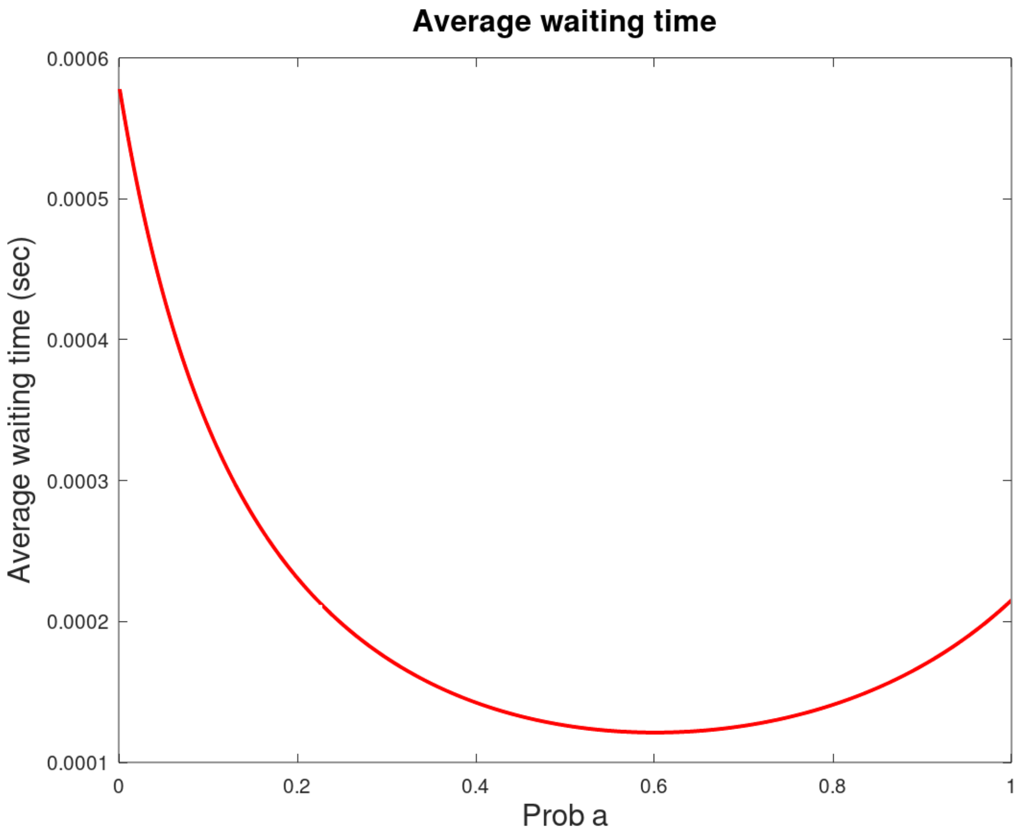
Using Jackson theorem, the average number of packages in the network is

$$E(n) = E(n_1) + E(n_2) = \frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2} = \frac{\frac{\alpha\lambda}{\mu_1}}{1-\frac{\alpha\lambda}{\mu_1}} + \frac{\frac{(1-\alpha)\lambda}{\mu_2}}{1-\frac{(1-\alpha)\lambda}{\mu_2}} = \frac{\alpha\lambda}{\mu_1 - \alpha\lambda} + \frac{(1-\alpha)\lambda}{\mu_2 - (1-\alpha)\lambda}$$

While from Little's formula we know that,

$$E(T) = \frac{E(n)}{\gamma} = \frac{E(n)}{\lambda}$$

So, using Octave we calculated, the minimum delay $E(T) = 0.00012120sec$ at $a = 0.601$




## Code

```
pkg load queueing

clc;
clear all;
close all;

breakProb = 0.001:0.001:0.999;
lambda = 10000;

mu1 = (15 * 10^6) / (128 * 8);
mu2 = (12 * 10^6) / (128 * 8);

lambda1 = breakProb.*lambda;
lambda2 = (1-breakProb).*lambda;

[U1 R1 Q1 X1 P1] = qsmm1(lambda1,mu1);
[U2 R2 Q2 X2 P2] = qsmm1(lambda2,mu2);

R = breakProb.*R1 + (1-breakProb).*R2;
```

```
figure(1);
plot(breakProb,R,'r',"linewidth",2);
title("Average waiting time","fontsize", 15);
xlabel("Prob a","fontsize", 15);
ylabel("Average waiting time (sec)","fontsize", 15);
saveas (1, "figure1.png");
[minR,position] = min(R);
display(minR);
display(position);

clc;
clear all;
close all;
exit;
```

# 2 Open Network

### (1)

In order to use Jackson theorem we should do the following assumptions,

- The arrivals must follow independent Poisson distributions
- The deaths must follow independent Exponential distributions
- The service times will customers move through the network should be memmoryless so, service time should be dependent only from the current server according to Kleinrock's Independence Assumption.
- Inside the network every separation should be stochastic.

### (1)

We know that $\rho = \frac{\lambda}{\mu}$ so,

- $Q1 : \rho_1 = \frac{\lambda_1}{\mu_1}$
- $Q2 : \rho_2 = \frac{\lambda_2 + \rho_{12}\lambda_1}{\mu_2} = \frac{\lambda_2 + \frac{2}{7}\lambda_1}{\mu_2}$
- $Q3 : \rho_3 = \frac{\rho_{13}\lambda_1}{\mu_3} = \frac{4\lambda_1}{7\mu_3}$
- $Q4 : \rho_4 = \frac{(\rho_{14} + \rho_{34}\rho_{13})\lambda_1}{\mu_4} = \frac{3\lambda_1}{7\mu_4}$
- $Q5 : \rho_5 = \frac{(\rho_{12} + \rho_{35}\rho_{13})\lambda_1 + \lambda_2}{\mu_5} = \frac{\frac{4}{7}\lambda_1 + \lambda_2}{\mu_5}$

### (2)

#### Intensities

```
function [rho,is_ergodic] = intensities(lambda,mu)
rho(1) = lambda(1)/mu(1);
rho(2) = (lambda(2) + 2*lambda(1)/7)/mu(2);
rho(3) = (4*lambda(1)/7)/mu(3);
```

```
rho(4) = (3*lambda(1)/7)/mu(4);
rho(5) = (lambda(2) + (4/7)*lambda(1))/mu(5);
is_ergodic = true;
for i=1:5
  printf('Q%d: %f\n',i,rho(i));
  is_ergodic  = is_ergodic && (rho(i) < 1)
endfor
printf("Ergodicity: %d \n",is_ergodic)
endfunction
```

### (3)

#### Mean Clients

```
function [Rho] = mean_clients(lambda,mu)
[rho,is_ergodic] = intensities(lambda,mu);
Rho = rho ./ (1-rho);
for i=1:5
  printf("Mean Clients at Q%d: %d\n",i,Rho(i))
endfor
endfunction
```

### (4)

#### Average Waiting time

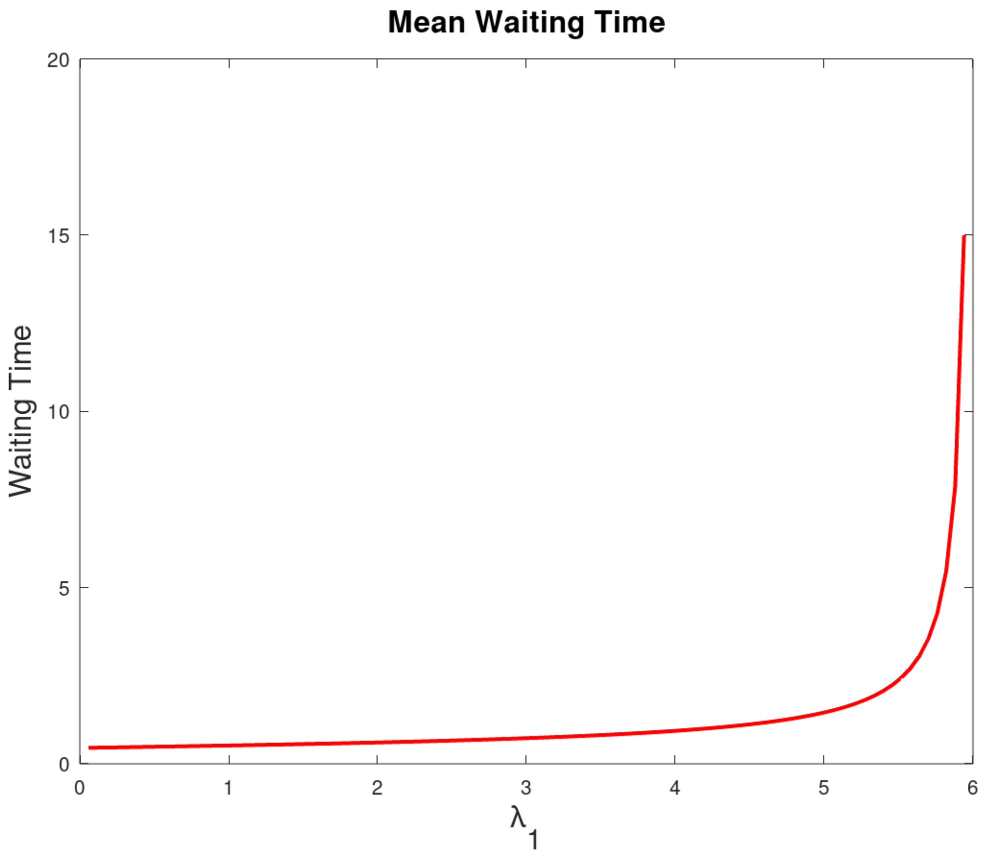
```
lambda = [4,1];
mu = [6,5,8,7,6];
Rho = mean_clients(lambda,mu);
sumation = sum(Rho)/sum(lambda);
printf("Average service time: %d", sumation);
```

### (5)

We notice that $Q_1$ has the most load intensity in the network. So, $Q_1$ creates the bottleneck. The maximum value of $\lambda_1$ is $\lambda_{1_{max}} = \rho_{1_{max}} \cdot \mu_1$, hence we know that the maximum value of $\rho$ in an ergodic network is 1, so, $\lambda_{1_{max}} = 6$.

### (6)

**Mean Waiting time Plot**




## Code

```
addpath(pwd);
% (2)
function [rho,is_ergodic] = intensities(lambda,mu)
rho(1) = lambda(1)/mu(1);
rho(2) = (lambda(2) + 2*lambda(1)/7)/mu(2);
rho(3) = (4*lambda(1)/7)/mu(3);
rho(4) = (3*lambda(1)/7)/mu(4);
rho(5) = (lambda(2) + (4/7)*lambda(1))/mu(5);
is_ergodic = true;
for i=1:5
  printf('Q%d: %f\n',i,rho(i));
  is_ergodic  = is_ergodic && (rho(i) < 1)
endfor
printf("Ergodicity: %d \n",is_ergodic)
endfunction
% (3)
function [Rho] = mean_clients(lambda,mu)
```

```
[rho,is_ergodic] = intensities(lambda,mu);
Rho = rho ./ (1-rho);
for i=1:5
  printf("Mean Clients at Q%d: %d\n",i,Rho(i))
endfor
endfunction
% (4)
l = 4;
lambda = [l,1];
mu = [6,5,8,7,6];
Rho = mean_clients(lambda,mu);
sumation = sum(Rho)/sum(lambda);
printf("Average service time: %d", sumation);

% (6)

max_lambda = 6
for i=1:99
   l = max_lambda*i/100;
   vec_lambda(i) = l;
   lambda = [l,1];
   mu = [6,5,8,7,6];
   vec_sum(i) = sum(mean_clients(lambda,mu))/sum(lambda);
endfor

figure(1);
plot(vec_lambda,vec_sum,"r","linewidth",2);
title("Mean Waiting Time","fontsize", 15);
xlabel('\lambda_1',"fontsize", 15);
ylabel("Waiting Time","fontsize", 15);
saveas (1, "figure2.png");

clc;
clear all;
close all;
exit;
%intensities([1,2],[2,5,3,4,5]);
```