ΗΜΜΥ ΕΜΠ

6^{o} εξάμηνο

Συστήματα Αναμονής

Lab 5

Ήβη Χατζή

[redacted]

Άσκηση 1: Δίκτυο με εναλλακτική δρομολόγηση

(1) Για να μπορεί να μοντελοποιηθεί ως ουρά M/M/1 πρέπει αρχικά η ροή λ να είναι Poisson, και μετά να χωριστεί τυχαία με πιθανότητες α και $1-\alpha$ σε δύο ροές Poisson $\lambda_1 = \alpha\lambda$ και $\lambda_2 = (1-\alpha)\lambda$.

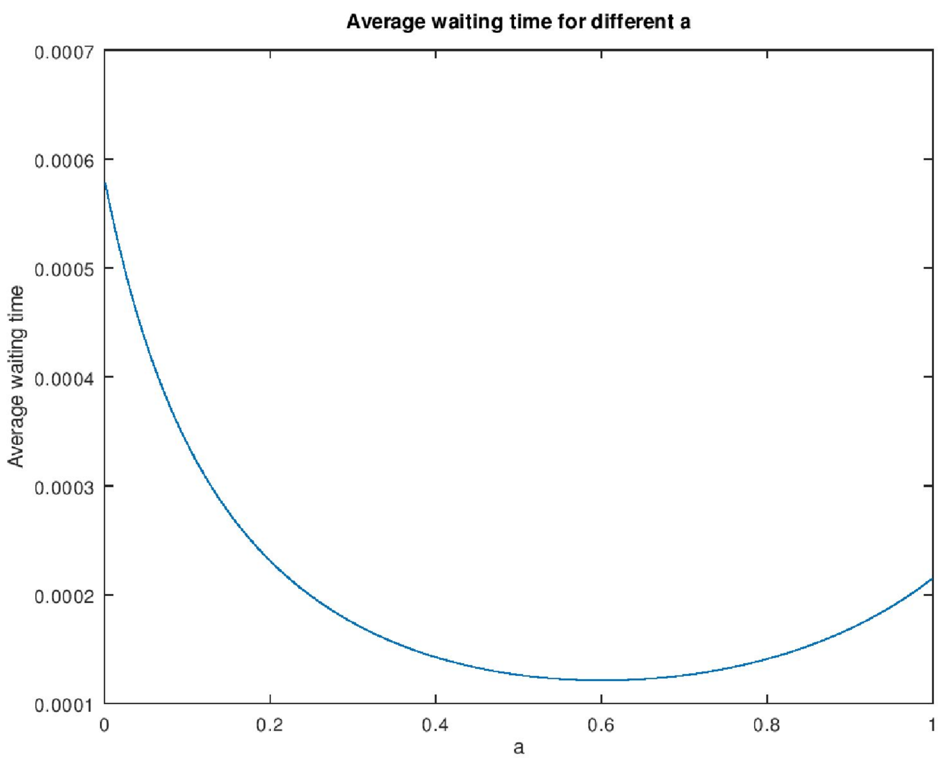
Θεωρούμε επίσης ότι οι εξυπηρετήσεις είναι ανεξάρτητες εκθετικές κατανομές και ικανοποιούν την παραδοχή ανεξαρτησίας του Kleinrock. Θεωρώντας εκθετικό μήκος πακέτων μπορούμε να βρούμε τα μ:

$\mu_1 = \frac{C_1}{E[L]} = \frac{15Mbps}{128bytes} = 14648.4$ και $\mu_2 = \frac{C_2}{E[L]} = \frac{12Mbps}{128bytes} = 11718.8$

Τέλος πρέπει οι ουρές να μην έχουν απώλειες και να λειτουργούν με πολιτική FIFO.

(2) Μέσω της qsmm1 βρίσκουμε το ζητούμενο:

```
Minimum average waiting time is 0.000121198 for a=0.601
```



Κώδικας για την άσκηση 1: αρχείο *ask1.m*

```
pkg load queueing;

clear all; close all; clc;

a=0.001:0.001:0.999;
lambda=10000;
m1=(15*10^6)/128/8;
m2=(12*10^6)/128/8;
lambda1=lambda.*a;
lambda2=lambda.*(1-a);

[U1,R1,Q1,X1]=qsmm1(lambda1,m1);
[U2,R2,Q2,X2]=qsmm1(lambda2,m2);

R=a.*R1+(1-a).*R2;
[minR,mina]=min(R);

figure(1);
plot(a,R);
title("Average waiting time for different a");
xlabel("a"); ylabel("Average waiting time");

printf("Minimum average waiting time is %d for a=%d\n",minR,a(mina));
```

Άσκηση 2: Ανοιχτό δίκτυο ουρών αναμονής

(1) Για να μελετηθεί το δίκτυο ως ανοιχτό δίκτυο με το θεώρημα Jackson πρέπει:

- Όλες οι αφίξεις λ_i να είναι ανεξάρτητες και να ακολουθούν κατανομή Poisson (ισχύει από εκφώνηση).
- Όλες οι εξυπηρετήσεις μ_i να ακολουθούν εκθετική κατανομή (ισχύει από εκφώνηση).
- Ο χρόνος εξυπηρέτησης σε έναν εξυπηρετητή είναι memoryless και εξαρτώνται μόνο από την κατανομή του εξυπηρετητή (Kleinrock's Independence Assumption).
- Τυχαία δρομολόγηση ώστε κάθε πελάτης που τελειώνει από την ουρά i πηγαίνει σε επόμενη ουρά j με πιθανότητα P_{ij} ή φεύγει από το σύστημα με πιθανότητα $1 - \sum P_{ij}$ (ισχύει βλέποντας τις πιθανότητες του σχήματος).
- Οι ουρές να είναι άπειρες και να λειτουργούν με FIFO χωρίς απώλειες.

(2) Βρίσκουμε την ένταση του φορτίου σε κάθε ουρά:

$$\rho_1 = \frac{\lambda_1}{\mu_1}$$

$$\rho_2 = \frac{\lambda_2 + \frac{2}{7}\lambda_1}{\mu_2}$$

$$\rho_3 = \frac{\frac{4}{7}\lambda_1}{\mu_3}$$

$$\rho_4 = \frac{\frac{1}{2}\frac{4}{7}\lambda_1 + \frac{1}{7}\lambda_1}{\mu_4} = \frac{\frac{3}{7}\lambda_1}{\mu_4}$$

$$\rho_5 = \frac{\frac{1}{2}\frac{4}{7}\lambda_1 + \lambda_2 + \frac{2}{7}\lambda_1}{\mu_5} = \frac{\frac{4}{7}\lambda_1 + \lambda_2}{\mu_5}$$

Η συνάρτηση intensities σε Octave. Έχω το print σε σχόλιο γιατί την χρησιμοποιώ μετά σε βρόχο και δε θέλω να τυπώνει συνέχεια.

```
function [r,ergodic]=intensities(lambda,mu)
  r(1)=lambda(1)/mu(1);
  r(2)=(lambda(2)+2/7*lambda(1))/mu(2);
  r(3)=4/7*lambda(1)/mu(3);
  r(4)=3/7*lambda(1)/mu(4);
  r(5)=(4/7*lambda(1)+lambda(2))/mu(5);
  ergodic=1;
  for i=1:5
##    printf("Intensity at queue %d: %d\n",i,r(i));
    if r(i)>=1
      ergodic=0;
    endif
  endfor
end
```

(3) Η συνάρτηση mean_clients σε Octave:

```
function [En]=mean_clients(lambda,mu)
  [r,ergodic]=intensities(lambda,mu);
  En=r./(1-r);
end
```

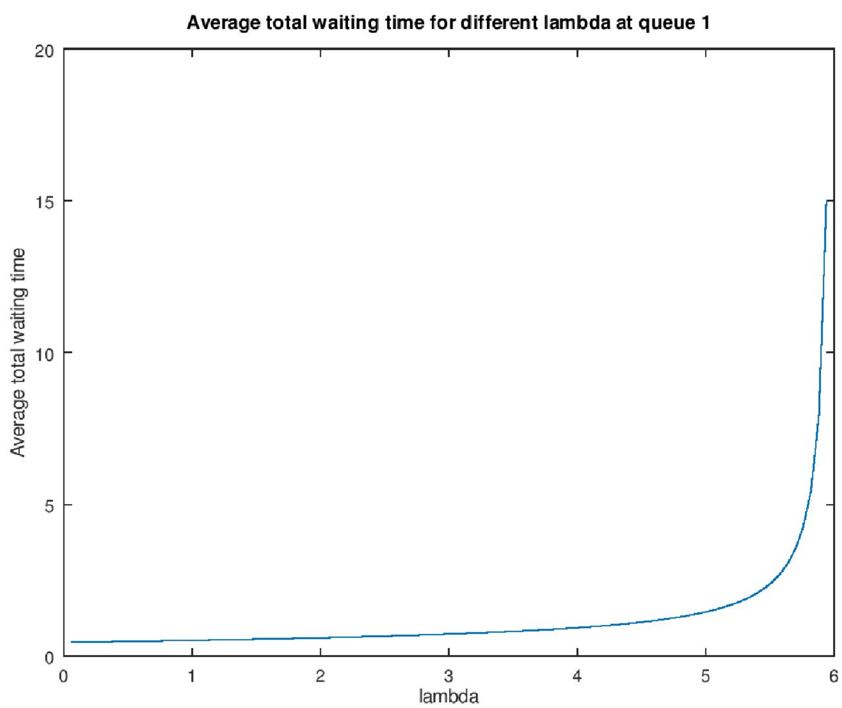
(4) Την ένταση του φορτίου σε κάθε ουρά τη βρίσκουμε μέσω της συνάρτησης intensities. Το μέσο χρόνο καθυστέρησης σε ολόκληρο το δίκτυο τον βρίσκουμε από τον τύπο του Little $E[T] = \frac{E[n]}{\gamma}$. $E[n]$ ο μέσος αριθμός πελατών στο δίκτυο, τον οποίο βρίσκουμε αθροίζοντας την mean_clients, και γ ο συνολικός μέσος εξωτερικός ρυθμός άφιξης πελατών, τον οποίο βρίσκουμε αθροίζοντας τα λ.

```
Intensity at queue 1: 0.666667
Intensity at queue 2: 0.428571
Intensity at queue 3: 0.285714
Intensity at queue 4: 0.244898
Intensity at queue 5: 0.547619

Average total waiting time: 0.93697
```

(5) Από τις συναρτήσεις μας βλέπουμε ότι η πρώτη ουρά έχει και την μεγαλύτερη ένταση φορτίου και τον μεγαλύτερο μέσο χρόνο αναμονής, οπότε αυτή είναι το bottleneck. Η μέγιστη τιμή που μπορούμε να δώσουμε στο λ_1 ώστε το σύστημα να είναι εργοδικό είναι 6, αφού πρέπει $\rho_1 = \frac{\lambda_1}{\mu_1} < 1 \;\Rightarrow\; \lambda_1 < \mu_1 = 6$.

(6)



Κώδικας για την άσκηση 2: αρχείο *ask2.m*

```
lambda=[4,1];
mu=[6,5,8,7,6];

function [r,ergodic]=intensities(lambda,mu)
  r(1)=lambda(1)/mu(1);
  r(2)=(lambda(2)+2/7*lambda(1))/mu(2);
  r(3)=4/7*lambda(1)/mu(3);
  r(4)=3/7*lambda(1)/mu(4);
  r(5)=(4/7*lambda(1)+lambda(2))/mu(5);
  ergodic=1;
  for i=1:5
##    printf("Intensity at queue %d: %d\n",i,r(i));
    if r(i)>=1
      ergodic=0;
    endif
  endfor
end

function [En]=mean_clients(lambda,mu)
  [r,ergodic]=intensities(lambda,mu);
  En=r./(1-r);
end

[ro,erg]=intensities(lambda,mu);
if erg==1
  printf("The system is ergodic\n");
else
  printf("The system is not ergodic\n");
end
printf("\n");

for i=1:5
  printf("Intensity at queue %d: %d\n",i,ro(i));
end
printf("\n");

En=mean_clients(lambda,mu);
for i=1:5
  printf("Mean clients at queue %d: %d\n",i,En(i));
end
printf("\n");

avgTotalTime=sum(En)/sum(lambda);
printf("Average total waiting time: %d\n",avgTotalTime);

lambda_max=6;
for i=1:99
  lmd(i)=lambda_max*i/100;
  lambda=[lmd(i),1];
  wait(i)=sum(mean_clients(lambda,mu))/sum(lambda);
```

```
endfor

figure(1);
plot(lmd,wait);
title("Average total waiting time for different lambda1");
xlabel("lambda"); ylabel("Average total waiting time");
```