Ε.Μ.Π., Σχολή Η.Μ.& Μ.Υ.

Συστήματα Αναμονής (Queuing Systems) - Ακαδημαϊκό Έτος 2020-2021

5η Ομάδα Ασκήσεων

1. Δίκτυο με εναλλακτική δρομολόγηση
2. Ανοιχτό δίκτυο ουρών αναμονής

Παραδοτέα: 21-06-2021

Του προπτυχιακού φοιτητή:

Νικήτας Τσίννας, ΑΜ : 03118187

<u>Δίκτυο με εναλλακτική δρομολόγηση</u>

1)

Για να θεωρήσουμε ως ουρά M/M/1 κάθε σύνδεσμο του σχήματος θα πρέπει να αποδεχτούμε πως:

- Ο μέσος ρυθμός άφιξης πελατών θα πρέπει να ακολουθεί την κατανομή Poisson
- Οι χρόνοι εξυπηρέτησης ανατίθονται στο κάθε σύστημα σαν ανεξάρτητες τυχαίες μεταβλητές.

2)

Είναι μ1 = Ci / E(L), και επομένως

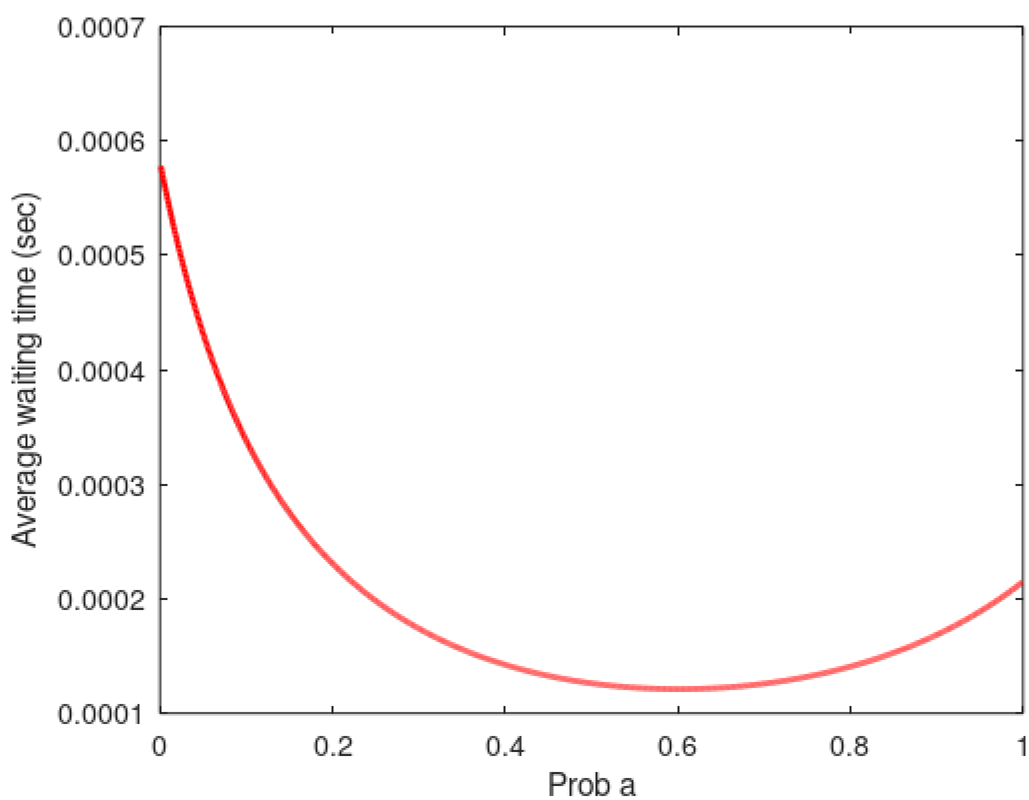
μ1 = 15*10^6 bps / 128 * 8 bits και **μ2=12*10^6 bps / 128*8 bits**

Χρησιμοποιώντας το θεώρημα του Jackson, ο μέσος αριθμός πακέτων στο δίκτυο θα είναι:

E(n) = E(n1) + E(n2) = ρ1 / (1-ρ1) + ρ2/(1-ρ2) = αλ/(μ1-αλ) + (1-α)λ / [μ2 - (1-α)λ]

Για ουρές M/M/1 ο νόμος του Little γίνεται: E(T) = E(n) / γ = E(n) / λ

Ακολουθεί το ζητούμενο διάγραμμα:



Η τιμή του α για την οποία ο χρόνος αναμονής **ελαχιστοποιείται στα 1.212*10^-4 δευτερόλεπτα** είναι **α_min = 0.601**.

Ακολουθεί ο κώδικας που χρησιμοποιήθηκε στο Octave:

```
pkg load queueing

clc;
clear all;
close all;

a = 0.001:0.001:0.999;
lamda = 10*10^3;

m1 = (15 * 10^6) / (128 * 8);
m2 = (12 * 10^6) / (128 * 8);

lamda1 = a.*lamda;
lamda2 = (1-a).*lamda;

[U1 R1 Q1 X1 P1] = qsmm1(lamda1,m1);
[U2 R2 Q2 X2 P2] = qsmm1(lamda2,m2);

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

figure(1);
plot(a,R,'r',"linewidth",2);
xlabel("Prob a");
ylabel("Average waiting time (sec)");
[minR,position] = min(R);
display(minR);
display(position*0.001);

```

Ανοιχτό δίκτυο ουρών αναμονής

1)

Για να χρησιμοποιήσουμε το θεώρημα Jackson πρέπει να δεχτούμε τις εξής παραδοχές:

- Οι χρόνοι άφιξης και ακολουθούν ανεξάρτητες κατανομές Poisson
- Οι θάνατοι ακολουθούν την εκθετική κατανομή.
- Εσωτερική δρομολόγηση (rooting) με τυχαίο τρόπο.
- Kleinrock's Independence Assumption: Οι χρόνοι εξυπηρέτησης πελατών όπως διαπερνούν το δίκτυο δεν διατηρούν την τιμή τους αλλά σε κάθε εξυπηρετητή αποκτούν νέο ανάλογα με την κατανομή του.

2)

- $\rho_1 = \lambda_1/\mu_1$
- $\rho_2 = (\lambda_2 + \rho_{12}*\lambda_1)/\mu_2 = (\lambda_2 + 2/7*\lambda_1) / \mu_2$
- $\rho_3 = \rho_{13}*\lambda_1 / \mu_3 = 4*\lambda_1/7*\mu_3$
- $\rho_4 = (\rho_{14} + \rho_{34}*\rho_{13})\lambda_1 / \mu_4 = 3*\lambda_1 / 7*\mu_4$
- $\rho_5 = [(\rho_{12}+\rho_{35}*\rho_{13})\lambda_1 + \lambda_2] / \mu_5 = [4/7*\lambda_1 + \lambda_2] / \mu_5$

```
function [rho,is_ergodic] = intensities(lamda,m)
rho(1) = lamda(1)/mu(1)
rho(2) = (lamda(2) + 2*lamda(1)/7)/m(2)
rho(3) = (4*lamda(1)/7)/m(3)
rho(4) = (3*lamda(1)/7)/m(4)
rho(5) = (lamda(2) + (4/7)*lamda(1))/m(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 [answer] = mean_clients(lamda,m)
[rho,is_ergodic] = intensities(lamda,m)
answer = rho ./ (1-rho)
for i=1:5
  printf("Mean Clients at Q%d: %d\n",i,answer(i))
endfor
endfunction
```

4)

α.Υπολογίζεται η ένταση του φορτίου που δέχεται κάθε ουρά:

```
Q1: 0.666667
is_ergodic = 1
Q2: 0.428571
is_ergodic = 1
Q3: 0.285714
is_ergodic = 1
Q4: 0.244898
is_ergodic = 1
Q5: 0.547619
is_ergodic = 1
```

Β. Υπολογίζεται ο συνολικός χρόνος καθυστέρησης του πελάτη στο σύστημα:

```
Average service time: 0.93697
>> |
```

Ακολουθεί το printout του κώδικα του συγκεκριμένου ερωτήματος:

```
lamda = [4,1]
m = [6,5,8,7,6]
Rho = mean_clients(lamda,m)
av_service_time = sum(Rho)/sum(lamda)
printf("Average service time: %d\n", av_service_time)
```

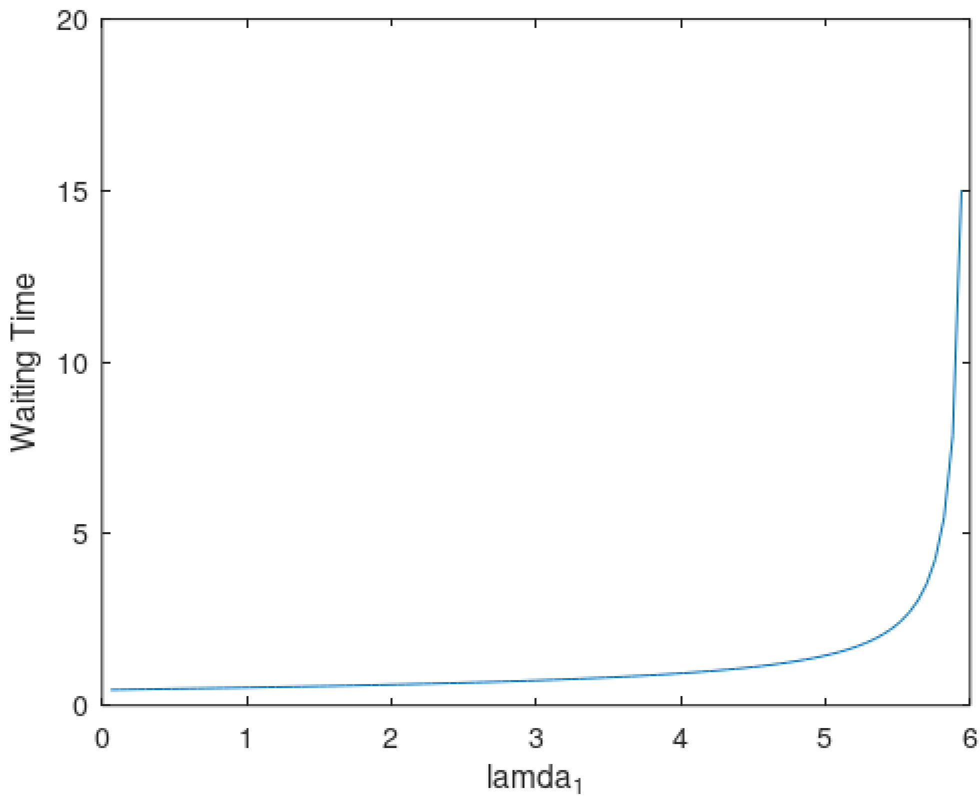
5)

Παρατηρούμε πως η ουρά Q1 έχει την μεγαλύτερη ένταση φορτίου (ρ1=0.66) με βάση τα δεδομένα που μας δόθηκαν. Επομένως η ουρά Q1 δημιουργεί το bottleneck στο δίκτυο. Για να παραμείνει το σύστημα εργοδικό θα πρέπει το ρ < 1. Επομένως ρmax = 1 => λ1max / μ1 = 1 => λ1max = μ1 ή

λ1max = 6

6)

Ακολουθεί το διάγραμμα του χρόνου αναμονής πελάτη στο σύστημα σε συνάρτηση με την τιμή λ1.



Ο κώδικας που χρησιμοποιήθηκε ακολουθεί παρακάτω:

```
close all;
clear all;


for i=1:99
    lamda = 6 * i/100
    lamda_vector(i) = lamda
    lamdas = [lamda,1]
    m = [6,5,8,7,6]
    vec_sum(i) = sum(mean_clients(lamdas,m))/sum(lamdas)
endfor

figure(1);
plot(lamda_vector,vec_sum)
xlabel('lamda_1')
ylabel("Waiting Time")
```