<u>ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ</u>

<u>Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών</u>

<u>Ονοματεπώνυμο</u>: Γκούμε Λαουρεντιάν
<u>Α.Μ.</u>: 031 18 014
<u>Έτος/Εξάμηνο</u>: 3ο/6ο

<u>ΣΥΣΤΗΜΑΤΑ ΑΝΑΜΟΝΗΣ (Queuing Systems)</u>

<u>5η Σειρά Ασκήσεων</u>

<u>Δίκτυο με εναλλακτική δρομολόγηση:</u>

(1) Προκειμένου να μπορέσουμε να μοντελοποιήσουμε τους συνδέσμους σαν M/M/1 ουρές, είναι απαραίτητο να κάνουμε τις παρακάτω παραδοχές:

- Τυχαίες, με Poisson κατανομή, εξωτερικές αφίξεις.
- Τυχαία δρομολόγηση πακέτων βάσει των πιθανοτήτων α, (1-α).
- Οι χρόνοι εξυπηρέτησης πακέτων (εκθετική κατανομή εξυπηρέτησης), καθώς αυτά διαπερνούν το δίκτυο, δε διατηρούν την τιμή τους (Μαρκοβιανή ιδιότητα έλλειψης μνήμης), αλλά αποκτούν χρόνο εξυπηρέτησης ανάλογα με την κατανομή του κάθε εξυπηρετητή (παραδοχή ανεξαρτησίας εξυπηρετητών Kleinrock).
- Άπειρες ουρές χωρίς απώλειες, με FCFS εξυπηρέτηση.

(2) Είναι λ ο ρυθμός αφίξεων στον κόμβο 1. Ο ρυθμός αφίξεων στον κόμβο 2 θα είναι τότε λ_1 = (αλ) μέσω της γραμμής 1 και λ_2 = [(1-α)λ] μέσω της γραμμής 2.

Επομένως:

- Από τη γραμμή 1: λ_{2_1} = αλ και μ_1 = $\frac{15\cdot 10^6 (bits/sec)}{8\cdot 128\, bits}$= 14.648,4375 (packs/sec). Επομένως, $E[n_1]$ = $\frac{\lambda_1}{\mu_1-\lambda_1}=\frac{\alpha\lambda}{\mu_1-\alpha\lambda}$ και $E[T_1]$ = $\frac{E[n_1]}{\lambda_1}=\frac{\left(\frac{\alpha\lambda}{\mu_1-\alpha\lambda}\right)}{\alpha\lambda}=\frac{1}{\mu_1-\alpha\lambda}$, άρα $E[T_1]$ = $\frac{1}{\mu_1-\alpha\lambda}$.

- Από τη γραμμή 2: λ_{2_2} = (1-α)λ και μ_2 = $\frac{12\cdot 10^6 (bits/sec)}{8\cdot 128\, bits}$= 11.718,75 (packs/sec). Επομένως, E[n_2] = $\frac{\lambda_2}{\mu_2-\lambda_2}=\frac{(1-\alpha)\lambda}{\mu_2-(1-\alpha)\lambda}$ και E[T_2] = $\frac{E[n_2]}{\lambda_2}=\frac{(\frac{(1-\alpha)\lambda}{\mu_2-(1-\alpha)\lambda})}{(1-\alpha)\lambda}=\frac{1}{\mu_2-(1-\alpha)\lambda}$.

Συνολικά, ο μέσος χρόνος καθυστέρησης ενός πακέτου στο σύστημα είναι: $E[T]=\frac{E[n_1]+E[n_2]}{\lambda}=$

$$\frac{(\frac{\alpha\lambda}{\mu_1-\alpha\lambda})+(\frac{(1-\alpha)\lambda}{\mu_2-(1-\alpha)\lambda})}{\lambda}=\frac{\alpha}{\mu_1-\alpha\lambda}+\frac{(1-\alpha)}{\mu_2-(1-\alpha)\lambda}\, και\, για : \mu_1=\frac{C_1}{8*128\, bits}=\frac{(15/12)C_2}{8*128\, bits}=1.25\,\mu_2$$

έχουμε:

$$E[T]=\frac{\alpha}{1.25\,\mu_2-\alpha\lambda}+\frac{(1-\alpha)}{\mu_2-(1-\alpha)\lambda}=\frac{\alpha[\mu_2-(1-\alpha)\lambda]+(1-\alpha)(1.25\,\mu_2-\alpha\lambda)}{(1.25\,\mu_2-\alpha\lambda)[\mu_2-(1-\alpha)\lambda]} \quad =$$

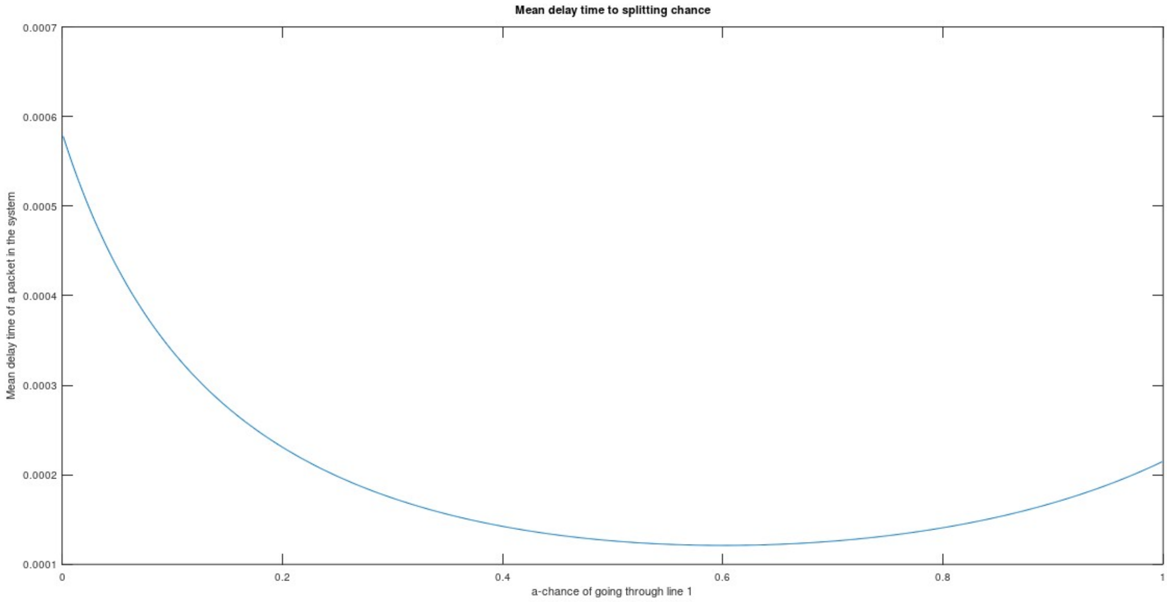
$$\frac{\mu_2\alpha-\lambda\alpha+\lambda\alpha^2+1.25\,\mu_2-\lambda\alpha-1.25\,\mu_2\alpha+\lambda\alpha^2}{1.25\,\mu_2^2-1.25\,\lambda\mu_2+1.25\,\lambda\mu_2\alpha-\lambda\mu_2\alpha+\lambda^2\alpha-\lambda^2\alpha^2}=\frac{2\lambda\alpha^2-(2\lambda+0.25\,\mu_2)\alpha+1.25\,\mu_2}{(-\lambda^2)\alpha^2+(0.25\lambda\mu_2+\lambda^2)\alpha+1.25\,\mu_2(\mu_2-\lambda)}$$

(Λάβαμε τη ζητούμενη μέση καθυστέρηση συναρτήσει όχι μόνο του α, αλλά και των παραμέτρων μ_2, λ σε περίπτωση που επιθυμούμε πληρέστερη ανάλυση).

Εκτελούμε σε Octave τη ζητούμενη προσομοίωση, οπότε και λαμβάνουμε τα εξής αποτελέσματα αναφορικά με τον ελάχιστο χρόνο καθυστέρησης E(T)_min, καθώς και για την τιμή του α για την οποία αυτός προκύπτει:

```
E(T) is minimized for a = 0.601000
E(T)_min = 0.000121
```

Η γραφική παράσταση του μέσου χρόνου καθυστέρησης ενός τυχαίου πακέτου στο σύστημα συναρτήσει του α:



Ο κώδικας που υλοποιεί τα παραπάνω:

```
3  clc;
4  clear all;
5  close all;
6  #-----------------------------------
7  a = [0.001 : 0.001 : 0.999];
8  lambda = 10^4;
9  pack_length =128 * 8;          #In bits
10 C2 = 12 * 10^6;                 #In bits/s
11 m2 = C2/pack_length;
12 Numerator = (2 * lambda) * a.^2 - (2 * lambda + 0.25 * m2) * a + 1.25 * m2;
13 Denominator = (-(lambda)^2) * a.^2 + (0.25 * m2 * lambda + (lambda)^2 ) * a + 1.25 * m
14 mean_delay = Numerator./Denominator;
15 figure(1);
16 plot(a, mean_delay);
17 xlabel("a-chance of going through line 1");
18 ylabel("Mean delay time of a packet in the system");
19 title("Mean delay time to splitting chance");
20 min_mean = mean_delay(1);
21 index = a(1);
22 for i = 2 : columns(a)
23   if(mean_delay(i) < min_mean)
24     min_mean = mean_delay(i);
25     index = a(i);
26   endif
27 endfor
28
29 printf("E(T) is minimized for a = %f\n", index);
30 printf("E(T)_min = %f\n", min_mean);
```

<u>Σημείωση:</u> Η γραμμή 13 περιέχει κάποιες πράξεις ακόμα, οι οποίες ωστόσο δε χωρούσαν στο απόκομμα. Ο πλήρης κώδικας παρατίθεται στο .zip αρχείο.

<u>Ανοιχτό δίκτυο ουρών αναμονής:</u>

(1) Παρατηρούμε αρχικά πως έχουμε είσοδο/έξοδο από/στον έξω κόσμο, επομένως έχουμε ανοικτό δίκτυο. Οι απαραίτητες παραδοχές που πρέπει να γίνουν, έτσι ώστε να μπορούμε να μελετήσουμε το ανοιχτό δίκτυο με το θεώρημα Jackson είναι οι εξής:

- Τυχαίες εξωτερικές αφίξεις ~ Poisson
- Τυχαία δρομολόγηση πελατών βάσει των πιθανοτήτων διάσπασης
- Ανεξάρτητες εκθετικές εξυπηρετήσεις πελατών, βασισμένοι στην παραδοχή Kleinrock περί ανεξαρτησίας εξυπηρετητών.
- Άπειρες FIFO ουρές, χωρίς απώλειες.

(2) Η **ένταση του φορτίου ρ_i**, που δέχεται η κάθε ουρά συναρτήσει των παραμέτρων λ_i (ρυθμός αφίξεων) και μ_i (ρυθμός εξυπηρετήσεων) είναι:

- $\rho_1 = \frac{\lambda_1}{\mu_1}$
- $\rho_2 = \frac{\lambda_2 + (2/7)\lambda_1}{\mu_2}$
- $\rho_3 = \frac{(4/7)\lambda_1}{\mu_3}$
- $\rho_4 = \frac{(1/2)(4/7)\lambda_1 + (1/7)\lambda_1}{\mu_4}$
- $\rho_5 = \frac{(\lambda 2 + (2/7)\lambda_1) + (1/2)(4/7)\lambda_1}{\mu_5}$

Παρακάτω, ο κώδικας για τη συνάρτηση **intensities** (όπου η συνθήκη εργοδικότητας είναι $\rho_i < 1$):

```
function [v1,v2,v3,v4,v5] = intensities(l1,l2,m1,m2,m3,m4,m5)
  ergodic = [0, 0, 0 ,0, 0];
  v1 = l1/m1;
  if(v1 < 1)
    ergodic(1) = 1;
  endif
  v2 = (l2 + (2/7) * l1)/m2;
  if(v2 < 1)
    ergodic(2) = 1;
  endif
  v3 = ((4/7) * l1)/m3;
  if(v3 < 1)
    ergodic(3) = 1;
  endif
  v4 = ((1/7) * l1 + (1/2) * ((4/7) * l1)) / m4;
  if(v4 < 1)
    ergodic(4) = 1;
  endif
  v5 = ((l2 + (2/7) * l1) + (1/2) * ((4/7) * l1))/m5;
  if(v5 < 1)
    ergodic(5) = 1;
  endif
  for i = ergodic
    if (i == 0)
      printf("Non-Ergodic System\n");
      break;
    endif
  endfor
  if(i)
    printf("Ergodic System (1)\n");
  endif
endfunction
```

Εκτελώντας την για “εργοδικούς” συνδυασμούς (π.χ. $\lambda_i = 1$ και $\mu_i = 10$) και για μη εργοδικούς ($\lambda_i = 10$, $\mu_i = 1$), λαμβάνουμε τα εξής αποτελέσματα:

```
>> [x1,x2,x3,x4,x5] = intensities(1,1,10,10,10,10,10)
Ergodic System (1)
x1 = 0.1000
x2 = 0.1286
x3 = 0.057143
x4 = 0.042857
x5 = 0.1571
>> [y1,y2,y3,y4,y5] = intensities(10,10,1,1,1,1,1)
Non-Ergodic System
y1 = 10
y2 = 12.857
y3 = 5.7143
y4 = 4.2857
y5 = 15.714
```

(3) Ο μέσος αριθμός πελατών της κάθε ουράς Q_i δίνεται από τη σχέση: $E[n_1] = \frac{\rho_i}{1-\rho_i}$. Υλοποιούμε, συνεπώς, τη συνάρτηση **mean_clients**, και στη συνέχεια την εκτελούμε (για $\lambda_1 = 1$, $\lambda_2 = 2$, $\mu_1 = 3$, $\mu_2 = 4$, $\mu_3 = 5$, $\mu_4 = 6$, $\mu_5 = 7$) :

```
#Task5_2_3 - Mean clients
function [x1,x2,x3,x4,x5] = mean_clients(l1,l2,m1,m2,m3,m4,m5)
[t1,t2,t3,t4,t5] = intensities(l1,l2,m1,m2,m3,m4,m5);
x1 = t1./(1-t1); #E[n_1]
x2 = t2./(1-t2); #E[n_2]
x3 = t3./(1-t3); #E[n_3]
x4 = t4./(1-t4); #E[n_4]
x5 = t5./(1-t5); #E[n_5]
endfunction
```

```
>> [x1,x2,x3,x4,x5] = mean_clients(1, 2, 3, 4, 5, 6 ,7)
Ergodic System (1)
x1 = 0.5000
x2 = 1.3333
x3 = 0.1290
x4 = 0.076923
x5 = 0.5806
```

(4)

(α) Για τις δοθείσες τιμές, η ένταση που δέχεται κάθε ουρά είναι:

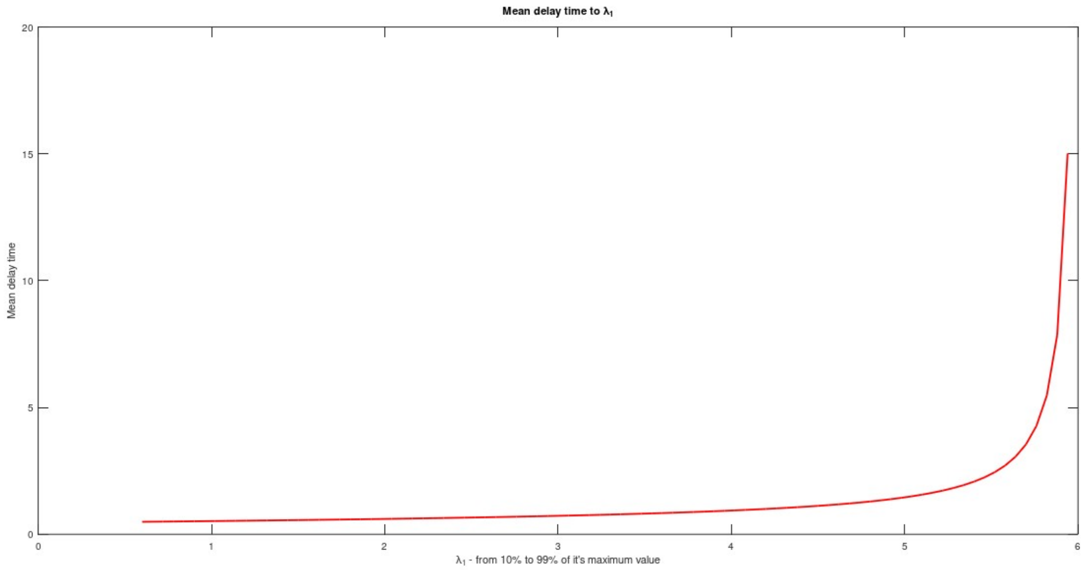
```
>> [vol1,vol2,vol3,vol4,vol5] = intensities(4,1,6,5,8,7,6)
Ergodic System (1)
vol1 = 0.6667
vol2 = 0.4286
vol3 = 0.2857
vol4 = 0.2449
vol5 = 0.5476
```

(β) Ο μέσος χρόνος καθυστέρησης ενός πελάτη από άκρο σε άκρο του δικτύου, υπολογίζεται παρακάτω μέσω της σχέσης **$E[T] = (E[n_1] + E[n_2] + E[n_3] + E[n_4] + E[n_5]) / (\lambda_1 + \lambda_2)$** :

```
>> [mean1,mean2,mean3,mean4,mean5] = intensities(4,1,6,5,8,7,6)
Ergodic System (1)
mean1 = 0.6667
mean2 = 0.4286
mean3 = 0.2857
mean4 = 0.2449
mean5 = 0.5476
>> delay = (mean1+mean2+mean3+mean4+mean5)/(4+1)
delay = 0.4347
```

(5) Από τα παραπάνω αποτελέσματα, παρατηρούμε πως μεγαλύτερη ένταση δέχεται η ουρά 1, η οποία, επομένως αποτελεί και τη στενωπό του δικτύου. Είδαμε πως η ένταση της ουράς 1 δίνεται από τη σχέση $\rho_1 = \frac{\lambda_1}{\mu_1}$, επομένως, για $\mu_1 = 6$, η μέγιστη τιμή του λ_1, ώστε να παραμένει εργοδικό το σύστημα είναι $\boldsymbol{\lambda_1 = 6}$.

(6) Η ζητούμενη γραφική παράσταση:



Όπως αναμενόταν, για μικρό λ_1, έχουμε μικρή ένταση, άρα και μικρή καθυστέρηση.

Ο κώδικας που την υλοποιεί:

```
#Task5_2_6
lambda1 = 0.6 : 0.06 : 5.94; #from 10% of lambda1 max to 99% of it
lambda2 = 1;
m1 = 6;
m2 = 5;
m3 = 8;
m4 = 7;
m5 = 6;
total_in = (lambda1 .+ lambda2);
[mean1,mean2,mean3,mean4,mean5] = mean_clients(lambda1,lambda2,m1,m2,m3,m4,m5);
figure(1)
total_mean = mean1 + mean2 + mean3 + mean4 + mean5
mean1
delay_mean = total_mean ./ total_in;
plot(lambda1, delay_mean, "r", "linewidth", 1.3);
xlabel("λ_1 - from 10% to 99% of it's maximum value");
ylabel("Mean delay time");
title("Mean delay time to λ_1");

```

<u>Σημείωση:</u> Σε όλες τις συναρτήσεις, καθώς και στο τελευταίο πρόγραμμα που υλοποιεί την γραφική παράσταση έχουν τεθεί element-wise operators για να γίνει η γραφική παράσταση. Αν οι τιμές των λ_i, μ_i είναι “απλές” και όχι πίνακες, οι συναρτήσεις ορίζονται και με απλούς operators (χωρίς την “.”).