Συστήματα Αναμονής – 5η Σειρά Ασκήσεων

Γεώργιος Κυριακόπουλος – el18153

Δίκτυο με εναλλακτική δρομολόγηση:

1. Για να μπορούν να μοντελοποιηθούν οι σύνδεσμοι ως M/M/1 ουρές πρέπει να έχουμε τυχαίες εξωτερικές αφίξεις Poisson, τυχαία δρομολόγηση πελατών βάσει των πιθανοτήτων που φαίνονται στο σχήμα, ανεξάρτητες εκθετικές εξυπηρετήσεις πελατών με την παραδοχή Kleinrock για την ανεξαρτησία εξυπηρετήσεων και άπειρες ουρές FIFO, χωρίς απώλειες.

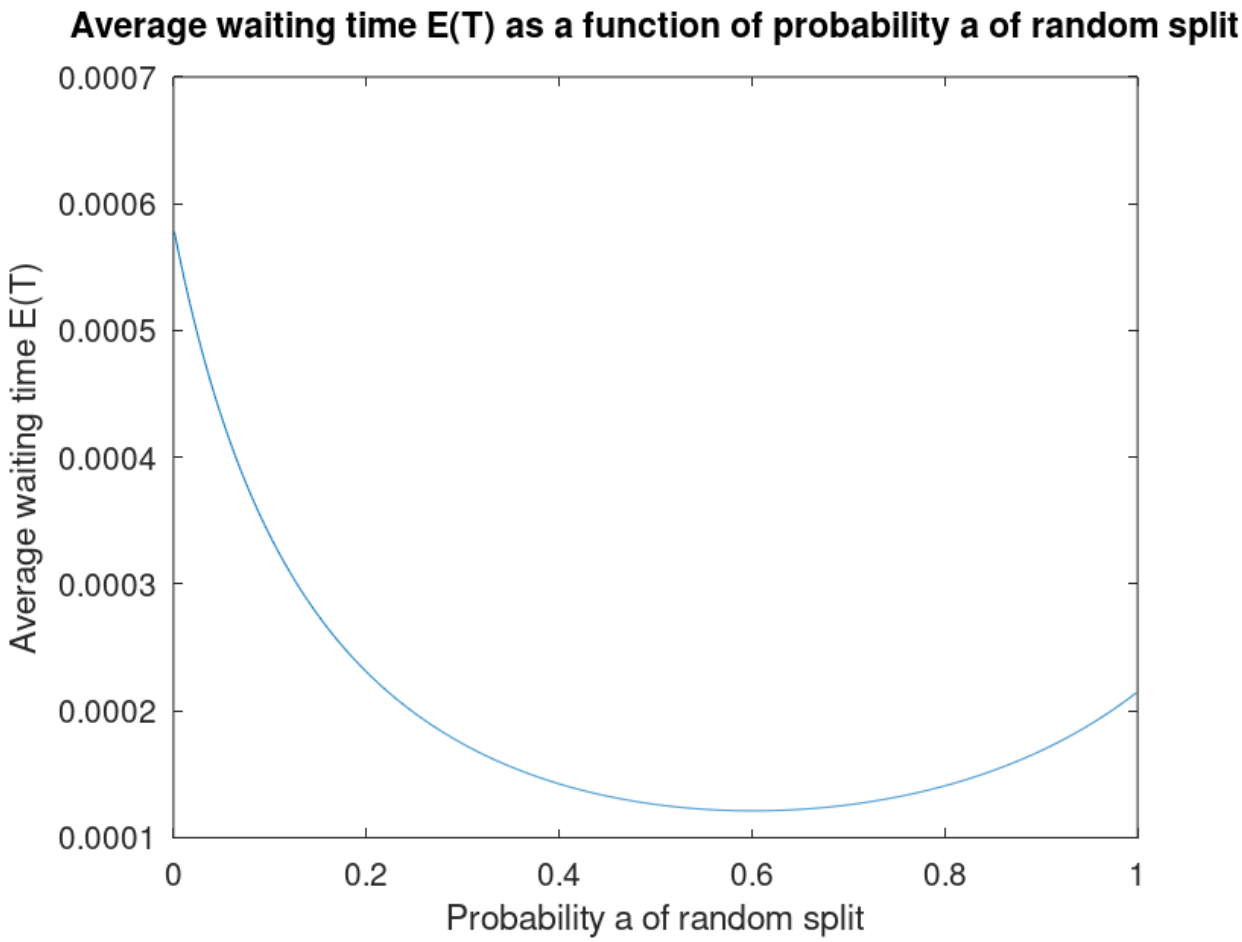
2. Μέσω του τύπου Little και του θεωρήματος του Jackson έχουμε:

$$E(T) = \frac{E(n)}{\gamma} = \frac{E(n)}{\lambda} = \frac{E(n1) + E(n2)}{\lambda} = \frac{\frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2}}{\lambda} = \frac{\frac{\frac{\alpha\lambda}{\mu_1}}{1-\frac{\alpha\lambda}{\mu_1}} + \frac{\frac{(1-\alpha)\lambda}{\mu_2}}{1-\frac{(1-\alpha)\lambda}{\mu_2}}}{\lambda}$$
$$= \frac{\frac{\alpha\lambda}{\mu_1 - \alpha\lambda} + \frac{(1-\alpha)\lambda}{\mu_2 - (1-\alpha)\lambda}}{\lambda} = \frac{\alpha}{\mu_1 - \alpha\lambda} + \frac{(1-\alpha)}{\mu_2 - (1-\alpha)\lambda}$$

Με τη βοήθεια του Octave μέσω του παρακάτω κώδικα και του διαγράμματος που ακολουθεί, λαμβάνουμε ότι για α = 0.6 έχουμε τον ελάχιστο χρόνο καθυστέρησης $E(T) = 0.000121198\ sec$.

```
% Calculate E(T) and pair of min a and min E(T) for given network.

clc;
clear all;
close all;
pkg load queueing;

a = 0.001:0.001:0.999;
lambda = 10 .* 10^3;

mu1 = (15 .* 10^6) / (128 .* 8);
```

```
mu2 = (12 .* 10^6) / (128 .* 8);

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

[U1, R1, Q1, X1, p01]= qsmm1(lambda1, mu1);
[U2, R2, Q2, X2, p02]= qsmm1(lambda2, mu2);

Et = (Q1 .+ Q2) ./ lambda;

figure(1);
plot(a, Et);
title("Average waiting time E(T) as a function of probability a of ra
ndom split");
xlabel("Probability a of random split");
ylabel("Average waiting time E(T)");

[minEt, minA] = min(Et);
printf("The minimum value of E(T) is equal to %d when a is equal to %
d.\n", minEt, minA .* 0.001 .- a(1));
```




```
The minimum value of E(T) is equal to 0.000121198 when a is equal to 0.6.
```

Ανοιχτό δίκτυο ουρών αναμονής:

1. Για να μπορεί το παραπάνω δίκτυο να μελετηθεί ως ένα ανοιχτό δίκτυο με το θεώρημα Jackson πρέπει να έχουμε τυχαίες εξωτερικές αφίξεις Poisson, τυχαία δρομολόγηση πελατών βάσει των πιθανοτήτων που φαίνονται στο σχήμα, ανεξάρτητες εκθετικές εξυπηρετήσεις πελατών με την παραδοχή Kleinrock για την ανεξαρτησία εξυπηρετήσεων και άπειρες ουρές FIFO, χωρίς απώλειες.

2. Ξέρουμε ότι $\rho_i = \frac{\lambda_i}{\mu_i}$, επομένως έχουμε:

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

$$\rho_2 = \frac{\lambda_2 + \rho_{12}\lambda_1}{\mu_2} = \frac{\lambda_2 + \frac{2}{7}\lambda_1}{\mu_2}$$

$$\rho_3 = \frac{\rho_{13}\lambda_1}{\mu_3} = \frac{4\lambda_1}{7\mu_3}$$

$$\rho_4 = \frac{(\rho_{14} + \rho_{34}\rho_{13})\lambda_1}{\mu_4} = \frac{3\lambda_1}{7\mu_4}$$

$$\rho_5 = \frac{(\rho_{12} + \rho_{35}\rho_{13})\lambda_1 + \lambda_2}{\mu_5} = \frac{\frac{4}{7}\lambda_1 + \lambda_2}{\mu_5}$$

Παρακάτω παραθέτουμε και τον κώδικα για τη συνάρτηση που ζητείται:

```
function [rho, ergodic] = intensities(lambda, mu)
  rho(1) = lambda(1)/mu(1);
  rho(2) = (lambda(2) + (2/7)*lambda(1))/mu(2);
  rho(3) = ((4/7)*lambda(1))/mu(3);
  rho(4) = ((3/7)*lambda(1))/mu(4);
  rho(5) = (lambda(2) + (4/7)*lambda(1))/mu(5);
  ergodic = 1;
  for i=1:5
    if rho(i) >= 1
      ergodic = 0;
    endif
    printf("Intensity of Queue %d is equal to: %d.\n", i, rho(i));
  endfor
endfunction
```

3. Ακολουθεί ο κώδικας της συνάρτησης που ζητείται:

```
function [q] = mean_clients(lambda, mu)
  [rho, ergodic] = intensities(lambda, mu);
  q = rho ./ (1 - rho);
Endfunction
```

4. Παρακάτω ακολουθεί ο κώδικας που χρησιμοποιήθηκε, καθώς και η έξοδος αυτού, όπου βλέπουμε τις εντάσεις κάθε ουράς καθώς και το μέσο χρόνο καθυστέρησης ενός πελάτη από άκρο σε άκρο του δικτύου:

```
lambda = [4, 1];
mu = [6, 5, 8, 7, 6];
q = mean_clients(lambda, mu);
et = sum(q)/sum(lambda);
printf("Mean service time is equal to: %d.\n", et);
```

```
Intensity of Queue 1 is equal to: 0.666667
Intensity of Queue 2 is equal to: 0.428571
Intensity of Queue 3 is equal to: 0.285714
Intensity of Queue 4 is equal to: 0.244898
Intensity of Queue 5 is equal to: 0.547619
Mean service time is equal to: 0.93697
```

5. Παραθέτουμε το απόσπασμα του κώδικα που χρησιμοποιήθηκε και την έξοδο του, όπου σύμφωνα και με τα προηγούμενα ερωτήματα, παρατηρούμε πως η ουρά 1 είναι αυτή με το μεγαλύτερο φορτίο και επομένως η μέγιστη τιμή του λ_1 θα είναι ίση με $\lambda_{1_{max}} = \rho_{1_{max}} \cdot \mu_1$ και γνωρίζοντας ότι $\rho_{1_{max}} = 1$, θα πάρουμε $\lambda_{1_{max}} = 1 \cdot 6 = 6$.

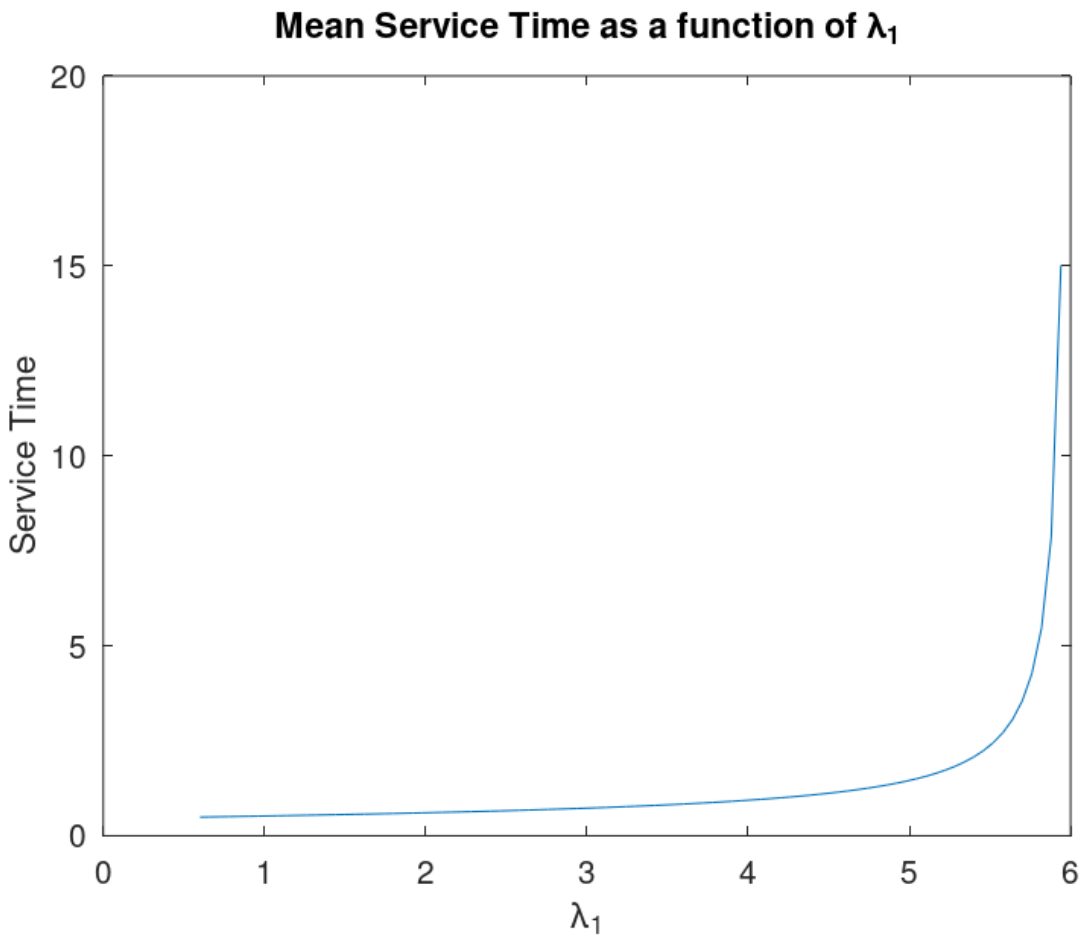
```
[maxq, maxi] = max(q);
max_lambda = 1 * mu(maxi);
printf('The maximum value lambda_1 can have is: %d.\n', max_lambda);
```

```
The maximum value lambda_1 can have is: 6.
```

6. Ακολουθεί ο κώδικας με βάση τον οποίον παρήγαμε το διάγραμμα του μέσου χρόνου καθυστέρησης ενός πελάτη από άκρη σε άκρη του δικτύου που ακολουθεί:

```
for i=1:9
  x(i) = NaN;
  et2(i) = NaN;
endfor
```

```
for i=10:99
  l = max_lambda*i/100;
  x(i) = l;
  lambda(1) = l;
  et2(i) = sum(mean_clients(lambda, mu))/sum(lambda);
endfor

figure(1);
plot(x, et2);
title('Mean Service Time as a function of \lambda_1');
xlabel('\lambda_1');
ylabel('Service Time');
```



Τέλος, ακολουθεί ολόκληρο το απόσπασμα του κώδικα που χρησιμοποιήθηκε:

```
% Calculate intensities, mean clients and average service time for gi
ven network.

clc;
clear all;
close all;
pkg load queueing;

function [rho, ergodic] = intensities(lambda, mu)
  rho(1) = lambda(1)/mu(1);
  rho(2) = (lambda(2) + (2/7)*lambda(1))/mu(2);
  rho(3) = ((4/7)*lambda(1))/mu(3);
  rho(4) = ((3/7)*lambda(1))/mu(4);
  rho(5) = (lambda(2) + (4/7)*lambda(1))/mu(5);
  ergodic = 1;
  for i=1:5
    if rho(i) >= 1
      ergodic = 0;
    endif
    printf("Intensity of Queue %d is equal to: %d.\n", i, rho(i));
  endfor
endfunction

function [q] = mean_clients(lambda, mu)
  [rho, ergodic] = intensities(lambda, mu);
  q = rho ./ (1 - rho);
endfunction

lambda = [4, 1];
mu = [6, 5, 8, 7, 6];
q = mean_clients(lambda, mu);
et = sum(q)/sum(lambda);
printf("Mean service time is equal to: %d.\n", et);

[maxq, maxi] = max(q);
max_lambda = 1 * mu(maxi);
printf('The maximum value lambda_1 can have is: %d.\n', max_lambda);

for i=1:9
  x(i) = NaN;
  et2(i) = NaN;
```

```
endfor

for i=10:99
  l = max_lambda*i/100;
  x(i) = l;
  lambda(1) = l;
  et2(i) = sum(mean_clients(lambda, mu))/sum(lambda);
endfor

figure(1);
plot(x, et2);
title('Mean Service Time as a function of \lambda_1');
xlabel('\lambda_1');
ylabel('Service Time');
```