Συστήματα Αναμονής, 5η εργαστηριακή άσκηση

Νικόλαος Παγώνας, el18175

Δίκτυο με εναλλακτική δρομολόγηση

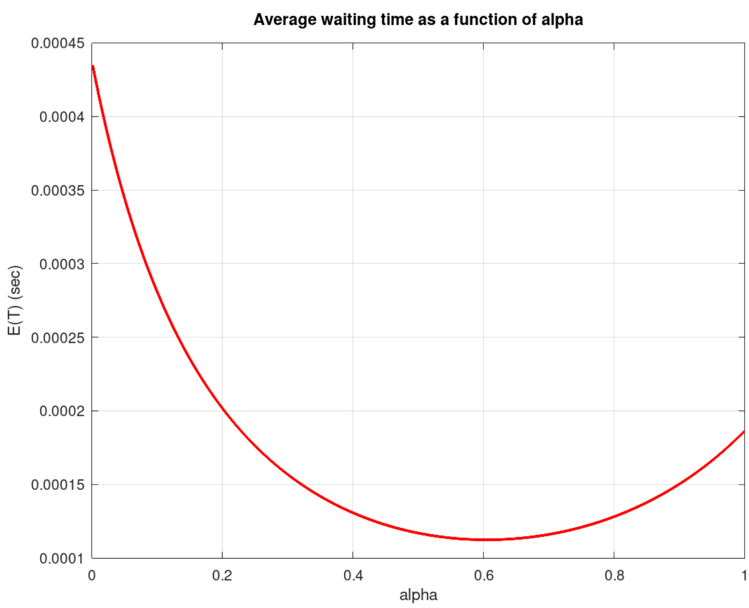
(1)

Οι παραδοχές που πρέπει να κάνουμε προκειμένου οι σύνδεσμοι (γραμμές) να μπορούν να μοντελοποιηθούν σαν M/M/1 ουρές είναι:

- Οι εξωτερικές αφίξεις είναι ανεξάρτητες ροές Poisson.
- Έχουμε ανεξάρτητους εκθετικούς ρυθμούς εξυπηρέτησης μ_i.
- Η εσωτερική δρομολόγηση γίνεται με τυχαίο τρόπο.
- Οι χρόνοι εξυπηρέτησης πελατών χαρακτηρίζονται από έλλειψη μνήμης (Kleinrock's Independence Assumption)
- Έχουμε άπειρες FIFO ουρές, χωρίς απώλειες.

(2)

Κάνοντας τις ανωτέρω παραδοχές, χρησιμοποιούμε το Octave για να σχεδιάσουμε το διάγραμμα μέσου χρόνου καθυστέρησης E(T) ενός τυχαίου πακέτου στο σύστημα συναρτήσει του α, (α = 0.001, 0.002, ..., 0.999):



Για το παραπάνω διάγραμμα χρησιμοποιήθηκαν οι σχέσεις:

$$\lambda_1 = a \cdot \lambda$$

$$\lambda_2 = (1-a) \cdot \lambda$$

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

$$\rho_2 = \frac{\lambda_2}{\mu_2}$$

$$E(n_1) = \frac{\rho_1}{1-\rho_1}$$

$$E(n_2) = \frac{\rho_2}{1-\rho_2}$$

$$E(n) = E(n_1) + E(n_2)$$

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

Στη συνέχεια, υπολογίζουμε την τιμή του α που ελαχιστοποιεί το E(T), καθώς και τον ελάχιστο χρόνο καθυστέρησης E(T):

`The minimum E(T) is equal to 0.000112357 sec (112.357 usec), for alpha = 0.604`

Ο κώδικας που χρησιμοποιήθηκε

two_lines.m

```
clc;
clear all;
close all;

lambda = 10*10^3;        % lambda = 10 Kpps
mps = 128*8;             % mean packet size = 128 bytes = 128 * 8 bits
C1 = 15*1024*1024;       % C1 = 15 Mbps
C2 = 12*1024*1024;       % C2 = 12 Mbps

% Conversion from bps to pps
C1 = C1 / mps;
C2 = C2 / mps;

mu1 = C1;
mu2 = C2;

a = 0.001:0.001:0.999;

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

rho1 = lambda1 / mu1;
rho2 = lambda2 / mu2;

% Calculate E(n)

E_n1 = rho1./(1-rho1);
E_n2 = rho2./(1-rho2);
E_n = E_n1 + E_n2;

```

```
% E(T) = E(n) / gamma = E(n) / lambda

gamma = lambda;

E_T = E_n / gamma;

plot(a, E_T, "r", "linewidth", 2);

title("Average waiting time as a function of alpha");
xlabel("alpha");
ylabel("E(T) (sec)");
grid on;

saveas(1, "two_lines.png");

[minimum, argmin] = min(E_T);

a_min = a(argmin);

fd = fopen("two_lines.txt", "w");

fprintf(fd, "The minimum E(T) is equal to %d sec (%d usec), for alpha = %d\n", minimum,
    minimum*1e6, a_min);

fclose(fd);
```

Ανοιχτό δίκτυο ουρών αναμονής

(1)

Οι παραδοχές που πρέπει να κάνουμε ώστε το παραπάνω δίκτυο να μπορεί να μελετηθεί ως ανοιχτό δίκτυο με το θεώρημα Jackson είναι:

- Οι εξωτερικές αφίξεις είναι ανεξάρτητες ροές Poisson.
- Έχουμε ανεξάρτητους εκθετικούς ρυθμούς εξυπηρέτησης μ_i.
- Η εσωτερική δρομολόγηση γίνεται με τυχαίο τρόπο.
- Οι χρόνοι εξυπηρέτησης πελατών χαρακτηρίζονται από έλλειψη μνήμης (Kleinrock's Independence Assumption)
- Έχουμε άπειρες FIFO ουρές, χωρίς απώλειες.

(2)

Για τις ουρές $Q_1 - Q_5$ έχουμε:

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

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

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

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

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

Η ζητούμενη συνάρτηση είναι υλοποιημένη στο αρχείο `intensities.m`

```
function [rho1, rho2, rho3, rho4, rho5, is_ergodic] = ...
                          intensities(lambda1, lambda2, mu1, mu2, mu3, mu4, mu5)
  rho1 = lambda1 / mu1;
  rho2 = (2/7*lambda1 + lambda2) / mu2;
  rho3 = (4/7*lambda1) / mu3;
  rho4 = (3/7*lambda1) / mu4;
  rho5 = (4/7*lambda1+lambda2) / mu5;

  printf("rho1 = %d\nrho2 = %d\nrho3 = %d\nrho4 = %d\nrho5 = %d\n", ...
          rho1, rho2, rho3, rho4, rho5);

  is_ergodic = (rho1 < 1 && rho2 < 1 && rho3 < 1 && rho4 < 1 && rho5 < 1);
endfunction
```

(3)

Η ζητούμενη συνάρτηση είναι υλοποιημένη στο αρχείο `mean_clients.m`

```
function [E1, E2, E3, E4, E5] = ...
                         mean_clients(lambda1, lambda2, mu1, mu2, mu3, mu4, mu5)

  [rho1, rho2, rho3, rho4, rho5, _] = ...
                         intensities(lambda1, lambda2, mu1, mu2, mu3, mu4, mu5);

  E1 = rho1 ./ (1 - rho1);
  E2 = rho2 ./ (1 - rho2);
  E3 = rho3 ./ (1 - rho3);
  E4 = rho4 ./ (1 - rho4);
  E5 = rho5 ./ (1 - rho5);
endfunction
```

(4)

Για τις τιμές των παραμέτρων που δίνονται, υπολογίζουμε την ένταση του φορτίου κάθε ουράς και τον μέσο χρόνο καθυστέρησης ενός πελάτη από άκρο σε άκρο του δικτύου:

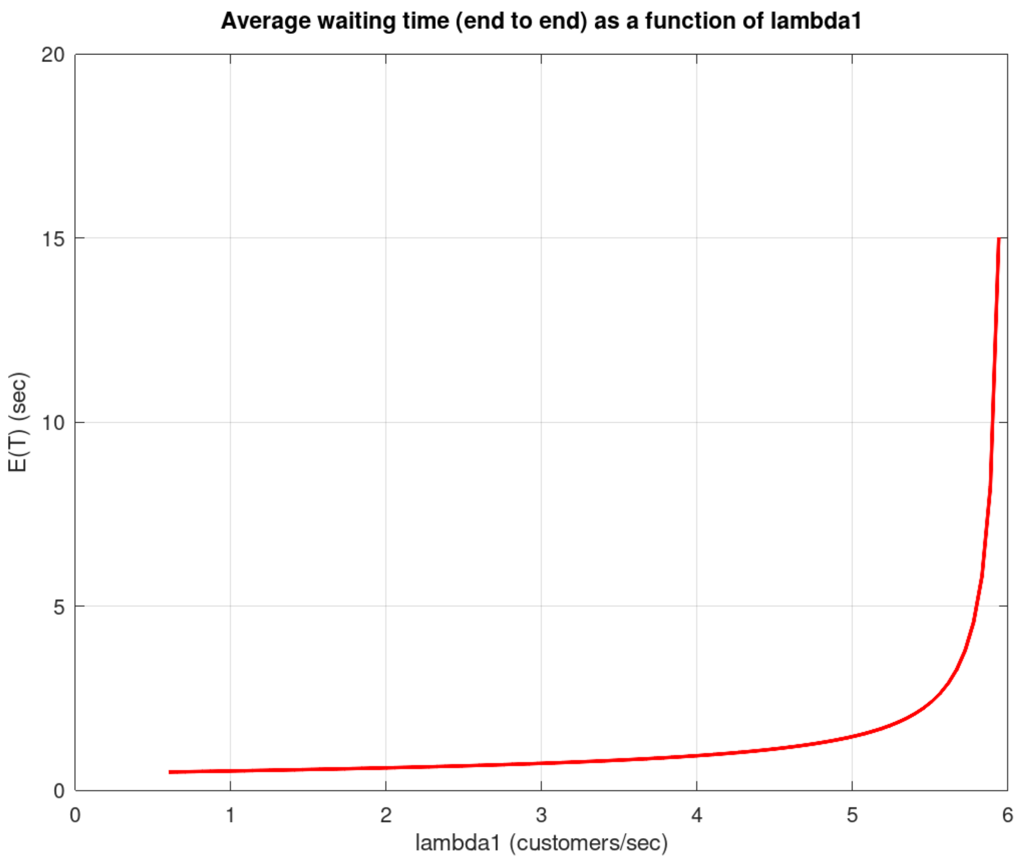
```
rho1 = 0.666667
rho2 = 0.428571
rho3 = 0.285714
rho4 = 0.244898
rho5 = 0.547619

Average waiting time (end to end) = 0.93697 sec
```

(5)

Από το παραπάνω αποτέλεσμα, παρατηρούμε ότι το bottleneck του δικτύου (δηλαδή η ουρά με την μεγαλύτερη ένταση φορτίου) είναι η Q_1. Έτσι, για να παραμείνει το σύστημα εργοδικό, πρέπει και αρκεί να έχουμε $\rho_1 < 1$, δηλαδή $\lambda_1 < \mu_1 = 6$.

(6)

Για τις τιμές των παραμέτρων που δόθηκαν παραπάνω και για λ_1 από 0.1×6 έως 0.99×6, σχεδιάζουμε το διάγραμμα του μέσου χρόνου καθυστέρησης ενός πελάτη από άκρο σε άκρο του δικτύου:



Ο κώδικας που χρησιμοποιήθηκε

network.m

```
clc;
clear all;
close all;

lambda1 = 4;
lambda2 = 1;
mu1 = 6;
mu2 = 5;
mu3 = 8;
mu4 = 7;
mu5 = 6;

[rho1, rho2, rho3, rho4, rho5, _] = ...
                        intensities(lambda1, lambda2, mu1, mu2, mu3, mu4, mu5);

[E1, E2, E3, E4, E5] = mean_clients(lambda1, lambda2, mu1, mu2, mu3, mu4, mu5);

E_T_end_to_end = (E1 + E2 + E3 + E4 + E5) / (lambda1 + lambda2);

fd = fopen("network1.txt", "w");

fprintf(fd, "rho1 = %d\nrho2 = %d\nrho3 = %d\nrho4 = %d\nrho5 = %d\n\n",
        rho1, rho2, rho3, rho4, rho5);

fprintf(fd, "Average waiting time (end to end) = %d sec\n", E_T_end_to_end);

fclose(fd);

[_, argmax] = max([rho1, rho2, rho3, rho4, rho5]);

bottleneck = argmax;

fd = fopen("network2.txt", "w");
fprintf(fd, "The bottleneck is Q%d\n", bottleneck);
fclose(fd);

% ...
% We solve by hand in order to find the maximum value of lambda1,
% such that the system remains ergodic. It turns out that max_lambda1 = 6
% ...

max_lambda1 = 6;
number_of_points = 100;

lambda1 = linspace(0.1*max_lambda1, 0.99*max_lambda1, number_of_points);

[rho1, rho2, rho3, rho4, rho5, _] = ...
                        intensities(lambda1, lambda2, mu1, mu2, mu3, mu4, mu5);

[E1, E2, E3, E4, E5] = mean_clients(lambda1, lambda2, mu1, mu2, mu3, mu4, mu5);

E_T_end_to_end = (E1 + E2 + E3 + E4 + E5) ./ (lambda1 + lambda2);

plot(lambda1, E_T_end_to_end, "r", "linewidth", 2);
grid on;
title("Average waiting time (end to end) as a function of lambda1");
xlabel("lambda1 (customers/sec)");
ylabel("E(T) (sec)");

saveas(1, "network.png");
```