Συστήματα Αναμονής
(Queuing Systems)

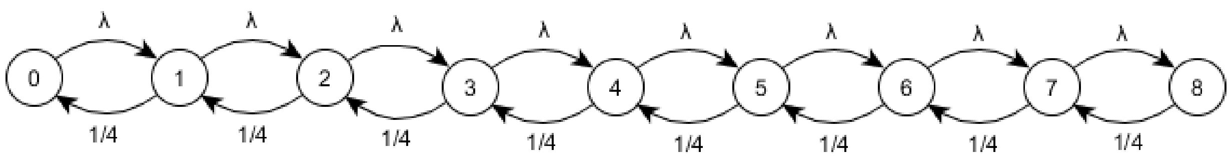
4η Εργαστηριακή Άσκηση

Λεούσης Σάββας

Α.Μ.: 03114945

Σύστημα M/M/N/K (call center)

1. Το διάγραμμα ρυθμών μεταβάσεων του συστήματος μεταξύ εργοδικών καταστάσεων είναι το παρακάτω:



2. Με τη βοήθεια των συναρτήσεων ctmcbd και ctmc του πακέτου queueing του Octave, οι εργοδικές πιθανότητες του συστήματος για τις διάφορες τιμές του λ είναι:
   a. Για λ=1/4

```
P0 =

   0.159399    0.153377    0.142154    0.127223    0.110509    0.094077    0.079859    0.069447    0.063955
```

   b. Για λ=1

```
P0 =

   3.3335e-004   9.1387e-004   2.1177e-003   4.4873e-003   9.3308e-003   2.1012e-002   5.6871e-002   1.8894e-001   7.1599e-001
```

3. Με τη βοήθεια της συνάρτησης erlangc, προκύπτουν για κάθε λ οι εξής πιθανότητες παραμονής ενός πελάτη στο σύστημα:
   a. Για λ=1/4

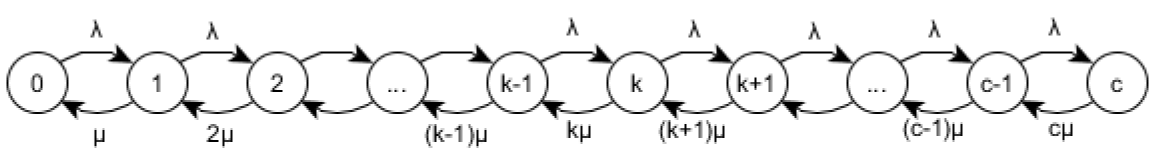
```
C =  0.0038314
```

   b. Για λ=1

```
C =  0.55411
```

Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου

1. Το διάγραμμα ρυθμού μεταβάσεων του συστήματος M/M/c/c είναι το παρακάτω:



Η ζητούμενη συνάρτηση `erlangb_factorial` είναι η παρακάτω:

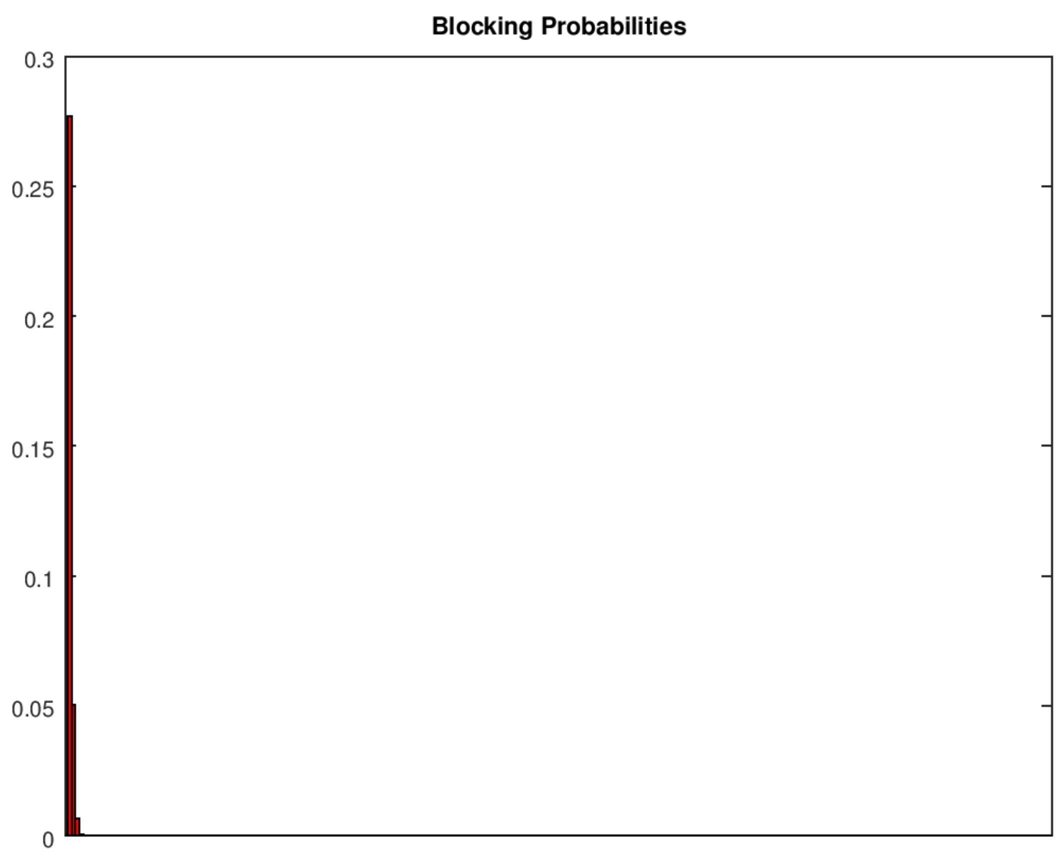
```
function B = erlangb_factorial(rho,c)
  sum = 0;
  for i = 0:c
    sum = sum + (rho^i)/factorial(i);
  endfor
  B=((rho^c)/factorial(c))/sum;
endfunction
```

2. Η ζητούμενη συνάρτηση `erlangb_iterative` είναι η παρακάτω:

```
function B = erlangb_iterative(rho, c)
  if c == 0
    B = 1;
    return;
  else
    B = (rho*erlangb_iterative(rho,c-
1))/(rho*erlangb_iterative(rho,c-1)+c);
  endif
endfunction
```

3. Τρέχοντας τις παραπάνω συναρτήσεις με παραμέτρους ρ=1024 και c=1024, παρατηρούμε ότι το Octave αδυνατεί να υπολογίσει το αποτέλεσμα, διότι έχει ξεπεραστεί το όριο αναδρομών στις κλήσεις συναρτήσεων.

4.
   a. Η συνολική ένταση του φορτίου που καλείται να εξυπηρετήσει το τηλεφωνικό δίκτυο της εταιρείας είναι $\rho = \frac{23\ \lambda\epsilon\pi\tau\acute{\alpha}}{60\ \lambda\epsilon\pi\tau\acute{\alpha}} = 0.383$.
   b. Χρησιμοποιώντας την συνάρτηση erlang_iterative, υπολογίστηκε η πιθανότητα απόρριψης πελάτη για κάθε σύστημα που έχει από 1 έως 200 τηλεφωνικές γραμμές. Το διάγραμμα αυτών των πιθανοτήτων παρατίθεται παρακάτω:

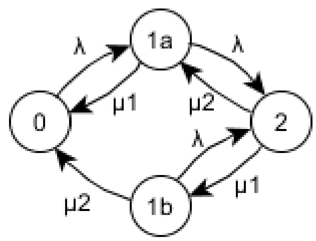


Παρατηρούμε ότι η πιθανότητα αυτή μειώνεται δραματικά όσο αυξάνεται ο αριθμός των γραμμών, από τα 3 πρώτα συστήματα.

c. Σύμφωνα με το διάγραμμα του ερωτήματος (β), μπορούμε να συμπεράνουμε ότι ο πλέον κατάλληλος αριθμός τηλεφωνικών γραμμών έτσι ώστε ηπιθανότητα απόρριψης τηλεφωνικής κλήσης να είναι μικρότερη από 1%, είναι για 3 τηλεφωνικές γραμμές, καθώς η πιθανότητα απόρριψης πελάτη είναι 0.0064031.

Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές

1. Το διάγραμμα ρυθμών μεταβάσεων του συστήματος είναι το παρακάτω:



Οι εργοδικές πιθανότητες του συστήματος προκύπτουν από το παρακάτω σύστημα εξισώσεων ισορροπίας:

- $(1)\ P_0 + P_{1a} + P_{1b} + P_2 = 1$
- $(2)\ P_0 = 0.8P_{1a} + 0.4P_{1b}$
- $(3)\ 1.8P_{1a} = P_0 + 0.4P_2$
- $(4)\ 1.4P_{1b} = 0.8P_2$
- $(5)\ P_{1a} + P_{1b} = 1.2P_2$

Τελικά προκύπτουν οι εξής εργιδικές πιθανότητες:

- $P_0 = 0.249513$
- $P_{1a} = 0.214425$
- $P_{1b} = 0.194932$
- $P_2 = 0.341131$

Η πιθανότητα απόρριψης πελάτη από το σύστημα είναι ίση με:

$$P_{blocking} = P_2 = 0.341131$$

2. Εκτελώντας την προσομοίωση του παραπάνω συστήματος, προκύπτουν οι παρακάτω εργοδικές πιθανότητες, οι οποίες συγκλίνουν αρκετά με τις θεωρητικές:

- $P_0 = 0.24654$
- $P_1 = 0.41362$
- $P_2 = 0.33984$
- $P_{blocking} = P_2 = 0.33984$
- Μέσος αριθμός πελατών = 1.2020

Παράρτημα (κώδικας Lab4.m)

```
clc;
clear all;
close all;
fig_num = 1;
############# M/M/N/K SYSTEM (CALL CENTER) #############

# 2

states = [0,1,2,3,4,5,6,7,8];
initial_state = [1,0,0,0,0,0,0,0,0];
lambda = 1/4;
mu = 1/4;
births_B =
[lambda,lambda,lambda,lambda,lambda,lambda,lambda,lambda];
deaths_D = [mu,mu,mu,mu,mu,mu,mu,mu];
transition_matrix = ctmcbd(births_B,deaths_D);
P = ctmc(transition_matrix);
for i=[1,2,3,4,5,6,7,8,9]
  index = 0;
  for T=0:0.01:50
    index = index + 1;
    P0 = ctmc(transition_matrix,T,initial_state);
    Prob0(index) = P0(i);
    if P0-P < 0.01
      break;
    endif
  endfor
endfor
display(lambda);
display(P0);

states = [0,1,2,3,4,5,6,7,8];
initial_state = [1,0,0,0,0,0,0,0,0];
lambda = 1;
mu = 1/4;
births_B =
[lambda,lambda,lambda,lambda,lambda,lambda,lambda,lambda];
deaths_D = [mu,mu,mu,mu,mu,mu,mu,mu];
transition_matrix = ctmcbd(births_B,deaths_D);
P = ctmc(transition_matrix);
for i=[1,2,3,4,5,6,7,8,9]
  index = 0;
  for T=0:0.01:50
    index = index + 1;
    P0 = ctmc(transition_matrix,T,initial_state);
    Prob0(index) = P0(i);
    if P0-P < 0.01
      break;
    endif
```

```
  endfor
endfor
display(lambda);
display(P0);

# 3

C = erlangc(lambda/mu, 5);

############# CALL CENTER DESIGNING AND ANALYSIS
#############

# 1

function B = erlangb_factorial(rho, c)
  sum = 0;
  for i = 0:c
    sum = sum + (rho^i)/factorial(i);
  endfor
  B=((rho^c)/factorial(c))/sum;
endfunction

# 2

function B = erlangb_iterative(rho, c)
  if c == 0
    B = 1;
    return;
  else
    B = (rho*erlangb_iterative(rho,c-
1))/(rho*erlangb_iterative(rho,c-1)+c);
  endif
endfunction

# 3

#B = erlangb_factorial(1024, 1024);
#display(B);
#B= erlangb_iterative(1024,1024);
#display(B);

# 4b

rho = 23/60;
min = 1;
min_i = -1;
not_found = true;
P_b = zeros(200,1);
for i=1:200
  if i==1
    P_b(i) = erlangb_iterative(rho,i);
  else
```

```
    P_b(i) = (rho*P_b(i-1))/(rho*P_b(i-1)+i);
  endif
  if P_b(i)< 0.01 && not_found
    min = P_b(i);
    min_i = i;
    not_found = false;
  endif
endfor

figure(fig_num++);
bar(P_b,'r','barwidth',1);
title("Blocking Probabilities");
set(gca,'xtick',[])
set(gca,'xticklabel',[])
display(min);
display(min_i);

############# SERVER SYSTEM WITH 2 NONIDENTICAL SERVERS
#############

# 2

lambda = 1;
mu = [0.8,0.4,1.2];
total_arrivals = 0; % to measure the total number of arrivals
current_state = 0;  % holds the current state of the system
previous_mean_clients = 0; % will help in the convergence
test
index = 0; % the threshold used to calculate probabilities
rand("seed",1);
transitions = 0; % holds the transitions of the simulation in
transitions steps
threshold = lambda/(lambda + mu(1));

while transitions >= 0
  transitions = transitions + 1; % one more transitions step

  if mod(transitions,1000) == 0 % check for convergence every
1000 transitions steps
    index = index + 1;
    for i=1:1:length(arrivals)
        P(i) = arrivals(i)/total_arrivals; % calculate the
probability of every state in the system
    endfor
    P_blocking = P(length(arrivals));
    mean_clients = 0; % calculate the mean number of clients
in the system
    for i=1:1:length(arrivals)
       mean_clients = mean_clients + (i-1).*P(i);
    endfor

    to_plot(index) = mean_clients;
```

```
    if abs(mean_clients - previous_mean_clients) < 0.00001 ||
transitions > 300000 % convergence test
      break;
    endif

    previous_mean_clients = mean_clients;

  endif

  random_number = rand(1); % generate a random number
(Uniform distribution)
  if current_state == 0 || random_number < threshold %
arrival
    total_arrivals = total_arrivals + 1;
    try % to catch the exception if variable arrivals(i) is
undefined. Required only for systems with finite capacity.
      arrivals(current_state + 1) = arrivals(current_state +
1) + 1; % increase the number of arrivals in the current
state
    catch
      arrivals(current_state + 1) = 1;
    end
    if current_state == 1
      threshold = lambda/(lambda + mu(1));
    endif
    if current_state == 2
      threshold = lambda/(lambda + mu(3));
      continue;
    else
      current_state = current_state + 1;
    endif
  else % departure
    if current_state != 0 % no departure from an empty system
      current_state = current_state - 1;
    endif
    if current_state == 0
      threshold = lambda/(lambda + mu(1));
    endif
    if current_state == 1
      threshold = lambda/(lambda + mu(2));
    endif
    if current_state == 2
      threshold = lambda/(lambda + mu(3));
    endif
  endif
endwhile

for i=1:1:length(arrivals)
  display(P(i));
endfor
display(P_blocking);
```

```
display(mean_clients);
```