# Συστήματα Αναμονής Άσκηση $4^η$

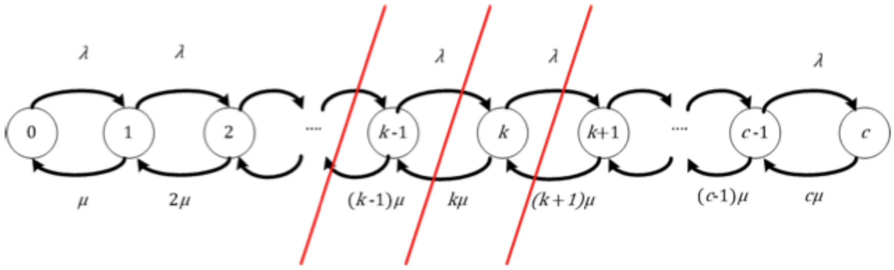
Αυγερινός Πέτρος 03115074

# Contents

# 1 Αναλυση και Σχεδιασμος τηλεφωνικου κεντρου

## 1.1 Ερώτημα 1

Το διάγραμμα ρυθμού μεταβάσεων ενός συστήματος M/M/c/c είναι το παρακάτω:




Από τις εξισώσεις ισορροπίας γνωρίζουμε ότι:

$$P_k = \frac{\lambda}{\mu k} P_{k-1} = \frac{\lambda}{\mu k} \cdot \frac{\lambda}{\mu(k-1)} P_{k-2} = \ldots = \frac{(\frac{\lambda}{\mu})^k}{k!} P_0 \tag{1}$$

και

$$P_0 + P_1 + \ldots + P_c = 1 \tag{2}$$

Από (1) και (2) προκύπτει ότι:

$$\begin{aligned} P_0 + P_1 + \ldots + P_c &= 1 \\ P_0 + \frac{\lambda}{\mu} P_0 + \ldots + \frac{(\frac{\lambda}{\mu})^c}{c!} P_0 &= 1 \\ P_0 (1 + \frac{\lambda}{\mu} + \ldots + \frac{(\frac{\lambda}{\mu})^c}{c!}) &= 1 \\ P_0 \sum_{k=0}^{c} \frac{(\frac{\lambda}{\mu})^k}{k!} &= 1 \\ P_0 &= \frac{1}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} \end{aligned}$$

Η πιθανότητα απόρριψης είναι ίση με:

$$\begin{aligned} P_{blocking} = P_c = B(\rho, c) = \frac{(\frac{\lambda}{\mu})^c}{c!} P_0 &= \\ \frac{\rho^c}{c!} \frac{1}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} \end{aligned}$$

Ο μέσος ρυθμός απωλειών από την ουρά είναι ίσος με:

$$\lambda - \gamma = \lambda - \lambda(1 - P_{blocking}) = \lambda P_{blocking} = \lambda \frac{\rho^c}{c!} \frac{1}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} \tag{3}$$

Ο κώδικας για την συνάρτηση erlangb_factorial φαίνεται παρακάτω:

```
function p = erlangb_factorial (r,c)
    s = 0;
    for k = 0:1:c
        s = s + (power(r,k)/factorial(k));
    endfor
    p = (power(r,c)/factorial(c))/s;
endfunction
```

Το αποτέλεσμα του factorial για τις τιμές $9, 9$ είναι ίδιες και ίσες με 0.2243.

## 1.2 Ερώτημα 2

Για την απόδειξη της iterative formula αρχικά γνωρίζουμε ότι αν δεν υπάρχει εξυπηρετητής όλες οι κλήσεις απορρίπτονται. Επομένως:

$$B(\rho, 0) = 1 \tag{4}$$

Για $c-1$ εξυπηρετητές έχουμε:

$$B(\rho, c-1) = \frac{\rho^{c-1}}{(c-1)!} \frac{1}{\sum_{k=0}^{c-1} \frac{\rho^k}{k!}} \tag{5}$$

Προσθέτοντας έναν ακόμα εξυπηρετητή έχουμε:

$$\begin{aligned} B(\rho, c) = \frac{\rho^c}{c!} \frac{1}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} &= \\ \frac{\frac{\rho^{c-1} \cdot \rho}{(c-1)! \cdot c}}{\frac{\rho^c}{c!} + \sum_{k=0}^{c-1} \frac{\rho^k}{k!}} &= \\ \frac{\frac{\rho^{c-1}}{(c-1)!} \cdot \frac{\rho}{c}}{\frac{\rho}{c} \cdot \frac{\rho^{c-1}}{(c-1)!} + \sum_{k=0}^{c-1} \frac{\rho^k}{k!}} &= \\ \frac{B(\rho, c-1) \cdot \frac{\rho}{c}}{B(\rho, c-1) \cdot \frac{\rho}{c} + 1} &= \\ \frac{\rho \cdot B(\rho, c-1)}{\rho \cdot B(\rho, c-1) + c} & \end{aligned}$$

Ο κώδικας για την συνάρτηση erlangb_iterative φαίνεται παρακάτω:

```
function p = erlangb_iterative (r,c)
    p = 1;
    for k = 0:1:c
        p = (r*p)/(k+r*p);
    endfor
endfunction
```

Η iterative λύση δίνει το ίδιο αποτέλεσμα με το factorial για τις τιμές $9, 9$ και είναι ίσο με 0.2243.

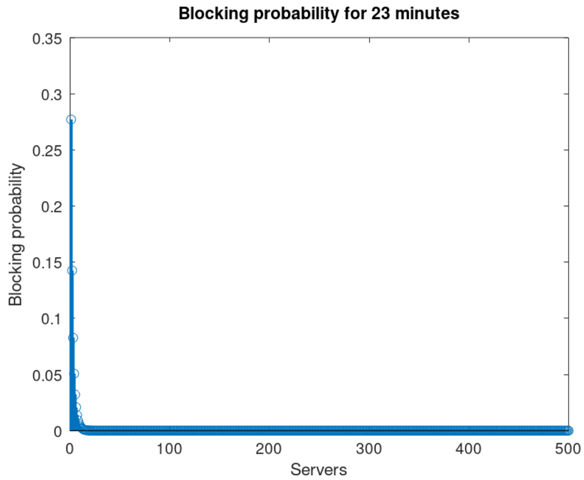
## 1.3 Ερώτημα 3

Στην περίπτωση του factorial παίρνουμε NaN ενώ στην περίπτωση του iterative παίρνουμε το αποτέλεσμα 0.0245243. Αυτό συμβαίνει γιατί το factorial παίρνει πολύ μεγάλες τιμές και υπερχειλίζει.

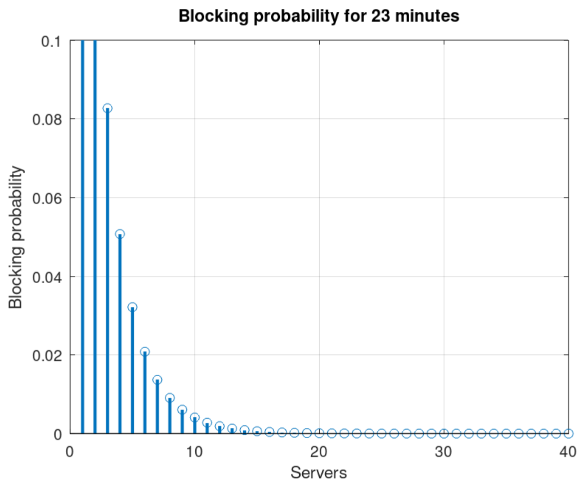
## 1.4 Ερώτημα 4

1. Η συνολική κυκλοφοριακή ένταση του δικτύου για τον πιο απαιτητικό χρήστη είναι:
$$\rho = \frac{500 \cdot 23}{60} = 191.66 \text{ Erlangs} \tag{6}$$

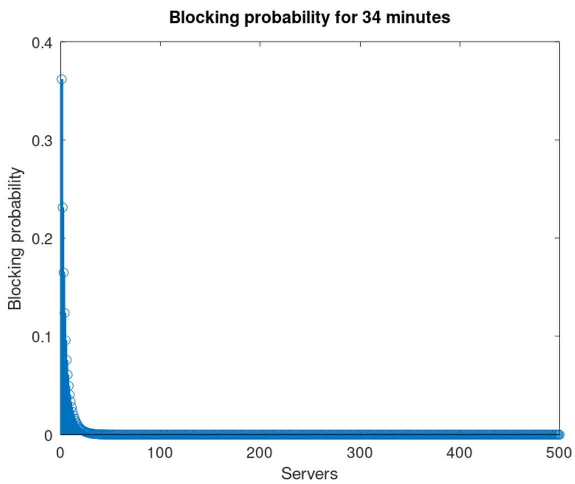
2. Τα διαγράμματα απόρριψης για τις τρεις τιμές λεπτών φαίνονται παρακάτω:
Για 23 λεπτά:



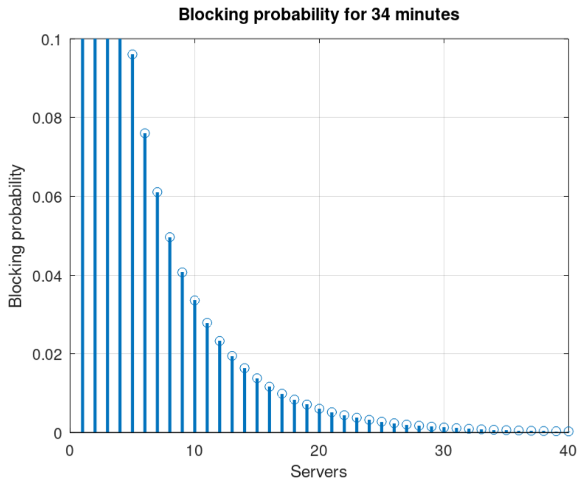

Με ζουμ:



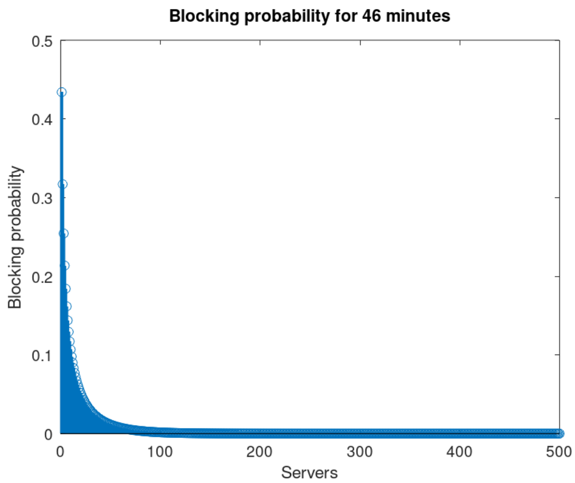

Για 34 λεπτά:



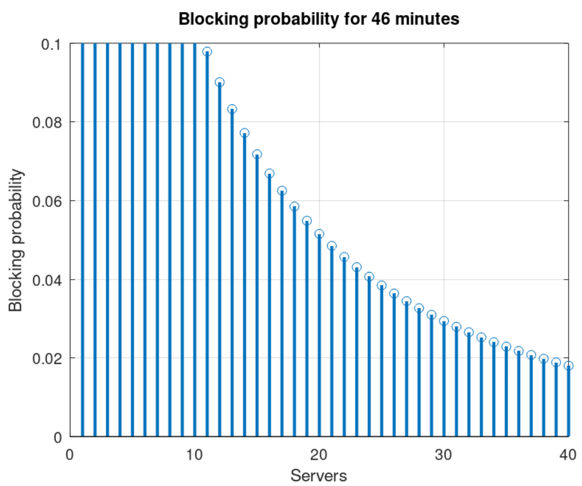
Με ζουμ:




Για 46 λεπτά:




Με ζουμ:




3. Προσθέτοντας μία συνθήκη για πιθανότητα μικρότερη του 0.01 στον κώδικα βλέπουμε ότι:

   (a) Για 23 λεπτά, οι εξυπηρετητές που απαιτούνται είναι 8.

   (b) Για 34 λεπτά, οι εξυπηρετητές που απαιτούνται είναι 17.

   (c) Για 46 λεπτά, οι εξυπηρετητές που απαιτούνται είναι 54.

## 1.5 Ερώτημα 5

1. Η μέση τιμή της συνολικής προσφερόμενης κίνησης είναι $A = 20000 \cdot 0.06 = 1200$ Erlangs.
2. Η κίνηση υπεραστικών είναι $A' = 0.05A = 60$ Erlangs. Για GoS 0.01, με χρήση ενός Erlang B calculator παίρνουμε ότι τα trunks που απαιτούνται είναι 75.
3. Με τριπλασιασμό του φορτίου η κίνηση υπεραστικών είναι $A'' = 3A' = 180$ Erlangs. Με χρήση του Erlang B calculator για 75 trunks παίρνουμε GoS 0.587.

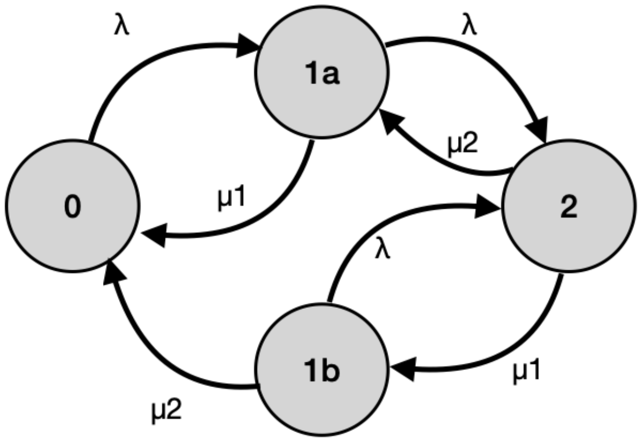
Ο κώδικας για τα παραπάνω ερωτήματα φαίνεται παρακάτω:

```
% pkg load queueing
c = 9;
arg_list = argv();
minutes = str2num(arg_list{1});

function p = erlangb_factorial (r,c)
    s = 0;
    for k = 0:1:c
        s = s + (power(r,k)/factorial(k));
    endfor
    p = (power(r,c)/factorial(c))/s;
endfunction

function p = erlangb_iterative (r,c)
    p = 1;
    for k = 0:1:c
        p = (r*p)/(k+r*p);
    endfor
endfunction

filename_text = sprintf("../erlangb-%d.txt", minutes);
fd = fopen(filename_text, "w");

text = sprintf("Minutes = %d\n", minutes);
fprintf(fd, text);

text = sprintf("erlangb_factorial(9,9) = %d\n", erlangb_factorial(9,9));
fprintf(fd, text);

text = sprintf("erlangb_iterative(9,9) = %d\n", erlangb_iterative(9,9));
fprintf(fd, text);

text = sprintf("erlangb_factorial(1024,1024) = %d\n", erlangb_factorial(1024,1024));
fprintf(fd, text);

text = sprintf("erlangb_iterative(1024,1024) = %d\n", erlangb_iterative(1024,1024));
fprintf(fd, text);


p = zeros(0,500);

flag = 0;

for i = 1:1:500
    p(i) = erlangb_iterative(i*(minutes/60),i);
    if (p(i) < 0.01 && flag == 0)
        flag = 1;
        text = sprintf("Blocking probability < 0.01 for %d servers\n", i);
        fprintf(fd, text);
    endif
endfor

fprintf(fd, "==========\n");

fclose(fd);

figure(1);
stem(p, "linewidth", 2);
titlename = sprintf("Blocking probability for %d minutes", minutes);
title(titlename);
axis([0 40 0 0.1]);
grid on;
xlabel("Servers");
ylabel("Blocking probability");

filename = sprintf("../images/erlangb-%d-zoom.png", minutes);
print("-dpng", filename);
```

# 2 Συστημα εξυπηρετησης με δυο ανομοιους εξυπηρετητες

## 2.1 Ερώτημα 1

Το διάγραμμα ρυθμών μεταβάσεων στην κατάσταση ισορροπίας για το δοσμένο σύστημα φαίνεται παρακάτω για $\lambda = 2, p = 1$ και $\mu_1 = 1.25, \mu_2 = 0.4$.




Οι εξισώσεις του συστήματος για ισορροπία είναι:

$$\lambda P_0 = \mu_1 P_{11} + \mu_2 P_{12} \Rightarrow 2P_0 = 1.25P_{11} + 0.4P_{12} \tag{7}$$

$$(\lambda + \mu_1)P_{11} = p\lambda P_0 + \mu_2 P_2 \Rightarrow 3.25P_{11} = 2P_0 + 0.4P_2 \tag{8}$$

$$(\lambda + \mu_2)P_{12} = (1 - p)\lambda P_0 + \mu_1 P_2 \Rightarrow 2.4P_{12} = 1.25P_2 \tag{9}$$

$$P_0 + P_{11} + P_{12} + P_2 = 1 \tag{10}$$

Επιλύοντας το σύστημα προκύπτουν οι εξής εργοδικές πιθανότητες:

| $P_0$ | $P_{11}$ | $P_{12}$ | $P_2$ | $P_{blocking}$ |
|---|---|---|---|---|
| 0.1387 | 0.1435 | 0.2457 | 0.4719 | 0.4719 |

Η πιθανότητα απόρριψης είναι 0.4719. Ο μέσος αριθμός πελάτων στο σύστημα είναι ίσος με:

$$L = \sum_{k=0}^{2} kP_k = P_{11} + P_{12} + 2P_2 = 0.1435 + 0.2457 + 2 * 0.4719 = 1.333 \tag{11}$$

## 2.2 Ερώτημα 2

Για την συμπλήρωση των thresholds έχουμε:

1. Το πρώτο threshold_1a είναι η πιθανότητα να έχουμε άφιξη ενώ ο πρώτος εξυπηρετητής, ο 1 σε αυτή την περίπτωση, είναι κατειλημμένος.

$$threshold\_1a = \frac{\lambda}{\lambda + \mu_1} \tag{12}$$

2. Το δεύτερο threshold_1b είναι η πιθανότητα να έχουμε άφιξη ενώ ο δεύτερος εξυπηρετητής είναι κατειλημμένος.

$$threshold\_1b = \frac{\lambda}{\lambda + \mu_2} \tag{13}$$

3. Το τρίτο threshold_2_first είναι η πιθανότητα να έχουμε άφιξη ενώ και οι δύο εξυπηρετητές είναι κατειλημμένοι.

$$threshold\_2a = \frac{\lambda}{\lambda + \mu_1 + \mu_2} \tag{14}$$

4. Το τέταρτο threshold_2_second είναι η πιθανότητα να έχουμε άφιξη ή ο πρώτος εξυπηρετητής να απελευθερωθεί. Σε περίπτωση που υπερβούμε αυτό το threshold τότε ο δεύτερος εξυπηρετητής απελευθερώνεται.

$$threshold\_2b = \frac{\lambda + \mu_1}{\lambda + \mu_1 + \mu_2} \tag{15}$$

Η συνθήκη σύγκλισης είναι μεταξύ δύο διαδοχικών μετρήσεων του μέσου αριθμού πελατών να έχουμε απόκλιση μικρότερη του 0.00001 και φαίνεται στον κώδικα στην σειρά 38.

Οι εργοδικές πιθανότητες που υπολογίζει η προσομοίωση φαίνονται παρακάτω και είναι σύμφωνες με τις πιθανότητες που υπολογίσαμε με μικρές αποκλίσεις.

| $P_0$ | $P_{11}$ | $P_{12}$ | $P_2$ | $P_{blocking}$ |
|---|---|---|---|---|
| 0.1392 | 0.1442 | 0.2442 | 0.4724 | 0.4724 |

Ο κώδικας που χρησιμοποιήθηκε για τα παραπάνω είναι ο εξής:

```
clc;
clear all;
close all;


lambda = 2;
m1 = 1.25;
m2 = 0.4;

threshold_1a = lambda/(lambda + m1);
threshold_1b = lambda/(lambda + m2);
threshold_2_first = lambda/(lambda + m1 + m2);
threshold_2_second = (lambda + m1)/(lambda + m1 + m2);

current_state = 0;
arrivals = zeros(1,4);
total_arrivals = 0;
maximum_state_capacity = 2;
previous_mean_clients = 0;
delay_counter = 0;
time = 0;

while 1 > 0
  time = time + 1;

  if mod(time,1000) == 0
```

```
    for i=1:1:4
      P(i) = arrivals(i)/total_arrivals;
    endfor

    delay_counter = delay_counter + 1;

    mean_clients = 0*P(1) + 1*P(2) + 1*P(3) + 2*P(4);

    delay_table(delay_counter) = mean_clients;

    if abs(mean_clients - previous_mean_clients) < 0.00001
       break;
    endif
    previous_mean_clients = mean_clients;
  endif

  random_number = rand(1);

  if current_state == 0
      current_state = 1;
      arrivals(1) = arrivals(1) + 1;
      total_arrivals = total_arrivals + 1;
  elseif current_state == 1
    if random_number < threshold_1a
      current_state = 3;
      arrivals(2) = arrivals(2) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  elseif current_state == 2
    if random_number < threshold_1b
      current_state = 3;
      arrivals(3) = arrivals(3) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  else
      if random_number < threshold_2_first
        arrivals(4) = arrivals(4) + 1;
        total_arrivals = total_arrivals + 1;
      elseif random_number < threshold_2_second
        current_state = 2;
      else
        current_state = 1;
      endif
   endif

endwhile

display(P(1));
display(P(2));
display(P(3));
display(P(4));
```