

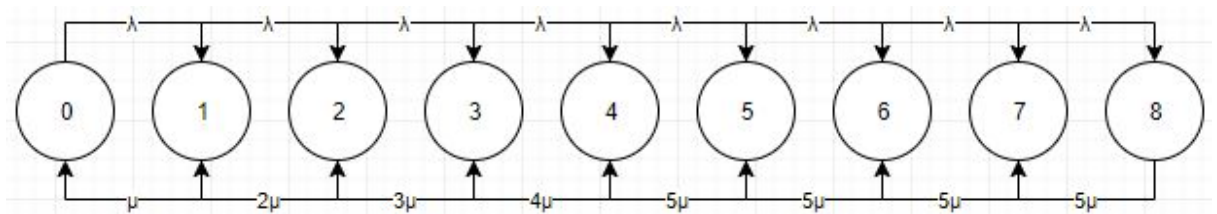
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΕΜΠ

“QUEUEING SYSTEMS”
Ορφανουδάκης Φίλιππος 03113140

4η Ομάδα Ασκήσεων

Σύστημα M/M/N/K (call center)

1) Για $\mu=4$ έχω το παρακάτω διάγραμμα μεταβάσεων :



2)

Για $\lambda=1/4$

```
Lambda = 0.25000
9.3941e-01  5.8713e-02  1.8348e-03  3.8225e-05  5.9726e-07  7.4658e-09  9.3322e-11  1.1665e-12  1.4582e-14
```

Για $\lambda=1$

```
Lambda = 1
Columns 1 through 6:
0.77880073656278  0.19470018414069  0.02433752301759  0.00202812691813  0.00012675793238  0.00000633789662

Columns 7 through 9:
0.00000031689483  0.00000001584474  0.00000000079224
```

3) Αφου το σύστημα μας είναι M/M/5/8 , η πιθανότητα ένας πελάτης να χρειαστεί να περιμένει είναι το εξής άθροισμα :

$$P(5)+P(6)+P(7)+P(8)$$

και τα αποτελέσματα μας είναι :

- για $\lambda=1/4$ **Wait Probability = 0.00000060482**
- για $\lambda=1$ **Wait Probability = 0.00013343**

Όταν χρησιμοποιώ την συνάρτηση erlang-c έχω τα εξής αποτελέσματα :

- για $\lambda=1/4$

```
erlang-c = 7.5603e-09 2.4223e-10 3.2165e-11 7.6647e-12 2.5178e-12 2.5178e-12 2.5178e-12 2.5178e-12
Wait Probability c erlang = 0.000000000010071
```

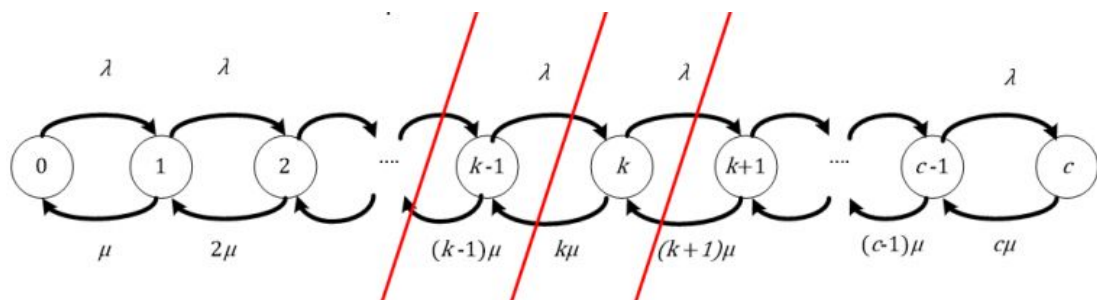
- για $\lambda=1$

```
erlang-c = Columns 1 through 6:
0.00000066714701 0.0000002301852 0.0000000313343 0.0000000075603 0.0000000025022 0.0000000025022
Columns 7 and 8:
0.0000000025022 0.0000000025022
Wait Probability c erlang = 0.000000010009
```

Τα οποία διαφέρουν αρκετά, και αυτό οφείλεται στο γεγονός ότι η φόρμουλα Erlang-c δεν εφαρμόζει απορρίψεις, ένας πελάτης θα περιμένει στην ουρά μέχρι να ελευθερωθεί ένας εξυπηρετητής. Δεν έχουμε αυτό το σύστημα στην συγκεκριμένη περίπτωση καθώς έχουμε απορρίψεις.

Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου

1) Από τις διαφάνειες το διάγραμμα είναι το εξής :



και η $P\{\text{blocking}\}$ δηλαδή η P_c προκύπτει ως εξής από τις διαφάνειες :

Εξισώσεις Ισορροπίας:

$$P_k = \left[\frac{\lambda}{k\mu} \right] P_{k-1} = \left(\frac{\rho^k}{k!} \right) P_0, \quad k = 1, 2, \dots, c \quad \rho \triangleq \frac{\lambda}{\mu} \text{ Erlangs}$$

$$P_0 + P_1 + \dots + P_{c-1} + P_c = 1 \Rightarrow P_0 = \frac{1}{\sum_{k=0}^c \frac{\rho^k}{k!}}$$

$$P_c = P_{\text{blocking}} = \frac{\rho^c / c!}{\sum_{k=0}^c \frac{\rho^k}{k!}} \triangleq B(\rho, c) \text{ (Erlang-B Formula)}$$

- Ο μέσος ρυθμός απωλειών υπολογίζεται ως εξής :

$$\lambda - \gamma = \lambda * P\{\text{blocking}\} = \lambda * P_c$$

- Η συνάρτηση που ζητείτε είναι η εξής :

```
function formul = erlangb_factorial(r, c)
    fac = factorial(c);
    num = r^c/fac;
    paron = 0;
    for i=0:1:c
        paron = paron + r^i/factorial(i);
    endfor
    formul = num / paron;
endfunction
```

2)

Για $n=0$: έχω $B(p,0) = \frac{p^0/0!}{\sum_{k=0}^0 p^k/k!}$

$$\Rightarrow B(p,0) = \frac{1}{1} = 1$$

$$\bullet B(p,n) = \frac{p^n/n!}{\sum_{k=0}^n p^k/k!} = \frac{p^n/n!}{\sum_{k=0}^{n-1} p^k/k! + \frac{p^n}{n!}}$$

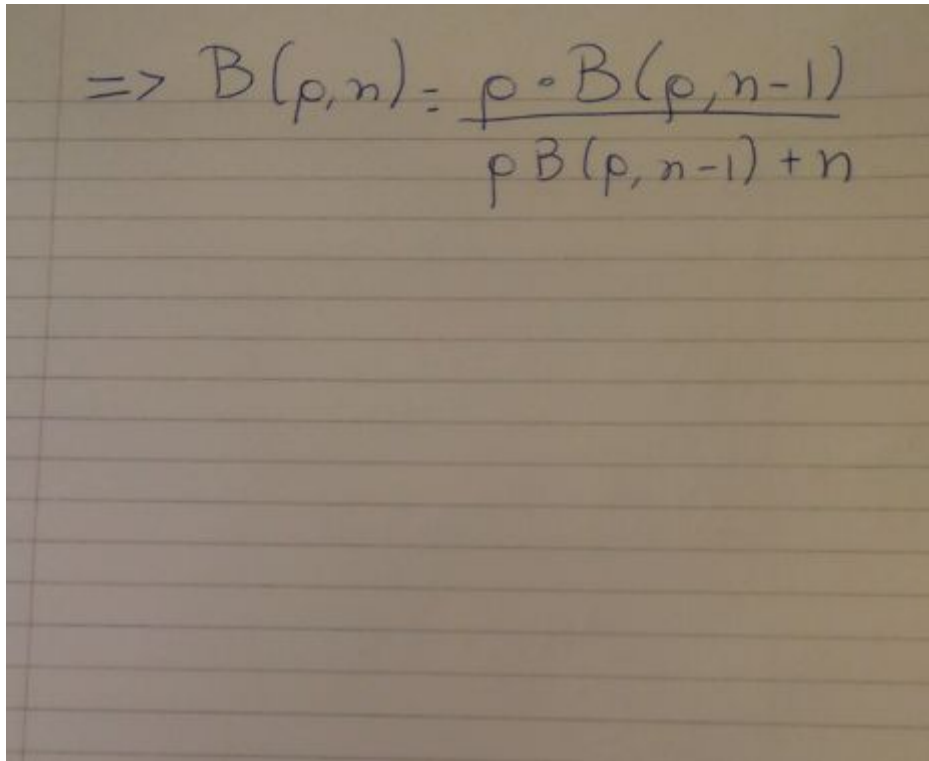
Ομως αν θεωρήσουμε

$$A(p,n) = \frac{1}{B(p,n)} = \frac{\sum_{k=0}^{n-1} \frac{p^k}{k!} + \frac{p^n}{n!}}{\frac{p^n}{n!}}$$

$$= 1 + \frac{\sum_{k=0}^{n-1} \frac{p^k}{k!}}{\frac{p^n}{n!}} = \frac{p}{n} \cdot \frac{p^{n-1}}{(n-1)!}$$

$$= 1 + \frac{n}{p} \cdot A(p,n-1) =$$

$$\Rightarrow \frac{1}{B(p,n)} = 1 + \frac{n}{p \cdot B(p,n-1)}$$



$$\Rightarrow B(r, n) = \frac{r \cdot B(r, n-1)}{r \cdot B(r, n-1) + n}$$

Επιτρέπεται η αντιστροφή του $B(r, n)$ αφού η formula Erlang b προϋποθέτει να υπάρχουν απορρίψεις αρα δεν θα είναι ποτέ ίση με μηδέν η $P\{\text{blocking}\}$.

Η συνάρτηση που υλοποιήσαμε είναι η εξής :

```
function retval = erlangb_iterative(r, c)
    retval = 1;
    counter = 0;
    while (counter < c)
        retval = r*retval / (r*retval + counter + 1);
        counter = counter + 1;
    endwhile
endfunction
```

3)

Το αποτέλεσμα που έχουμε είναι το εξής :

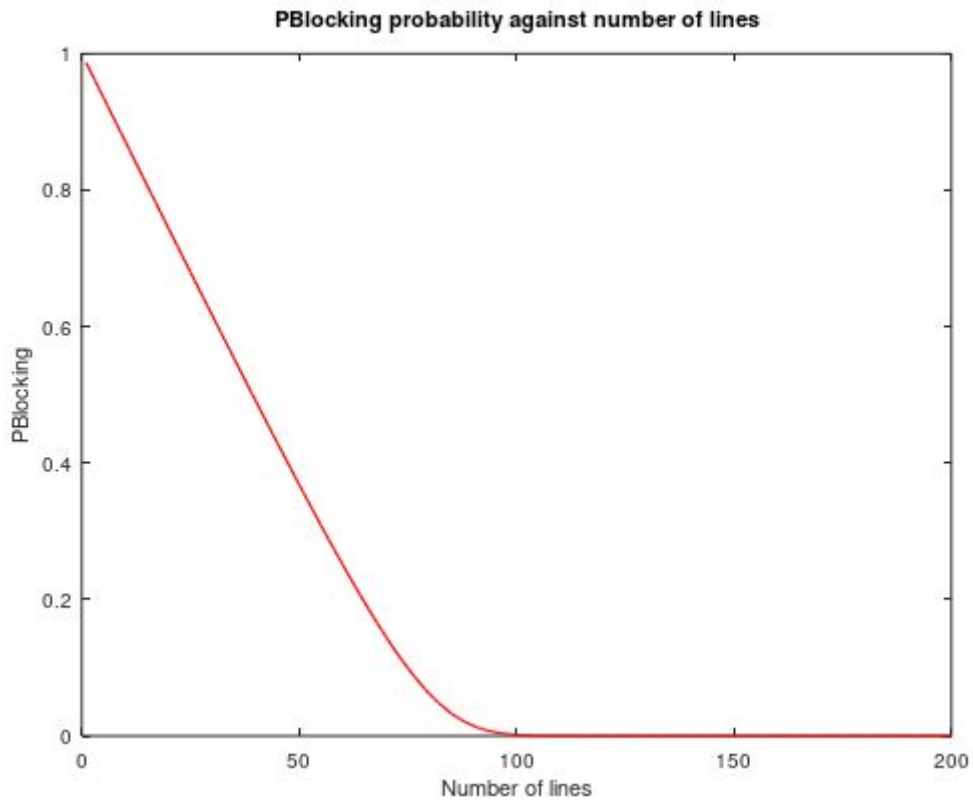
```
erlang_b factorial = NaN
erlang_b iterative = 0.024524
```

Αυτο θα μπορούσε να είναι λόγω της αδυναμίας του παραγοντικού να επιστρέψει τόσο μεγάλες τιμές.

4)

α) Η συνολική ένταση που καλείται να εξυπηρετηθεί είναι $200 \cdot 23 / 60 = 76.67$ Erlangs

β)

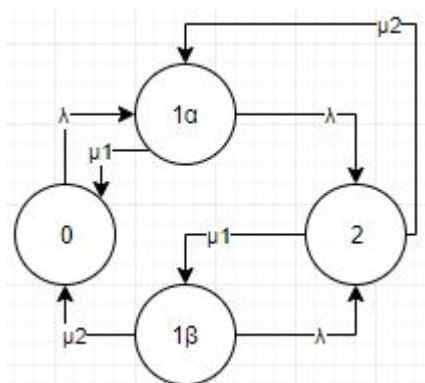


γ) Υπολογίζω για πόσες γραμμές θα έχω $P\{\text{blocking}\} < 0.01$ για πρώτη φορά και έχω σαν αποτέλεσμα 93 γραμμές.

Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές

1)

Το διάγραμμα είναι το εξής :



Οι εργοδικές πιθανότητες θα προκύψουν από τις εξισώσεις ισορροπίας, όπως αναφέρεται και στις διαφάνειες:

Έχω ότι:

$$\bullet \lambda P_0 = \mu_1 P_{1a} + \mu_2 P_{1b} \Rightarrow P_0 = 0,8 P_{1a} + 0,4 P_{1b}$$

$$\bullet (\lambda + \mu_1) P_{1a} = (1 - \rho) \lambda P_0 + \mu_1 P_2$$

$$\bullet (\lambda + \mu_1) P_{1a} = \rho \lambda P_0 + \mu_2 P_2$$

$$\bullet \lambda (P_{1a} + P_{1b}) = (\mu_1 + \mu_2) P_2$$

$$\bullet \text{Οπου } \rho = 1, \lambda = 1, \mu_1 = 0,8, \mu_2 = 0,4$$

και

$$P_0 + P_{1a} + P_{1b} + P_2 = 1$$

Απο αυτό το σύστημα προκύπτει:

$$P_0 = 0,249$$

$$P_{1a} = 0,214$$

$$P_{1b} = 0,195$$

$$P_2 = 0,341$$

$$\bullet P\{\text{Blocking}\} = P_2 = 0,341$$

$$\bullet E\{n(t)\} = 0 * P_0 + 1 * (P_{1a} + P_{1b}) + 2 * P_2 = 1,092$$

3) Τα αποτελέσματα της προσομοίωσης είναι τα εξής :

```
P(0)= 0.25108  
P(1a)= 0.21479  
P(1b)= 0.19124  
P(2)= 0.34289  
P(Blocking)= 0.34289  
Mean Number of Clients= 1.0918  
>> |
```

Ο κώδικας είναι ο εξής :

```
clc;  
clear all;  
close all;  
  
pkg load queueing;  
  
#M/M/N/K system  
lambda = [ones(1,8)*0.25;ones(1,8)];  
mu = ones(1,8)*4.*[1,2,3,4,5,5,5,5];  
  
for i=1:length(lambda(:,1))  
    Q = ctmcbsd(lambda(i, :), mu);  
    p = ctmc(Q);  
    fprintf("Lambda ="); disp(lambda(i,1));  
    disp(p);  
    fprintf("Wait Probability ="); disp(p(5)+p(6)+p(7)+p(8));  
    c = erlangc (lambda(i, :) ./ mu, 5);  
    fprintf("erlang-c ="); disp(c);  
    fprintf("Wait Probability c erlang ="); disp(c(5)+c(6)+c(7)+c(8));  
    disp("");  
endfor
```

```
function formul = erlangb_factorial(r, c)  
    fac = factorial(c);  
    num = r^c/fac;  
    paron = 0;  
    for i=0:1:c  
        paron = paron + r^i/factorial(i);  
    endfor  
    formul = num / paron;
```



```
endfunction
```

```
function retval = erlangb_iterative(r, c)
    retval = 1;
    counter = 0;
    while (counter < c)
        retval = r*retval / (r*retval + counter + 1);
        counter = counter + 1;
    endwhile
endfunction
```

```
fprintf("erlang_b factorial =");disp(erlangb_factorial(1024,1024)) ;
fprintf("erlang_b iterative =");disp(erlangb_iterative(1024,1024));
```

```
#Call Center design and analysis
```

```
Cs = 1:1:200;
for i=1:1:length(Cs)
    Pblock(i) = erlangb_iterative(200*23/60, i);
endfor
```

```
colors = "rbgm";
figure(1);
plot(Cs,Pblock,colors(1),"linewidth",1.2);
title("P{Blocking} probability against number of lines");
xlabel("Number of lines");
ylabel("P{Blocking}");
```

```
for i=1:1:length(Cs)
    if(Pblock(i) < 0.01)
        fprintf("PBlocking drops below 0.01 at c ="); disp(i);
        break;
    endif;
endfor
```

```
#2 different servers
```

```
total_arrivals = 0; % to measure the total number of arrivals
rejected_arrivals = 0; % to measure teh number of clients that were blocked
current_state = 0; % holds the current state of the system
previous_mean_clients = 0; % will help in the convergence test
index = 0;
```

```
lambda = 1;
mu1 = 0.8;
```

```

mu2 = 0.4;
#threshold0 = lambda/(lambda + mu1 + mu2); % the threshold used to calculate probabilities
threshold1a = lambda/(lambda + mu1);
threshold1b = lambda/(lambda + mu2);
threshold2 = lambda/(lambda + mu1 + mu2);
threshold2dep = mu1/(mu1 + mu2);

transitions = 0; % holds the transitions of the simulation in transitions steps
arrivals = [0,0,0,0];

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:length(arrivals)
            P(i) = arrivals(i)/total_arrivals; % calculate the probability of every state in the system
        endfor

        Pblock = rejected_arrivals / total_arrivals;

        mean_clients = 1*P(2) + 1*P(4) + 2*P(3); % calculate the mean number of clients in the
        system

        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
        total_arrivals = total_arrivals + 1;
        arrivals(current_state + 1) = arrivals(current_state + 1) + 1; % increase the number of
        arrivals in the current state
        current_state = 1; %s1 = 1 client in A
        %if(transitions < 30)
        %fprintf("current state is 0 and next is 1, ");
        %endif
        continue;
    end
end

```

```

endif
if current_state == 1
    if random_number < threshold1a %arrival
        total_arrivals = total_arrivals + 1;
        arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
        current_state = 2; %s2 = 2 clients
        %if(transitions < 30)
            %fprintf("current state is 1 and next is 2, "); disp(random_number);
        %endif
    else % departure
        current_state = 0; %s0 = empty
        %if(transitions < 30)
            %fprintf("current state is 1 and next is 0, "); disp(random_number);
        %endif
    endif
    continue;
endif
if current_state == 2
    if random_number > threshold2 %departure
        %if(transitions < 30)
            %fprintf("current state is 2 and departure, "); disp(random_number);
        %endif
        random_number = rand(1);
        if random_number < threshold2dep %A empties
            current_state = 3; %s3 = 1 client in B
            %if(transitions < 30)
                %fprintf("next is 3, (<"); disp(random_number); disp(threshold2dep);
            %endif
        else %B empties
            current_state = 1; %s1 = 1 client in A
            %if(transitions < 30)
                %fprintf("next is 1, "); disp(random_number); disp(threshold2dep);
            %endif
        endif
    else % arrival
        total_arrivals = total_arrivals + 1;
        rejected_arrivals = rejected_arrivals + 1;
        arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
        %if(transitions < 30)
            %fprintf("current state is 2 and next is 2"); disp(random_number);
        %endif
        % current_state stays the same
    endif
    continue;
endif
if current_state == 3

```

```

if random_number < threshold1b %arrival
    total_arrivals = total_arrivals + 1;
    arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
    current_state = 2; %s2 = 2 clients
    %if(transitions < 30)
        %fprintf("current state is 3 and next is 2"); disp(random_number);
    %endif
else % departure
    current_state = 0; %s0 = empty
    %if(transitions < 30)
        %fprintf("current state is 3 and next is 0"); disp(random_number);
    %endif
endif
continue;
endif

endwhile

fprintf("P(0)="); disp(P(1));
fprintf("P(1a)="); disp(P(2));
fprintf("P(1b)="); disp(P(4));
fprintf("P(2)="); disp(P(3));
fprintf("P(Blocking)="); disp(Pblock);
fprintf("Mean Number of Clients="); disp(mean_clients);

```