Συστήματα Αναμονής

(Queuing Systems)

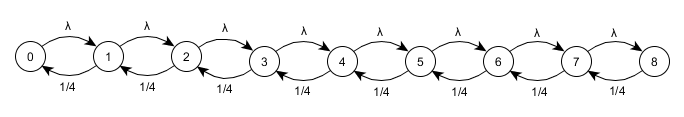
4η Εργαστηριακή Άσκηση

# Λεούσης Σάββας

# Α.Μ.: 03114945

Σύστημα Μ/Μ/Ν/Κ (call center)

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



1. 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:
   1. Για λ=1/4



* 1. Για λ=1



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



* 1. Για λ=1



# Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου

# 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

# 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

# 

# 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.

# Η συνολική ένταση του φορτίου που καλείται να εξυπηρετήσει το τηλεφωνικό δίκτυο της εταιρείας είναι

# 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.

# 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

# Το διάγραμμα ρυθμών μεταβάσεων του συστήματος είναι το παρακάτω:

# 

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

# Τελικά προκύπτουν οι εξής εργιδικές πιθανότητες:

# Η πιθανότητα απόρριψης πελάτη από το σύστημα είναι ίση με:

# 

# Εκτελώντας την προσωμοίωση του παραπάνω συστήματος, προκύπτουν οι παρακάτω εργοδικές πιθανότητες, οι οποίες συγκλίνουν αρκετά με τις θεωρητικές:

# 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);