Συστήματα Αναμονής

(Queuing Systems)

6η Εργαστηριακή Άσκηση

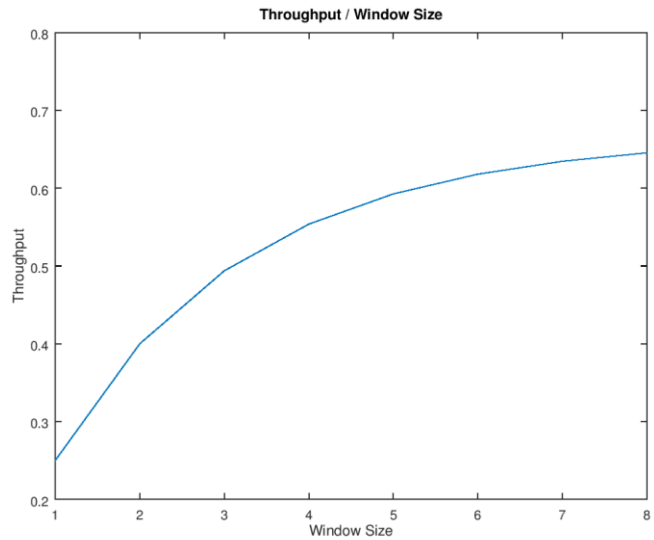
Λεούσης Σάββας

Α.Μ.: 03114945

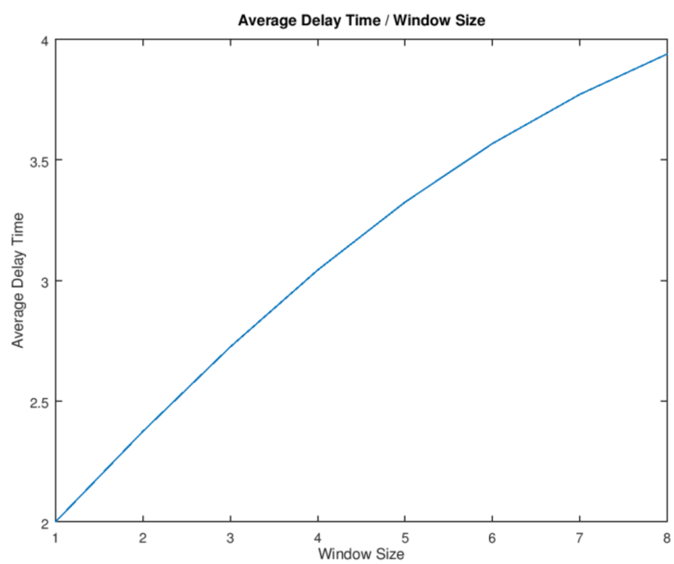
Μηχανισμός Ελέγχου Ροής Παραθύρου

1.

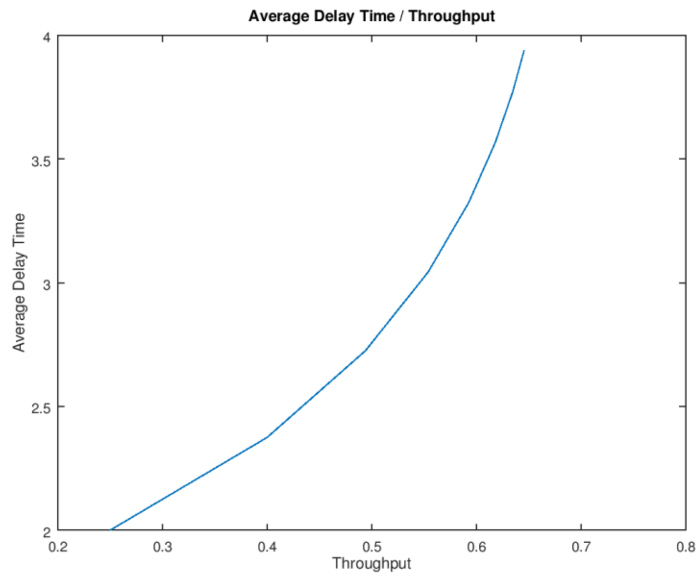
- Η ρυθμαπόδοση (throughput) του συστήματος ως συνάρτηση του αριθμού των πακέτων στο σύστημα είναι η παρακάτω:



- Ο μέσος χρόνος καθυστέρησης του συστήματος από το S μέχρι το D ως συνάρτηση του αριθμού των πακέτων στο σύστημα είναι ο παρακάτω:



- Ο μέσος χρόνος καθυστέρησης του συστήματος από το S μέχρι το D ως συνάρτηση της ρυθμαπόδοσης του συστήματος είναι ο παρακάτω:



2. Παρατηρούμε ότι η χρησιμοποίηση και ο μέσος αριθμός πελατών μένουν σταθερά, ο μέσος χρόνος καθυστέρησης μειώνεται όσο μεγαλώνει το k, και το throughput αυξάνεται.

Ο αλγόριθμος του Buzen

1. Είναι:

$$\mu_1 X_1 = (1-p)\mu_1 X_1 + \mu_2 X_2$$
$$\mu_2 X_2 = p\mu_1 X_1$$

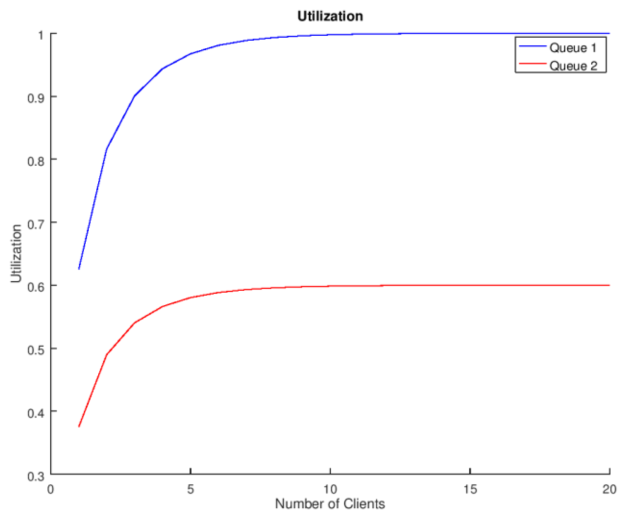
Άρα $X_1 = 1, X_2 = 0{,}6$

2. Η ζητούμενη συνάρτηση buzen είναι η παρακάτω:

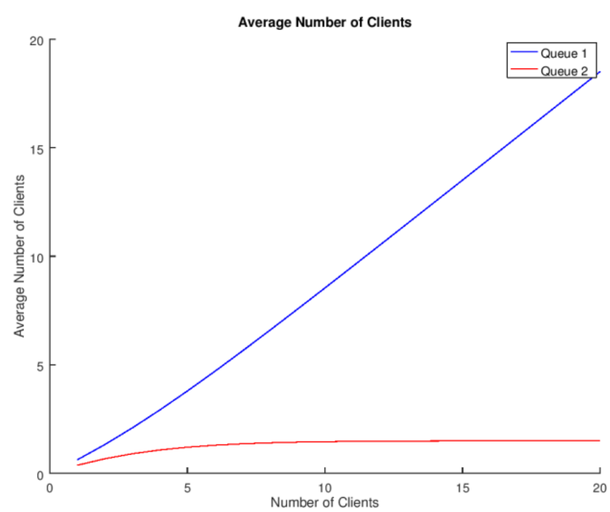
```
function G = buzen(N,M,X)
  C(1)=1;
  for n=2:N+1
    C(n)=0;
  endfor
  for m=1:M
    for n=2:N+1
      C(n)  = C(n)+X(m)*C(n-1)
    endfor
  endfor
  G=C(N+1);
Endfunction
```

3.

- Ο βαθμός χρησιμοποίησης (utilization) των δύο ουρών ως συνάρτηση του αριθμού πελατών σε κοινό διάγραμμα αξόνων δίνεται παρακάτω:

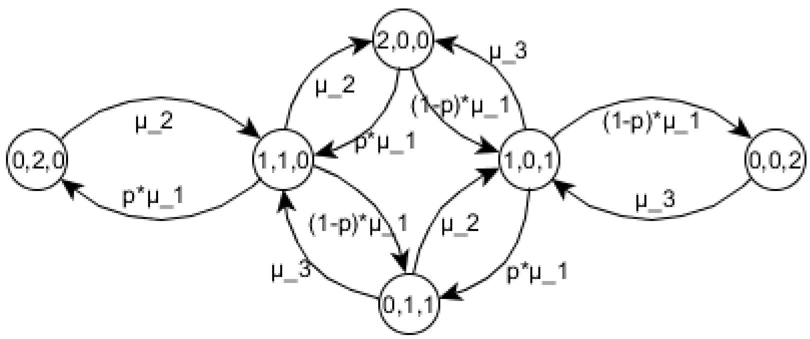


- Ο μέσος αριθμός πελατών στις δύο ουρές ως συνάρτηση του αριθμού πελατών σε κοινό διάγραμμα αξόνων δίνεται παρακάτω:



<u>Προσομοίωση σε κλειστό δίκτυο εκθετικών ουρών αναμονής</u>

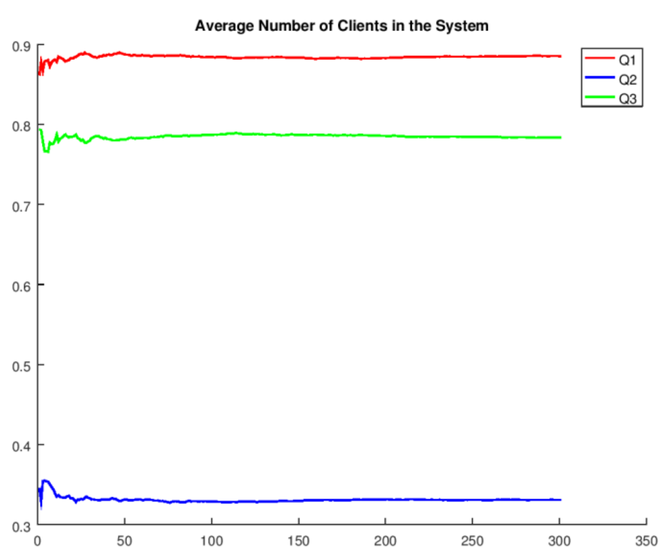
Το διάγραμμα ρυθμών μεταβάσεων του συστήματος στην κατάσταση ισορροπίας είναι το παρακάτω:



όπου (n_1, n_2, n_3) η κατάσταση του συστήματος.

1. Οι εργοδικές πιθανότητες είναι οι εξής:
   - $P(2,0,0) = 0.24755$
   - $P(0,2,0) = 0.055783$
   - $P(0,0,2) = 0.19663$
   - $P(1,1,0) = 0.10806$
   - $P(1,0,1) = 0.28096$
   - $P(0,1,1) = 0.11101$
2. Τρέχοντας την προσομοίωση, προκύπτουν τα παρακάτω αποτελέσματα και η γραφική παράσταση:

```
 0.88589
 0.32864
 0.78547
sum =  2.0000
```



Εδώ μπορούμε να παρατηρήσουμε ότι το άθροισμα όλων των μέσων όρων είναι 2, όσο και ο συνολικός αριθμός των πελατών στο σύστημα.

Παράρτημα (κώδικας Lab6.m)

```
clc;
clear all;
close all;

##### WINDOW FLOW CONTROL MECHANISM #####

#1#

P = [0 1 0 0 0;
     0 0 1 0 0;
     0 0 0 1 0;
     0 0 0 0 1;
     1 0 0 0 0;];             # Transition probability matrix
S = [1 1/2 1/2 1/2 3/2]
m = ones(size(S));           # All centers are single-server
Z = 0;                        # External delay
N = 8;                        # Maximum population to consider
V = qncsvisits(P);            # Compute number of visits
for n=1:N
  [U R Q X] = qnclosed( n, S, V, m, Z );
  Throughput(n)=X(1);
  E_T_sd(n) = (Q(1)+Q(2)+Q(3))/X(1);
endfor
figure(1);
plot(Throughput);
xlabel("Window Size");
ylabel("Throughput");
title("Throughput / Window Size");
figure(2);
plot(E_T_sd);
xlabel("Window Size");
ylabel("Average Delay Time");
title("Average Delay Time / Window Size");
figure(3);
plot(Throughput,E_T_sd);
xlabel("Throughput");
ylabel("Average Delay Time");
title("Average Delay Time / Throughput");

#2#

P = [0 1 0 0 0;
     0 0 1 0 0;
```

```
     0 0 0 1 0;
     0 0 0 0 1;
     1 0 0 0 0;];              # Transition probability matrix
S = [1 1/2 1/2 1/2 3/2];       # Average service times
m = ones(size(S));             # All centers are single-server
Z = 0;                         # External delay
N = 8;                         # Maximum population to consider
V = qncsvisits(P);             # Compute number of visits
for k=1:5
  display(k);
  S=S/k;
  [U R Q X] = qnclosed( N, S, V, m, Z );
  display(U);
  e_t_sd=Q/X;
  display(e_t_sd);
  display(Q);
  display(X);
  display("\n");
  S = [1 1/2 1/2 1/2 3/2];
endfor

##### BUZEN ALGORITHM #####

#2#

function G = buzen(N,M,X)
  C(1)=1;
  for n=2:N+1
    C(n)=0;
  endfor
  for m=1:M
    for n=2:N+1
      C(n) = C(n)+X(m)*C(n-1);
    endfor
  endfor
  G=C(N+1);
endfunction

#3#

  #a#

X=[1 0.6]
for N=1:20
  U_1(N)=X(1)*buzen(N-1,2,X)/buzen(N,2,X);
  U_2(N)=X(2)*buzen(N-1,2,X)/buzen(N,2,X);
endfor
figure(4);
hold on;
plot(U_1,'b');
plot(U_2,'r');
legend("Queue 1","Queue 2");
```

```
xlabel("Number of Clients");
ylabel("Utilization");
title("Utilization");
hold off;

  #b#

X=[1 0.6]
for N=1:20
  E_1(N)=0;
  E_2(N)=0;
  for k=1:N
    E_1(N)=E_1(N)+(X(1)^k)*(buzen(N-k,2,X)/buzen(N,2,X));
    E_2(N)=E_2(N)+(X(2)^k)*(buzen(N-k,2,X)/buzen(N,2,X));
  endfor
endfor

figure(5);
hold on;
plot(E_1,'b');
plot(E_2,'r');
legend("Queue 1","Queue 2");
xlabel("Number of Clients");
ylabel("Average Number of Clients");
title("Average Number of Clients");
hold off;

##### SIMULATION OF CLOSED NETWORK OF EXPONENTIAL QUEUES #####

#2#

mu1 = 2; % queue 1
mu2 = 3; % queue 2
mu3 = 4; % queue 3
p= 0.4;

arrivals(200) = 0;
arrivals(020) = 0;
arrivals(002) = 0;
arrivals(110) = 0;
arrivals(101) = 0;
arrivals(011) = 0;
total_arrivals = 0;

% threshold definition
threshold = mu1/(mu1 + mu2 + mu3);
% system starts at state 3
current_state = 200;
% count the time steps of the simulation
steps = 0;

previous_mean1 = 0;
```

```
previous_mean2 = 0;
previous_mean3 = 0;

% times checked for convergence
times = 0;

while true
  steps = steps + 1;
  % every 1000 steps check for convergence
  if mod(steps,1000) == 0
    times = times + 1;

    % total time in every state
    T200 = 1/mu1 * arrivals(200);
    T020 = 1/mu2 * arrivals(020);
    T002 = 1/mu3 * arrivals(002);
    T110 = 1/(mu1+mu2) * arrivals(110);
    T101 = 1/(mu1+mu3) * arrivals(101);
    T011 = 1/(mu2+mu3) * arrivals(011);

    % total time in all states
    total_time = T200 + T020 + T002 + T110 + T101 + T011;
    % Probability of every state
    P(200) = T200/total_time;
    P(020) = T020/total_time;
    P(002) = T002/total_time;
    P(110) = T110/total_time;
    P(101) = T101/total_time;
    P(011) = T011/total_time;

    % mean number of clients in queues 1 and 2
    current_mean1 = P(200)*2 + P(110) + P(101);
    current_mean2 = P(020)*2 + P(110) + P(011);
    current_mean3 = P(002)*2 + P(101) + P(011);

    clients_1(times) = current_mean1;
    clients_2(times) = current_mean2;
    clients_3(times) = current_mean3;

    % check both queues for convergence
    if (abs(current_mean1 - previous_mean1)<0.00001 &&
abs(current_mean2 - previous_mean2) < 0.00001 &&
abs(current_mean3 - previous_mean3) < 0.00001) || (steps >
300000)
      break;
    endif

    previous_mean1 = current_mean1;
    previous_mean2 = current_mean2;

  endif
```

```
  arrivals(current_state) = arrivals(current_state) + 1;
  total_arrivals = total_arrivals + 1;

  % get a random number from uniform distribution
  random_number = rand(1);
  random = rand(1);
  if current_state == 200 #1
    if random < p
      current_state = 110;
    else
      current_state = 101;
    endif
  elseif current_state == 101 #2
    if random_number < threshold
      current_state = 200;
    else
      if random < p
        current_state = 011;
      else
        current_state = 002;
      endif
    endif
  elseif current_state == 110 #3
    if random_number < threshold
      current_state = 200;
    else
      if random < p
        current_state = 020;
      else
        current_state = 011;
      endif
    endif
  elseif current_state == 011 #4
    if random_number < threshold
      current_state = 110;
    else
      current_state = 101;
    endif
  elseif current_state == 020 #5
    current_state = 110;
  elseif current_state == 002 #6
    current_state = 101;
  endif
endwhile

display(clients_1(end));
display(clients_2(end));
display(clients_3(end));

sum=clients_1(end)+clients_2(end)+clients_3(end);
display(sum);
figure(6);
```

```
 hold on;
 plot(clients_1,'r',"linewidth",1.3);
 plot(clients_2,'b',"linewidth",1.3);
 plot(clients_3,'g',"linewidth",1.3);
 title("Average Number of Clients in the System");
 legend("Q1","Q2","Q3");
hold off;
```