Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Ακ. έτος 2020-2021, 6ο Εξάμηνο: Συστήματα Αναμονής

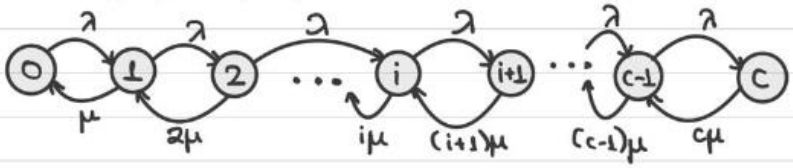
Παναγιώτα-Μικαέλα Ξυλιά
ΑΜ:03118859

4η Ομάδα Ασκήσεων

Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου

ουρά M/M/c/c
αφίξεις με κατανομή Poisson, μέσος ρυθμός λ,
εξυπηρετήσεις με ομοιόμορφο μέσο ρυθμό μ,

1.(1) Διάγραμμα ρυθμού μεταβάσεων



ισχύει $k\mu P_k = \lambda P_{k-1} \Rightarrow P_k = \frac{\lambda}{k\mu} P_{k-1} \Rightarrow P_k = P_{k-1} \frac{\rho}{k}$ $k=1,2,\dots,c$

$k=1: P_1 = \rho P_0$
$k=2: P_2 = \frac{\rho}{2} P_1 = \frac{\rho^2}{2!} P_0$
$k=3: P_3 = \frac{\rho}{3} P_2 = \frac{\rho^3}{3!} P_0$

} Συνεπώς $P_k = \frac{\rho^k}{k!} P_0$

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

$P_{rejecting} = P_c = \frac{\rho^c}{c!} P_0$

$P_{blocking} = \frac{\frac{\rho^c}{c!}}{\sum_{i=0}^{c} \frac{\rho^k}{k!}}$

Μέσος ρυθμός απωλειών $= \lambda P_{blocking}$

Συνάρτηση στο Octave:

```
pkg load queueing
addpath(pwd);

function result = erlang_factorial(ro,c)
  arithm=(ro^c)/factorial(c);
  paranom=0;
  k=0;
  while(k<=c)
  paranom+=(ro^k)/factorial(k);
  k++;
  endwhile
  result = arithm/paranom;
endfunction
display(erlang_factorial(1024,1024));
display(erlangb(1024,1024));
```

(2) Συνάρτηση στο Octave:

```
addpath(pwd);
function result = erlang_iterative(ro,n)
  k=0;
  result = 1;
  while k<=n
    result = ro * result / (ro*result + k);
    k++
  endwhile
endfunction
display(erlang_iterative(1024,1024));
display(erlangb(1024,1024));
```

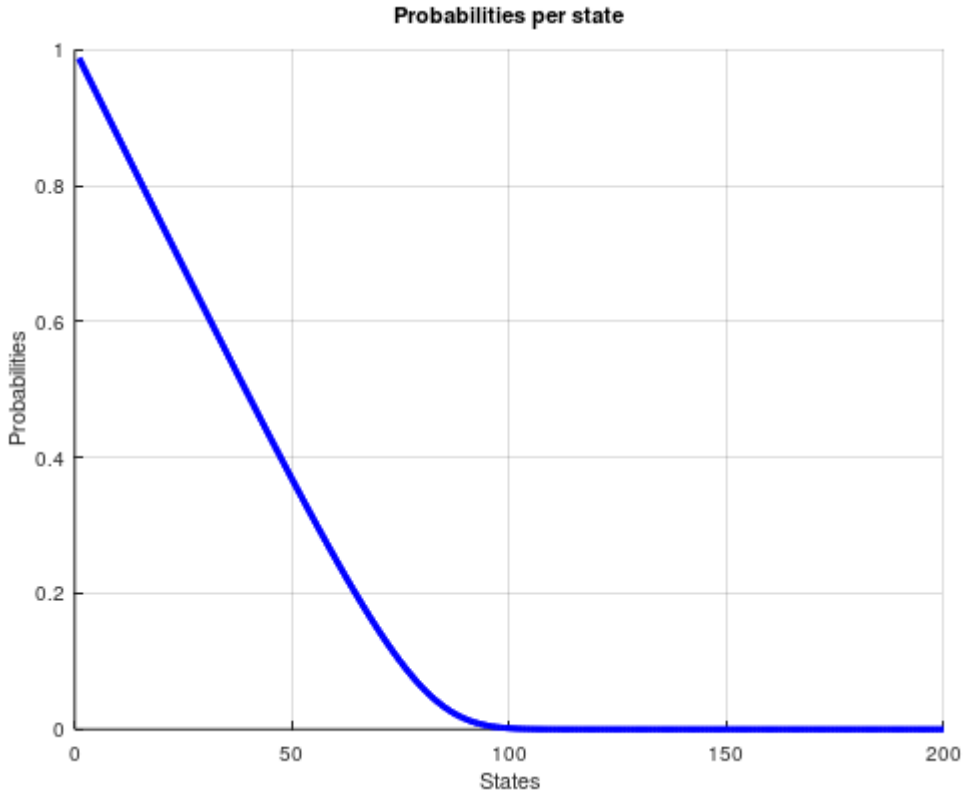
(3) Η erlang_iterative μπορεί να υπολογίσει το αποτέλεσμα σε αντίθεση με την factorial , καθώς για μεγάλους αριθμούς όπως ο 1024^1024 έχουμε σφάλμα. Επομένως ως έξοδο παίρνουμε την τιμή NaN.

```
>> ask4_1

NaN
0.024524
```

(4)α) Χρησιμοποιώντας ως πρότυπο τον πιο απαιτητικό χρήστη ο οποίος μιλάει 23 λεπτά την ώρα έχουμε ένταση φορτίου ρ =200*23/60=76,67 erlangs.

β)



γ) Βρίσκουμε πως θα χρησιμοποιηθούν 94 γραμμές

```
lines = 94
```

Κώδικας:

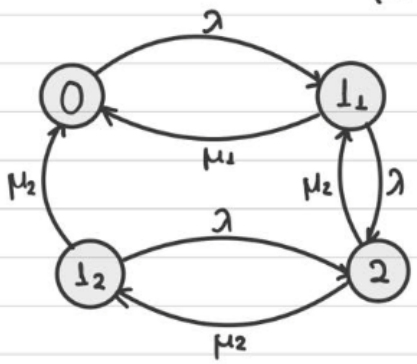
```
addpath(pwd);
function result = erlang_iterative(ro,n)
  k=0;
  result = 1;
  while k<=n
    result = ro * result / (ro*result + k);
    k++
  endwhile
endfunction

display(erlang_iterative(1024,1024));
display(erlangb(1024,1024));
ro=200*23/60;
c=1:200;
for i=1:200
  erl(i)=erlang_iterative(ro,i)
endfor
figure(1);
hold on;
title("Probabilities per state")
xlabel("States")
ylabel("Probabilities")
plot(c,erl,"b","linewidth",1.5);
grid on;
hold off;
P=1;
lines=0;
```

```
while(P>0.01)
P=erlang_iterative(ro,lines);
lines++
endwhile
display(lines);
```

Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές

2.(1) $\lambda = 1$ πελ/s , $\frac{1}{\mu_1} = 1{,}25$s , $\frac{1}{\mu_2} = 2{,}5$s



$\lambda P_0 = \mu_1 P_{1_1} + \mu_2 P_{1_2} \Rightarrow P_0 = 0{,}8 P_{1_1} + 0{,}4 P_{1_2}$ (1)

$\mu_2 P_2 + \mu_1 P_2 = \lambda P_{1_1} + \lambda P_{1_2} \Rightarrow P_2 = \frac{5}{6}(P_{1_1} + P_{1_2})$ (2)

$\mu_1 P_{1_1} + \lambda P_{1_1} = \lambda P_0 + \mu_2 P_2 \Rightarrow P_{1_1} = \frac{5}{9} P_0 + \frac{2}{9} P_2$ (3)

$\mu_2 P_{1_2} + \lambda P_{1_2} = \mu_1 P_2 \Rightarrow P_{1_2} = \frac{4}{7} P_2$ (4)

$P_0 + P_{1_1} + P_{1_2} + P_2 = 1 \overset{(1)(2)(3)(4)}{\Rightarrow}$

$P_0 \simeq 0{,}25$

$P_{1_1} \simeq 0{,}215$

$P_{1_2} \simeq 0{,}195$

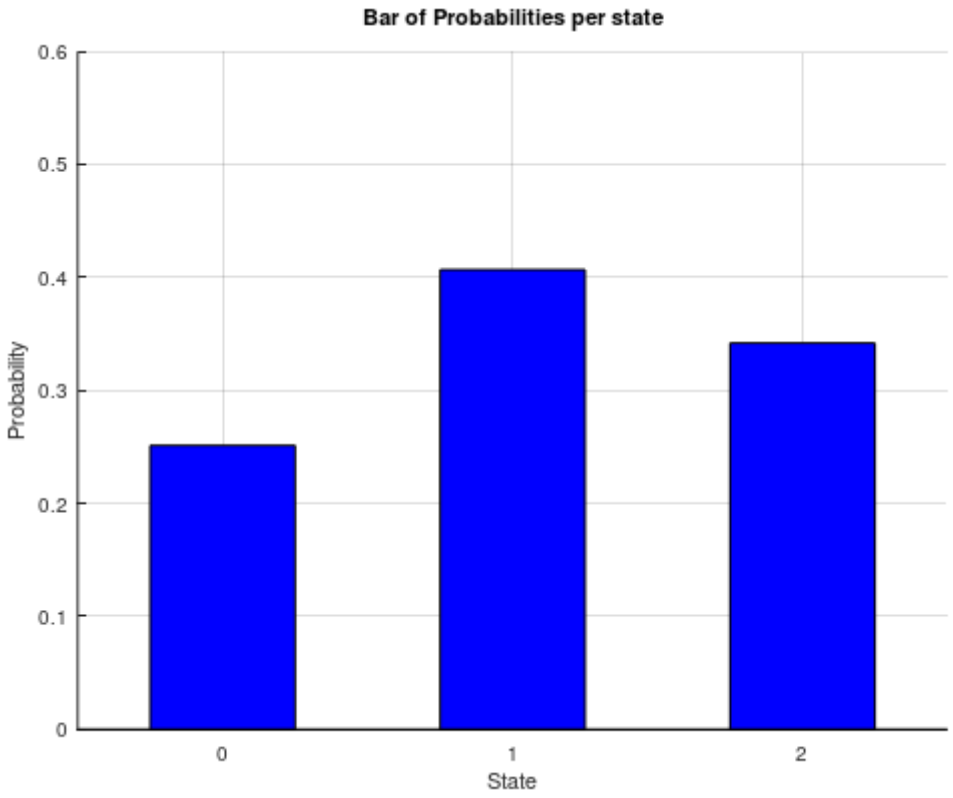
$P_2 \simeq 0{,}34$

$P_2 = P_{blocking}$

Μέσος αριθμός πελατών στο σύστημα:

$$E[n(t)] = \sum_{k=0}^{2} kP_k = 0 \cdot P_0 + 1(P_{1_1} + P_{1_2}) + 2P_2 \simeq 1{,}09$$

(2)



Οι πιθανότητες που υπολογίζει το πρόγραμμα είναι περίπου ίσες με αυτές που υπολογίσαμε παραπάνω:

```
0.2513
0.2132
0.1936
0.3419
```

Κριτήρια σύγκρισης:

```
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);
```

Κώδικας:

```
lambda = 1;
m1 = 0.8;
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));

figure(1);
hold on;
title("Bar of Probabilities per state")
xlabel("State")
ylabel("Probability")
bar([0,1,2], [P(1),P(2)+P(3),P(4)], "b", 0.5);
xticks ([0,1,2]);
grid on;

hold off;
```