دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

تمرینهای درس شبکه پیشرفته

استاد: دکتر احمد خونساری

مهران صفایانی 83203873

وحید مواجی 83205947

بهمن عرب رضایی 83204234

حسن تکابی 83210022

تابستان 1384

تمرین 1

احتمال ورود n مشتری به صورت زیر تعریف می‌شود:

For $1 \le n \le 10$:

$$P(\tilde{x} = n) = P_x(1 - P_x)^{n-1}$$
$$P(\tilde{t} = n) = P_t(1 - P_t)^{n-1}$$

Else:

$$P(\tilde{x} = n) = P(\tilde{t} = n) = 0$$

$$P_x = 0.292578$$
$$P_y = 0.14358$$

برنامه‌ای بنویسید که 20 متغیر تصادفی برای $\tilde{x}, \tilde{t}$ تولید کند و سپس زمان انتظار مجازی U(t) را به عنوان تابعی از t رسم کند. برای $1 \le n \le 20$ نیز U(n) و W(n) را بدست آورید. اثبات کنید که W(n) را می‌توان از روی U(t) بدست آورد.

x، یک متغیر تصادفی هندسی است که بیانگر تعداد مشتری در سیستم است.
t، یک متغیر تصادفی هندسی است که بیانگر زمان ورود مشتری در سیستم است.

اگر x یك متغیر تصادفی هندسی باشد ما طبق مراحل زیر می‌توانیم متغیر تصادفی تولید کنیم :

1. تولید متغیر تصادفی بین (0,1)
2. X=j است اگر $1-(1-p)^{j-1} < U \le 1-(1-p)^{j}$

به علت اینکه بایستی X بین 1 و 10 باشد پس بایستی متغیر تصادفی تولید شده در 1 در بازه $(0, 1-(1-p)^{10})$ باشد .

$$(1-p)^{j-1} > 1-U \ge (1-p)^{j}$$
$$(j-1)\log_{1-p}(1-p) < \log_{1-p} 1-U \le j\log_{1-p}(1-p)$$
$$j-1 < \log_{1-p} 1-U \le j$$
$$X = \text{int}\left(\frac{\log(1-U)}{\log(1-p)}\right)+1$$

فرض می‌شود که زمان سرویس هر مشتری یك است یعنی اگر در یك لحظه 5 مشتری به سیستم وارد شود زمان سرویس آنها 5 است.

با توجه به فرمولهای بالا برنامه MATLAB زیر برای محاسبه w(n) , u(n) نوشته شده است.

```
clear;
% 20 geometric random variable for x
n=10;
p=.292578;
b=1-(1-p)^n;
%b=0.968610;
x=b*rand(20,1);
X=floor(log(1-x)/log(1-p))+1;

% 20 geometric random variable for t
pt=.14358;
bt=1-(1-pt)^n;

%bt=0.787740;
t=bt*rand(20,1);
T=floor(log(1-t)/log(1-pt))+1;
m=cat(2,T,X);
N(1:20)=0;
for i=1:20
    for jj=1:20
        if (m(jj,1)==i)
            N(i)=N(i)+m(jj,2);
        end
    end
end
u(1:20)=0;
for i=1:20
    for j=1:i
        u(i)=u(i)+N(j);
    end;
    if (u(i)<=(i-1))
        u(i)=0;
    else
        u(i)=u(i)-(i-1);
    end
end;
plot(u,'-bo');

wn(1:20)=0;
wn=u-N;
hold on;
xlabel('U(n),W(n)');
ylabel('t');

plot(wn,'-.r')
h=legend('U(n)','W(n)',2);

hold off;
```

شکل 1 – 1: برنامه محاسبه کننده w(n), u(n)

در جدول 1- 1 اعداد تصادفی بدست آمده از يك بار اجرای برنامه نشان داده شده است. در شکل 1-2 U(n) و w(n)نشان داده شده است.برای حالتی که $1 \le n \le 20$ است کافی‌است در برنامه بالا مقدار n را در اول برنامه برابر با 20 قرار دهیم که در جدول 1-2 و شکل 1-3 نتایج اجرای برنامه برای این حالت نشان داده شده است.
W(n) را می‌توان به صورت زیر محاسبه کرد:

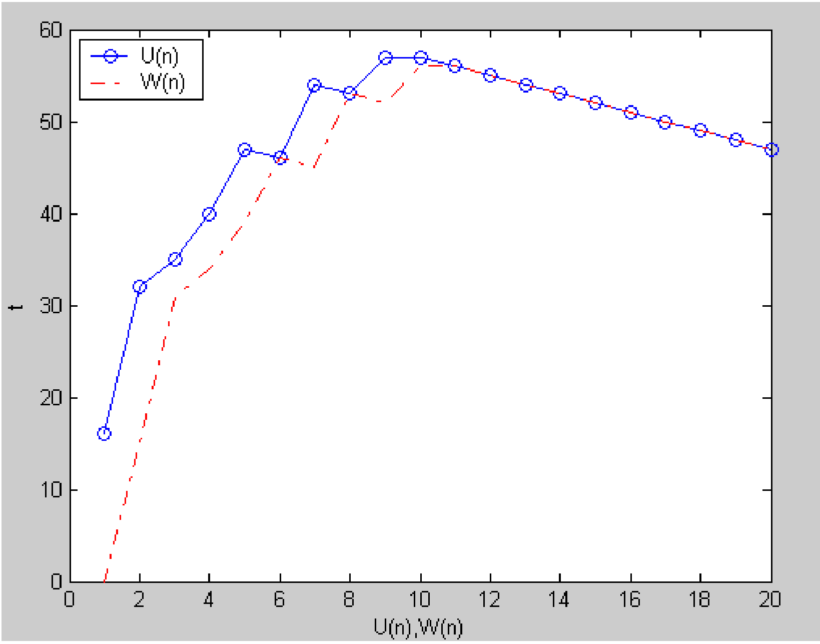
W(n)=U(n)-N(n)

که در آن N(n) تعداد مشتری است که در لحظه n وارد سیستم می‌شود .

جدول 1-1: متغیر های تصادفی تولید شده برای x و t برای حالت n=10

متغیر های تصادفی t	متغیر های تصادفی x

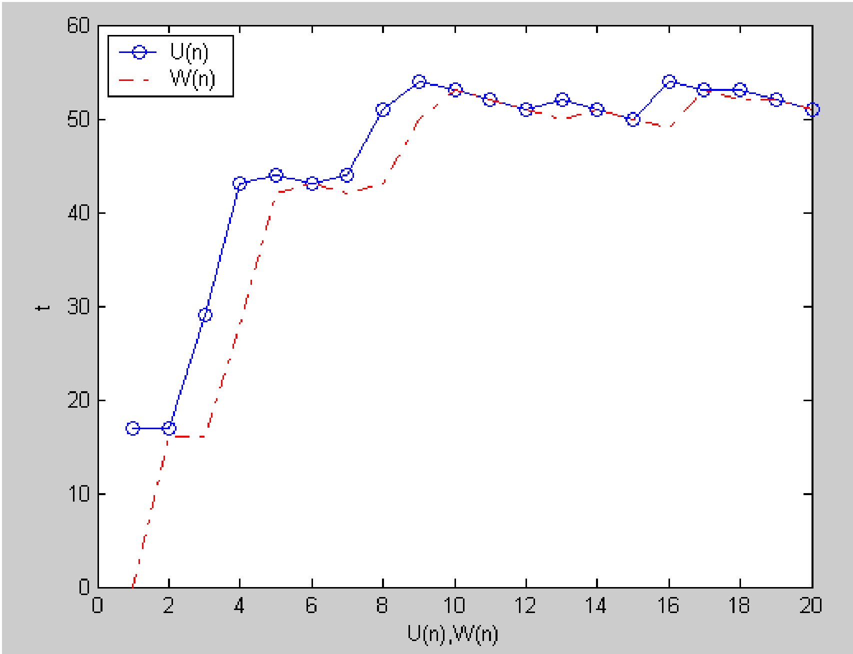
1	9
4	5
4	7
3	1
5	7
2	5
2	1
2	4
4	2
1	4



شکل 1 – 2: نتیجه اجرای برنامه برای 10 متغیر تصادفی نشان داده شده در جدول 1-1

جدول 1-2: متغیرهای تصادفی تولید شده برای x و t برای حالت n=20

متغیر های تصادفی x	متغیرهای تصادفی t
1	18
2	4
4	1
8	8
2	13
1	2
2	1
9	4
2	1
2	7



شکل 1 – 3: نتیجه اجرای برنامه برای 10 متغیر تصادفی نشان داده شده در جدول 1-2

تمرین 2

یك سیستم شبیه‌سازی ساده برای MM1 پیاده‌سازی شود و سپس برای ρ های مختلف با تولید متغیرهای پواسن ، زمان انتظار حساب شده و سپس با فرمول MM1 مقایسه شود؟

در صف M/M/1 مشتری‌ها به صورت یك توزیع پواسن با پارامتر λ به صف وارد می‌شوند و زمانی که برای هر مشتری استفاده می‌شود به صورت یك توزیع نمایی با پارامتر μ است . که ما می‌گوییم که مشتری‌ها دارای زمان سرویس نمایی هستند. اگر $\rho = \dfrac{\lambda}{\mu}$ باشد متوسط زمانی تعداد پکت‌های درون سیستم با فرمول 2-1 بیان می‌شود:

(2-1)

$$E[Q] = \frac{\rho}{1-\rho}.$$

و با استفاده از little متوسط زمان انتظار به صورت 2-2 محاسبه می‌شود:

(2-2)

$$W_t = \frac{E[Q]}{\lambda} = \frac{\rho}{(1-\rho)\lambda}$$

برای شبیه سازی از NS2 استفاده شده است . در زیر فایل tcl که برای شبیه سازی بکار می‌رود نشان داده شده است.در این برنامه یك صف نامحدود MM1 با پارامترهای $\lambda = 30, \mu = 33$ پیاده سازی شده است زمان شبیه سازی 1000 ثانیه است و در این زمان در حدود 160 میلیون پکت به صف وارد می‌شود این برنامه دو خروجی دارد یکی فایل out.tr که همه فعالیت های انجام شده در صف را نشان می‌دهد و دیگری فایل qmon.txt که هر 0.1 ثانیه یکبار وضعیت صف را نشان می‌دهد نمونه‌ای از اطلاعات فایل qmon در جدول 2-1 نشان داده شده است.

برای محاسبه احتمال حالت پایدار در ابتدا از روی تعداد پکتی که در صف سیستم قرار می‌گیرد حالت سیستم را تعیین می‌کنیم مثلاً حالت صفر به حالتی می‌گوییم که هیچ پکتی در صف نباشد و حالت یك به حالتی که یك پکت در صف موجود باشد از طریق اطلاعات موجود در ستون چهارم و پنجم جدول 2-1 می‌توان تعداد دفعاتی که سیستم در هر حالت قرار دارد را محاسبه کرد پس از این محاسبه نتایج به صورت جدول 2-2 بدست می‌آید.

برای محاسبه این روابط از یك کد پاسکال که در لیست برنامه 2 نشان داده شده است استفاده شده است.

حال از فرمول زیر تعداد پکت موجود در صف بدست می‌آید.

$$\sum_i n_i p_i$$

که در آن n شماره حالت است و p احتمال هر حالت می‌باشد. پس از محاسبه این جمع عدد 9.6908 بدست می‌آید که نزدیك به عدد 10 ای است که از فرمول 2-1 بدست می‌آید.

برای محاسبه متوسط زمان انتظار از روی جدول 2-1 به صورت زیر عمل می‌کنیم :

در ابتدا میزان زمانی که هر پکت منتظر می‌ماند را از روی ستونهای چهارم و پنجم بدست می‌آوریم پس از انجام این کار پکت‌ها دسته بندی می‌شوند ، مثلاً پکت‌هایی که هیچ تاخیری نداشته اند در دسته صفر و پکت‌هایی که 4 واحد زمانی تاخیر داشته اند نیز در دسته چهارم قرار می‌گیرند نتایج این قسمت در جدول 2-3 نشان داده شده است. همچنین یك زمان باقی‌مانده نیز به زمان‌های نشان داده شده اضافه می‌شود که این زمان در واقع تاخیر پکت هایی است که در یك واحد زمانی قرار گرفته‌اند . این تاخیرها را با هم جمع می‌کنیم و بر تعداد پکت که برابر با 160717770 است تقسیم می‌کنیم میزان تاخیر برابر 0.3365 می‌شود که با مقداری که از فرمول 2-2 بدست می‌آید(0.3333)اختلاف اندکی دارد. برای انجام این محاسبات از یك کد پاسکال که در لیست برنامه 3 نشان داده شده است استفاده می‌شود.

لیست برنامه 1

```
set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf
```

```
set lambda 30.0
set mu     33.0
set n1 [$ns node]
set n2 [$ns node]
set link [$ns simplex-link $n1 $n2 100kb 0ms DropTail]
$ns queue-limit $n1 $n2 100000

set InterArrivalTime [new RandomVariable/Exponential]
$InterArrivalTime set avg_ [expr 1/$lambda]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ [expr 100000.0/(8*$mu)]
set src [new Agent/UDP]
$ns attach-agent $n1 $src
# queue monitoring
set qmon [$ns monitor-queue $n1 $n2 [open qm.out w] 0.1]
$link queue-sample-timeout
proc finish {} {
    global ns tf
    $ns flush-trace
    close $tf
    exit 0
}
proc sendpacket {} {
    global ns src InterArrivalTime pktSize
    set time [$ns now]
    $ns at [expr $time + [$InterArrivalTime value]] "sendpacket"
    set bytes [expr round ([$pktSize value])]
    $src send $bytes
}

set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $src $sink
$ns at 0.0001 "sendpacket"
$ns at 1000.0 "finish"
$ns run
```

لیست برنامه 2

```
var
F,f1:textfile;
count:array[1..80]of integer;
prob:array[1..80]of real;


i:integer;
average:real;
s1,s2:string;
i2,i3,i6,i7,i8,i9,i10,i11:integer;
i1,i4,i5:real;
begin
assignfile(F,'c:\qm.out');
assignfile(f1,'c:\out.txt');
Reset(F);
rewrite(f1);
for i:=1 to 80 do
count[i]:=0;
for i:=1 to 10000 do
begin


readln(f,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11);
count[i6-i7+1]:=count[i6-i7+1]+1;


end;

closefile(f);
average:=0;
for i:=1 to 80 do
begin
prob[i]:=count[i]/10000;
average:=prob[i]*(i-1)+average;
```

```
listbox1.Items.Add('['+inttostr(i-1)+']= '+inttostr(count[i])+'
probability ='+floattostr(count[i]/10000));
writeln(f1,i-1,'     '+inttostr(count[i])+'    '+floattostr(count[i]/10000));
end;
closefile(f1);
  edit1.text:=floattostr(average);
end;
```

لیست برنامه 3

```
var
F:textfile;
count:array[1..80]of integer;
prob:array[1..80]of real;
qin,qout:array[1..10000]of integer;
wt:array[1..1000]of integer;

i,j:integer;
average,res:real;
s1,s2:string;
i2,i3,i6,i7,i8,i9,i10,i11:integer;
i1,i4,i5:real;
packets:integer;
begin
assignfile(F,'c:\qm.out');
Reset(F);
packets:=0;
for i:=1 to 80 do
count[i]:=0;
for i:=1 to 10000 do
begin
readln(f,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11);
qin[i]:=i6;
qout[i]:=i7;
packets:=packets+qout[i];
end;

closefile(f);
res:=0;
{average:=0;
for i:=1 to 80 do
begin
prob[i]:=count[i]/10000;
average:=prob[i]*(i-1)+average;
listbox1.Items.Add('['+inttostr(i-1)+']= '+inttostr(count[i])+'
probability ='+floattostr(count[i]/10000));
end;
 }
 for i:=1 to 1000 do
 wt[i]:=0;
 for i:=1 to 10000 do
  begin
   j:=1;
   while j<=i do
   begin
   if (qin[j]<>0) then
   begin

      if (qout[i]-qin[j]>=0) then
      begin

       qout[i]:=qout[i]-qin[j];
       wt[i-j+1]:=qin[j]+wt[i-j+1];
       if qin[j]>1 then
       res:=res+(0.1/(qin[j]-1))*(qin[j]-1)*(qin[j])/2;
       qin[j]:=0;
      end
      else
      begin

      qin[j]:=qin[j]-qout[i];
      wt[i-j+1]:=qout[i]+wt[i-j+1];
       if qout[i]>1 then
```

```
      res:=res+(0.1/(qout[i]-1))*(qout[i]-1)*(qout[i])/2;
      qout[i]:=0;
      end;

   end;
   j:=j+1;

   end;
  end;
average:=0;
for i:=1 to 10 do
begin
average:=average+(i-1)*0.1*wt[i];
listbox1.Items.Add('['+inttostr(i-1)+']= '+inttostr(wt[i])+'
time='+floattostr((i-1)*0.1*wt[i]));
end;
listbox1.Items.Add(inttostr(packets));
listbox1.Items.Add(floattostr(res));
edit1.Text:=floattostr((average+res)/packets);
end;
```

جدول 2-1: نمونه فایل qm.out

زمان	اندازه صف به بایت	اندازه صف به پکت	تعداد پکت دریافت شده	تعداد پکت خارج شده	تعداد پکت از دست رفته	تعداد بایت دریافت شده	تعداد بایت خارج شده	تعداد بایت از دست رفته
0	0	0	0	0	0	0	0	0
0.1	0	0	1	1	0	222	222	0
0.2	0	0	1	1	0	222	222	0
0.3	0	0	2	2	0	1142	1142	0
0.4	0	0	2	2	0	1142	1142	0
0.5	0	0	5	5	0	1474	1474	0
0.6	74.4306608	0.3836632	8	7	0	2736	2542	0
0.7	1098.205831	1.98488768	12	9	0	5178	3559	0
0.8	1463.581427	3.60104608	17	11	0	6501	4779	0
0.9	1561.147108	6.04729835	20	14	0	7289	5806	0

جدول 2-2: احتمال حضور در هر حالت

احتمال	تعداد رخداد	حالت
0.18	1800	0
0.072	720	1
0.0676	676	2
0.0613	613	3
0.0537	537	4
0.0523	523	5
0.0446	446	6
0.0439	439	7
0.0403	403	8
0.0357	357	9
0.0306	306	10
0.0303	303	11
0.0272	272	12
0.0258	258	13
0.0195	195	14

0.0189	189	15
0.0165	165	16
0.017	170	17
0.0136	136	18
0.0098	98	19
0.0125	125	20
0.0084	84	21
0.0088	88	22
0.007	70	23
0.0062	62	24
0.0054	54	25
0.0066	66	26
0.0052	52	27
0.006	60	28
0.0057	57	29
0.0033	33	30
0.0026	26	31
0.0038	38	32
0.0024	24	33
0.0024	24	34
0.003	30	35
0.003	30	36
0.0022	22	37
0.0031	31	38
0.0027	27	39
0.0022	22	40
0.0027	27	41
0.0023	23	42
0.0025	25	43
0.002	20	44
0.0019	19	45
0.0013	13	46
0.0013	13	47
0.001	10	48
0.0009	9	49
0.0013	13	50
0.0006	6	51
0.0005	5	52
0.0007	7	53
0.001	10	54
0.0017	17	55
0.0012	12	56
0.0014	14	57
0.001	10	58
0.0006	6	59
0.0008	8	60
0.0007	7	61
0.001	10	62
0.0004	4	63
0.0006	6	64

0.0011	11	65
0.001	10	66
0.0003	3	67
0.0006	6	68
0.0007	7	69
0.0011	11	70
0.0009	9	71
0.0004	4	72
0.0004	4	73
0.0001	1	74
0.0003	3	75
0.0004	4	76
0.0001	1	77

جدول 2-3

زمان	تعداد پکت در دسته	دسته
0	4046	0
36105.1	361051	1
5822000	29110000	2
36920006.4	123066688	3
3270394	8175985	4
8035887		residual

تمرین 3

تابع توزیع زمان انتظار و تعداد مشتری در سیستم M/G/1 را بدست آورید؟با تابع توزیع تعداد مشتری میانگین زمان انتظار را بدست آورید؟
در این صف ورود مشتری توزیع پواسن با پارامتر λ است در حالیکه زمان سرویس مشتری‌ها به طور مستقل با يك توزیع دلخواه G(x) می‌باشد.اگر σ_i,σ_j زمان سرویس دو مشتری باشد و $i \neq j$ باشد آنگاه

1- σ_i,σ_j مستقل هستند

2- $G(x)=P(\sigma_i \le x)=P(\sigma_j \le x)$ برای همه $x \ge 0$

$\frac{1}{\mu}$ را زمان متوسط زمان سرویس قرار می‌دهیم که $E[\sigma_i]=\frac{1}{\mu}$ است و همچنین $\rho=\frac{\lambda}{\mu}$ است و نیز مشتری‌ها براساس FIFO سرویس داده می‌شوند.
برای این سیستم صف فرایند $(N(t),t\ge 0)$ که در آن N(t) تعداد مشتری در صف در زمان t است ، يك فرایند مارکف نیست و این بخاطر اینست که اگر فقط ما N(s) را بدانیم نمی‌توانیم آینده احتمالی N(t) برای t>s را مشخص کنیم.بجز حالتی که N(s)=0 است .

متوسط طول صف و متوسط زمان پاسخ

W_n را زمان انتظار در صف برای n امین مشتری تعریف می‌کنیم.همچنین پارامترهای زیر را تعریف می‌کنیم:

- $\overline{W}$ ، متوسط زمان انتظار
- X(t)، تعداد مشتری در اتاق انتظار در زمان t
- R(t)، زمان سرویس باقی‌مانده مشتری در حال سرویس
- t_n ، زمان رسیدن n امین مشتری
- σ_n، زمان سرویس مشتری n

قرار داد می‌کنیم که $X(t_i)$ تعداد مشتری در اتاق انتظار درست قبل از رسیدن iامین مشتری است.
عبارات 3-1 را بدست می‌آوریم:
(3-1)

$$\begin{aligned} E[W_i] &= E[R(t_i)] + E\left[\sum_{j=i-X(t_i)}^{i-1} \sigma_j\right] \\ &= E[R(t_i)] + \sum_{k=0}^{\infty}\sum_{j=i-k}^{i-1} E[\sigma_j \mid X(t_i)=k]\, P(X(t_i)=k) \\ &= E[R(t_i)] + \frac{1}{\mu} E[X(t_i)]. \end{aligned}$$

برای بدست آوردن 3-1 ما از این واقعیت استفاده کرده‌ایم که σ_j مستقل از $X(t_i)$ برای $j=i-X(t_i),\ldots,i-1,$ است که بیان می‌کند $E[\sigma_j \mid X(t_i)=k]=1/\mu$.
البته $X(t_i)$ فقط به زمان سرویس σ_j برای $j=1,\ldots,i-X(t_i)-1$ وابسته است و به σ_j برای $j \ge i-X(t_i)$ وابسته نیست.
اگر $i \to \infty$ از 3-1 داریم :

(3-2)

$$\overline{W} = \overline{R} + \frac{\overline{X}}{\mu}$$

با

- $\overline{R} := \lim_{i \to \infty} E[R(t_i)]$ متوسط زمان سرویس در تناوبهای واردشدن در حالت پایدار است
- $\overline{X} := \lim_{i \to \infty} E[X(t_i)]$ متوسط تعداد مشتری در اتاق انتظار در تناوبهای واردشدن در حالت پایدار است

بخاطر اینکه فرایند واردشدن یك فرایند پواسن است (خاصیت PASTA) ما داریم که

(3-3)

$$\overline{R} \;=\; \lim_{t \to \infty} \frac{1}{t} \int_0^t R(s)\, ds$$

(3-4)

$$\overline{X} \;=\; \lim_{t \to \infty} \frac{1}{t} \int_0^t X(s)\, ds.$$

عبارت 3-3 بیان می‌کند که متوسط باقی‌مانده از زمان سرویس در تناوبهای واردشدن و همچنین در تناوبهای دلخواه به یك صورت هستند. به طور مشابه 3-4 نیز بیان می‌کند که متوسط تعداد مشتری در تناوبهای وارد شدن و همچنین در تناوبهای دلخواه همانند هم هستند.

تمرین 4

- ✓ **ALOHA:** در این روش هر وسیله داده خود را بدون توجه به وضعیت سایر اعضای شبکه بر روی شبکه ارسال می‌کند، لکن می‌بایست به طریقی از دریافت داده اطمینان حاصل کند. بدین منظور منتظر دریافت تصدیق از گیرنده می‌ماند. اگر مدت زمان انتشار بسته داده تا مقصد را بطور متوسط t_{prop} بدانیم، فرستنده $2t_{prop}$ منتظر می‌ماند چنانچه تصدیق دریافت نشود فرستنده فرض می‌کند که بسته داده به مقصد نرسیده است. این عدم دریافت می‌تواند به دو دلیل رخ بدهد (1) ممکن است داده ارسالی بنابه هر دلیل بدرستی به گیرنده نرسیده باشد و دچار خطا شده باشد. مثلا امواج رادیوی دیگر در شبکه‌های بی‌سیم یا میدان‌های الکتریکی داده‌های بسته را تغییر داده باشند و بسته از دست رفته باشد. (2) ممکن است همزمان با ارسال بسته عضو دیگری در شبکه اقدام به ارسال کرده باشد که در این صورت تصادم رخ داده است. در هر دو حالت فرستنده دوباره اقدام به ارسال می‌کند. اما اقدام به ارسال دوباره می‌بایست بر اساس الگوریتم مشخصی صورت گیرد و هر سیستم نمی‌بایست به صورت خودسر اقدام به ارسال دوباره کند. مثلا فرض کنید دو سیستم همزمان اقدام به ارسال کرده‌اند در نتیجه بسته‌های ارسالی تصادم پیدا می‌کنند حال اگر هر دو سیستم همزمان از این موضوع مطلع شوند و بخواهند دوباره اقدام به ارسال کنند در نتیجه این سناریو دوباره رخ می‌دهد و تقریبا سیستم ارسال به بن‌بست منتهی می‌شود. لذا می‌بایست برای ارسال مجدد الگوریتمی بدست آورد. برای ارسال مجدد دو الگوریتم وجود دارد:

1. هر گاه سیستم از تصادم در شبکه مطلع شد بصورت تصادفی زمانی را برای ارسال مجدد اختیار می‌کند و پس از سپری شدن این زمان سیستم اقدام به ارسال مجدد می‌نماید. به این روش Pure ALOHA می‌گویند. برای تحلیل این شبکه فرض کنید بسته‌های با طول ثابت L و زمان انتقال X=L/R بر روی شبکه منتشر می‌شوند. حال در نظر بگیرید سیستمی در زمان t_0 اقدام به ارسال بسته بر روی شبکه می‌نماید، بسته در زمان t_0+X به مقصد خواهد رسید، لکن در این زمان نمی‌بایست ارسال دیگر صورت گیرد. به طور کلی برای اینکه این بسته در زمان X به مقصد برسد در بازه زمانی $[t_0-X , t_0+X]$ هیچ ارسالی نباید صورت بگیرد، چرا که در غیر این صورت شاهد تصادم بسته‌ها خواهیم بود. برای محاسبه احتمال عدم رخداد تصادم اینگونه تحلیل می‌کنیم. فرض کنید S نرخ ورود بسته‌های جدید به ازای زمان X به شبکه می‌باشد به عبارت دیگر در هر بازه زمانی X ثانیه S بسته جدید در شبکه ارسال می‌شود. در نتیجه S توان عملیاتی(Throughput) شبکه خواهد بود. اما پرواضح است که بسته‌های موجود در شبکه دو دسته هستند یکی بسته‌های ارسالی جدید و دیگری بسته‌های که با تصادم مواجه شده‌اند و مجددا ارسال شده‌اند. پس فرض کنید مجموع تمامی بسته‌های ورودی به شبکه شامل بسته‌های جدید و بسته‌های ارسالی مجدد G واحد در X ثانیه باشند. برای تحلیل راحتتر Abramson فرض کرده است این بسته‌ها با توزیع پواسون و متوسط 2G بسته در هر 2X ثانیه وارد شبکه می‌شوند. در نتیجه خواهیم داشت.

$$P[k\,transmitions\,in\,2X\,\sec onds] = \frac{(2G)^k}{k!}e^{-2G} \quad ,k = 0,1,2,...$$

توان عملیاتی S برابر خواهد بود با:

$$\begin{aligned} S &= GP[no\,collision] = GP[0\,transmisions\,in\,2X\,\sec onds] \\ &= G\frac{(2G)^0}{0!}e^{-2G} \\ &= Ge^{-2G} \end{aligned}$$

متوسط تاخیر در این شبکه را می‌توان این‌گونه محاسبه کرد. متوسط تعداد ارسال‌های مجدد هر بسته برابر خواهد بود با: $G/S = e^{2G}\ attempts\,per\ packet$ در نتیجه تعداد اقدام‌های ناموفق برابر با $\varepsilon = G/S - 1 = e^{2G} - 1$ خواهد بود. انتقال اول به زمان $X+t_{prop}$ نیاز دارد و انتقال بعدی به زمان $2t_{prop}+X+B$ نیاز دارند.B نیز متوسط زمان برگشتی(Backoff) می‌باشد. پس متوسط زمان انتقال برابر خواهد بود با:

$E[T_{aloha}] = X + t_{prop} + (e^{2G} - 1)(X + 2t_{prop} + B)$ می‌توان این رابطه را بر اساس ضریبی از

X بصورت $E[T_{aloha}]/X = 1 + a + (e^{2G} - 1)(1 + 2a + B/X)$ بازنویسی کرد.A متوسط زمان ارسالی یکطرفه است که بصورت نرمال شده می‌باشد$a=t_{prop}/X$.

2. می‌توان کارایی الگوریتم قبل را با کاهش احتمال برخورد افزایش داد. بدین منظور همه اعضا شبکه می‌بایست به صورت هماهنگ اقدام به ارسال کنند. هر سیستم می‌بایست بازه‌های زمانی ارسالی را پیگیری کند و تنها در ابتدای هر بازه زمانی اقدام به ارسال کند. به این روش Slotted ALOHA می‌گویند. درنتیجه اگر سیستمی بسته‌ای را در زمان t_0 ارسال کرده باشد برای آنکه تصادمی رخ دهد می‌بایست سیستم دیگر در بازه $[t_0-X\ ,\ t_0]$ اقدام به ارسال کرده باشد. در نتیجه محاسبات پیشین بصورت زیر تغییر خواهد کرد:

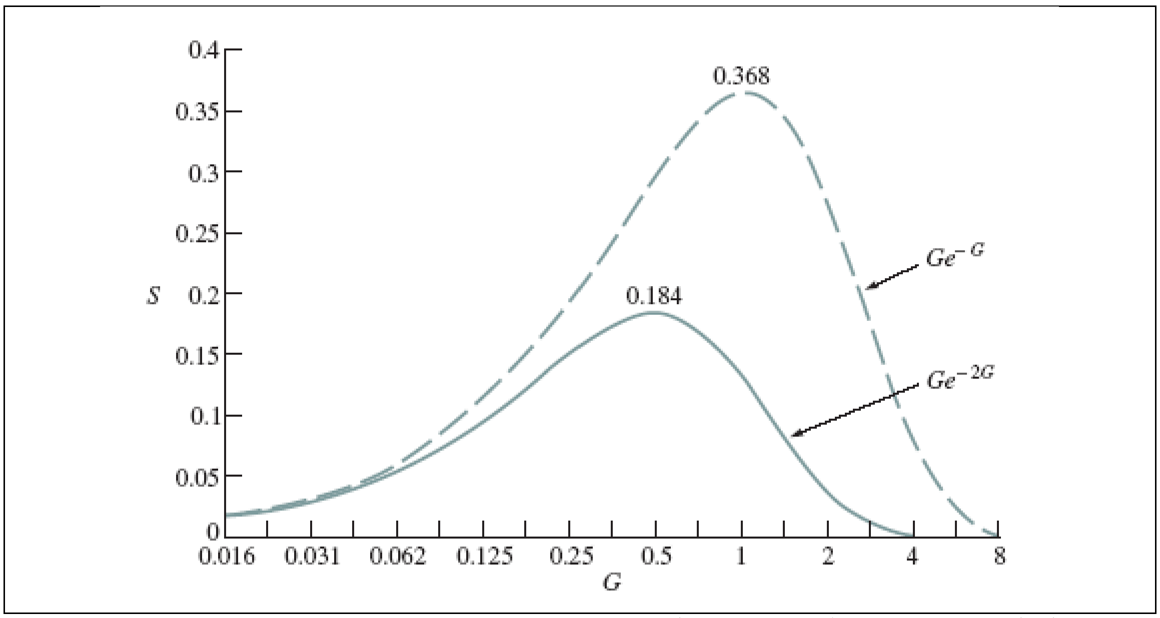
$$S = GP[no\,collision] = GP[0\,trnasmisions\,in\,X\,\sec onds]$$

$$= G\frac{(G)^0}{0!}e^{-G}$$

$$= Ge^{-G}$$

$$E[T_{slottedaloha}]/X = 1 + a + (e^{G} - 1)(1 + 2a + B/X)$$

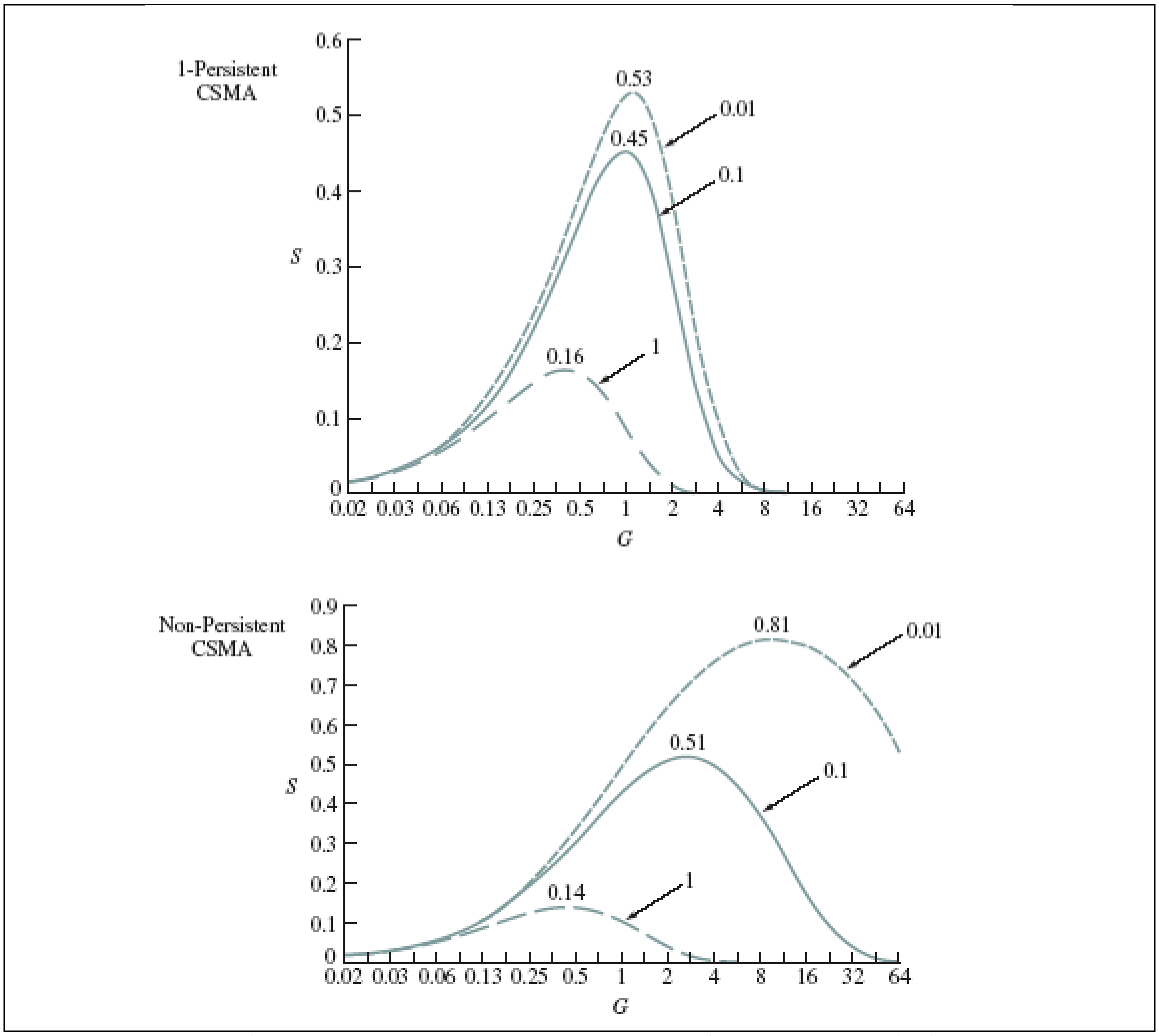
متوسط اقدام به ارسال نیز $G/S = e^{G}$ می‌باشد. در نمودار زیر مقایسه‌ای بین این دو روش ارائه شده است.



شکل 4 – 1: توان عملیاتی S در مقابل بار شبکه G برای Pure ALOHA و Slotted ALOHA

- ✓ **CSMA:** در روش قبل برای جلوگیری از تصادم هیچ راهکاری درنظر گرفته نشده است. اما در این روش سعس می‌شود از رخداد تصادم جلوگیری شود. بدین منظور هر سیستم فرستنده قبل از ارسال سعی می‌کند ابتدا از آزاد بودن شبکه اطمینان حاصل کند چنانچه شبکه را آزاد یافت اقدام به ارسال می‌کند. اگر شبکه ازاد نبود سیستم تا زمان آزاد شدن شبکه منتظر می‌ماند و هر زمان که شبکه را آزاد یافت اقدام به ارسال می‌نماید. هر گاه سیستم از تصادم در شبکه مطلع شود برای ارسال مجدد اقدام می‌نماید. مثلا فرض کنیذ سیستمی اقدام به ارسال بسته می‌کند. باقی سیستم‌ها از این ارسال آگاه می‌شوند در نتیجه تا پایان زمان انتقال باقی سیستم‌ها از ارسال بسته خودداری خواهند کرد. حال در پایان زمان سیستمی که منتظر آزاد شدن شبکه بوده است شبکه را ازاد می‌یابد و اقدام به ارسال می‌کند. اما اگر بیش از یك سیستم منتظر باشند آنگاه تصادم رخ خواهد داد. در این روش بسته به متدی که سیستم برای بررسی آزاد بودن خط به کار

می‌برد سه دسته شبکه وجود دارد. دسته اول 1-Persistentها می‌باشند. این دسته چنانچه خط را مشغول ببیند مرتب خط را بررسی می‌کنند به محض اینکه خط آزاد شد بسته خود را ارسال می‌کنند. این شبکه‌ها برای ترافیك‌های بالا اصلا مناسب نمی‌باشند. دسته دوم Non-Persistentها می‌باشند. این دسته چنانچه خط را مشغول ببیند مدت زمانی را بر اساس الگوریتم منتظر می‌مانند و سپس خط را بررسی می‌کنند. بدین ترتیب از وقوع تصادم پیشگیری می‌کنند. چنانچه این زمان خیلی کوتاه شود کارایی به مانند حالت قبل می‌شود. اما دسته سوم ترکیبی از دو دسته قبل می‌باشذ. این دسته p-Persistentها می‌باشند. این دسته هرگاه خط مشغول باشد زمانی را منتظر می‌مانند و آنگاه با احتمال p ارسال می‌کنند. محاسبه توان عملیاتی این شبکه‌ها پیچیده است. در اینجا نمودار زیر برگفته شده از کتاب Communication Networks نوشته Leon-Garcia Albert آورده شده است.

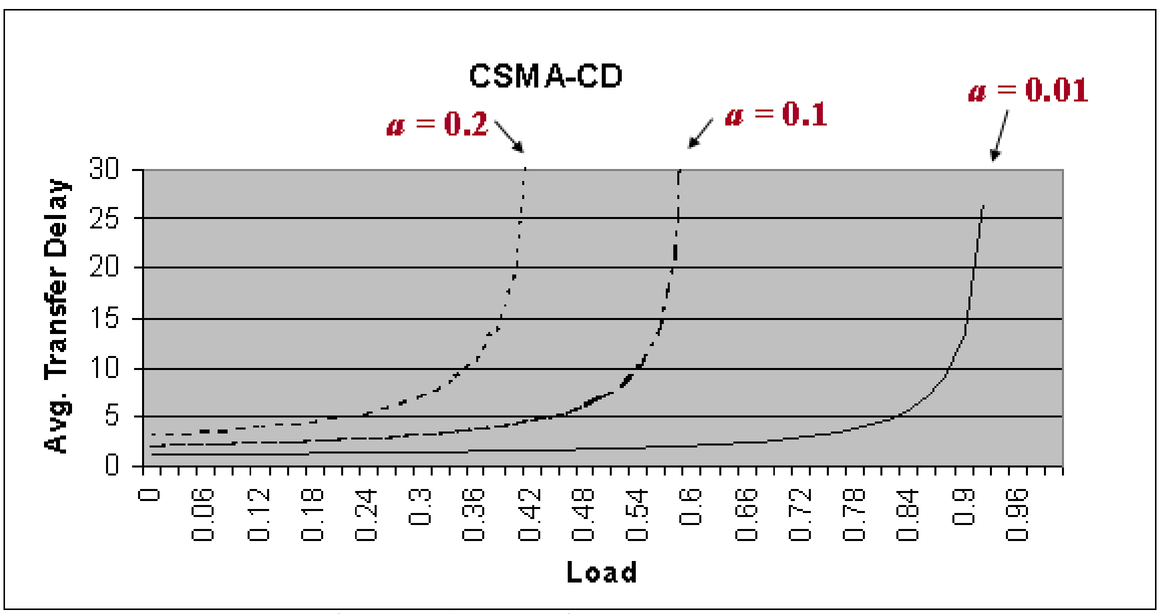


شکل 4 – 2: توان عملیاتی S براساس بار شبکه G برای aهای مختلف

- ✓ **CSMA/CD:** در روش‌های قبل تصادم به ازای هر بسته ارسالی امکان‌پذیر بود. چنانچه بتوان از رخداد تصادم به ازای برخی بسته‌ها جلوگیری کرد می‌توان کارایی شبکه را بالا برد. بدین منظور کافی است تصادم در شبکه بررسی شود و هر گاه تصادمی تشخیص داده شد از ارسال باقی بسته‌ها جلوگیری کرد. این روشی است که در الگوریتم CSMA/CD استفاده می‌شود. هرگاه تصادمی در شبکه تشخیص داده شود يك سيگنال پارازيت در شبکه اراسل می‌شود تا باقی سیستم‌ها نیز از تصادم باخبر شوند و از ارسال بسته خودداری کنند. هرگاه سیستم خط را مشغول دید می‌تواند به کمك یکی از روش‌های CSMA برای بررسی خط استفاده کند. برای تحلیل در حالت 1-Persistent می‌توان فرض کرد که زمان به ریز بازهایی

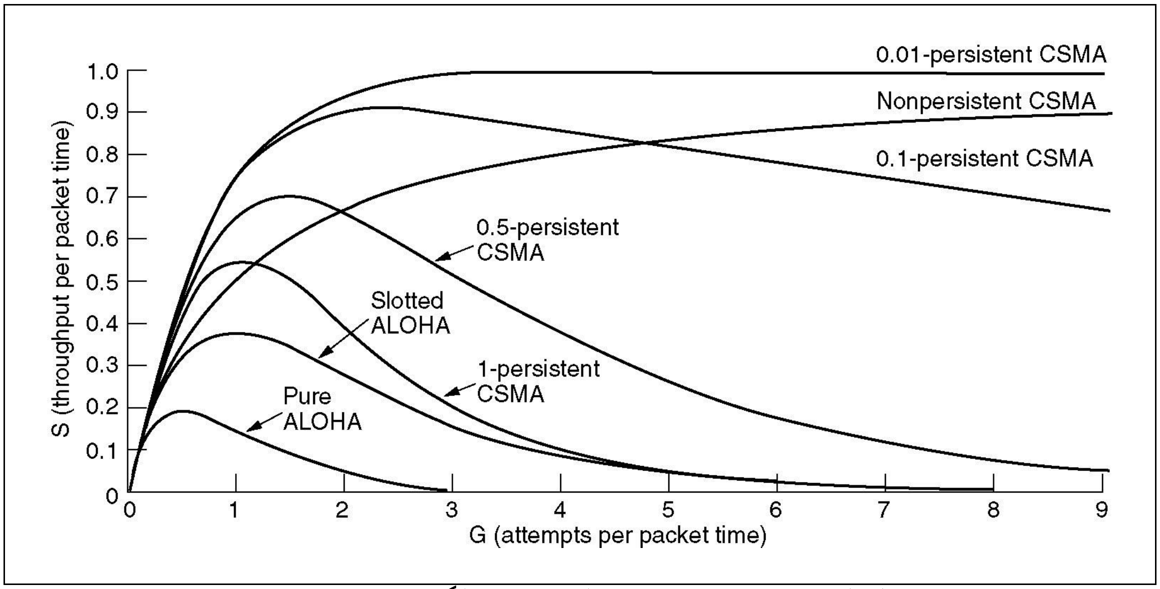
با طول $2t_{prop}$ تقسیم شده‌اند. هر گاه خط آزاد شده سیستم‌ها برای بدست آوردن خط مجادله می‌کنند، بدین ترتیب که بسته خود را ارسال می‌کنند و خط را برای اطمینان از دریافت بسته بررسی می‌کنند. هر مجادله برای بدست آوردن خط $2t_{prop}$ طول می‌کشد. حال متوسط زمان ارسال موفق را بدست می‌آوریم. فرض کنید n دستگاه برای بدست آوردن خط در حال مجادله هستند و هر دستگاه با احتمال p بسته خود را ارسال می‌کند. احتمال ارسال موفق برابر با احتمال آن است که تنها يك دستگاه اقدام به ارسال كند يعنی: $P_{success}=np(1-p)^{n-1}$. به راحتی دیده می‌شود به ازای p=1/n بیشترین احتمال ارسال موفق بدست می‌آید: $P^{\max}_{success}=n\frac{1}{n}(1-\frac{1}{n})^{n-1}=(1-\frac{1}{n})^{n-1}$. متوسط تعداد ریزبازهایی که نیاز تا خط در اختیار گرفته شود را می‌توان اینگونه محاسبه کرد، ابتدا احتمال آنکه j ریزبازه مورد نیاز باشدبرابر خواهد بود با: $P[j \min islot\ in\ contention \int erval]=(1-P^{\max}_{success})^{j-1}P^{\max}_{success}\quad j=1,2,..$ پس خواهیم داشت: $E[J]=\sum_{j=1}^{\infty} j(1-P^{\max}_{success})^{j-1}P^{\max}_{success}=\frac{1}{P^{\max}_{success}}$. به ازای n های خیلی بزرگ خواهیم داشت E[J]=e=2.718. بیشترین توان عملیاتی برای CSMA/CD زمانی بدست می‌آید که تمامی زمان خط صرف انتقال شود و به دنبال آن مجادله برای تصاحب روی دهد. زمان انتقال هر بسته E[X] است که همراه يك زمان انتقال كامل t_{prop} و يك بازه مجادله $2et_{prop}$ می‌باشد. پس حداكثر توان عملياتی برابر خواهد بود با: $\rho_{\max}=\frac{E[X]}{E[X]+t_{prop}+2et_{prop}}=\frac{1}{1+(2e+1)a}$. متوسط زمان تاخیر در CSMA/CD را می‌توان با محاسبات پیچیده‌ای بدست آورد که در اینجا نتیجه نهایی از Schwartz 1987 نقل می‌شود.

$$\frac{E[T]}{X}=\rho\frac{1+(4e+2)a+5a^2+4e(2e-1)a^2}{2\{1-\rho(1+(2e+1)a)\}}+1+2ea+\frac{(1-e^{-2a\rho})(\frac{2}{\rho}+2ae^{-1}-6a)}{2(e^{-\rho}e^{-\rho a-1}-1+e^{-2\rho a})}+\frac{a}{2}$$

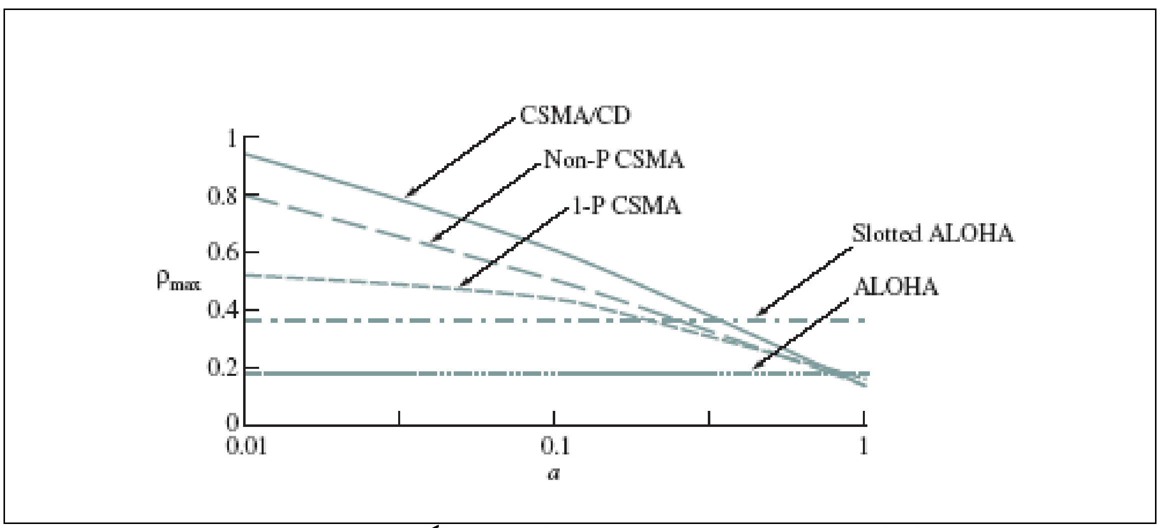


شکل 4 – 3: مقایسه متوسط زمان انتقال در شرایط مختلف بارشبکه

در ادامه نموداری که توان عملیاتی الگوریتم‌های قبلی را مقایسه کرده است از Leon-Garcia 2000 آورده شده است



شکل 4 – 4: مقایسه توان عملیاتی انواع الگوریتم‌ها دسترسی



شکل 4 – 5: مقایسه حداکثر توان عملیاتی در الگوریتم‌های ذکر شده

- ✓ **Token-Ring:** در این روش همواره در شبکه بسته‌ای وجود دارد هر سیستمی که قصد ارسال بسته‌ای را داشته باشد باید منتظر بماند تا یك بسته آزادا دریافت کند، آنگاه بسته آزاد را با یك بسته داده که به آن بسته مشغول گفته می‌شود جایگزین کرده و وارد شبکه می‌کند. بعبارت دیگر در این شبکه هر سیستم یك وردی و یك خروجی دارد که بسته ورودی را با یك تاخیر در خروجی ارسال می‌کند. هرگاه سیستمی بسته‌ای را دریافت کرد آن ار بررسی می‌کند اگر بسته متعلق به خودش بود از آن یك کپی برمی‌دارد حال باید بسته‌ای را وارد شبکه کند برای این منظور دو انتخاب دارد (1) بسته آزادی را وارد شبکه کند تا

فرستنده بعدی بتواند بسته خود را ارسال کند. (2) بسته را را دوباره در شبکه ارسال کند تا فرستنده با دریافت آن از رسیدن بسته به گیرنده اطمینان حاصل کند. معمولا روش دوم مورد استفاده قرار می‌گیرد. حال بسته‌ای که در شبکه قرار گرفته است باید برداشته شود و جایگزین گردد. جایگزینی بسته سه روش عمده و معروف دارد.

1. Multi token: در این روش بسته آزاد بلافاصله پس از انتقال آخرین بیت بسته، وارد شبکه می‌شود. این روش امکان ارسال همزمان چندین بسته را در شبکه فراهم می‌آورد.

2. Single token: در این روش بسته آزاد زمانی وارد شبکه می‌شود که آخرین بیت بسته مشغولی ارسالی دریافت شود.

3. Sigle packet: بسته آزاد تنها زمانی وارد شبکه می‌شود که آخرین بیت بسته مشغولی به درستی دریافت شود. در این روش به ایستگاه فرستنده این امکان داده می‌شود قبل از اتمام فرایند کنترل بسته ارسالی از خطاها آگاه شود.

فرض کنید حداکثر يك بسته می‌تواند ارسال شود. اگر τ' لختی حلقه (تعداد بیت‌هایی که همزمان می‌توانند در شبکه باشند) در ثانیه باشد و a' نیز لختی نرمال شده حلقه به ازای زمان انتقال باشد خواهیم داشت:

$$\tau' = \tau + \frac{Mb}{R} \qquad\qquad a' = \frac{\tau'}{E[X]}$$

τزمان انتشار در طول حلقه است، b تاخیر بیت در يك سیستم می‌باشد، Mb نیز تاخیر بیت در کل M سیستم است و R سرعت انتقال خط می‌باشد. حداکثر توان عملیاتی زمانی رخ می‌دهد که تمامی سیستم‌ها بسته‌ای را انتقال دهند. اگر سیستم به روش Multi token عمل کند زمانی که طول می‌کشد که بسته از تمامی M ایستگاه عبور کند $ME[X]+\tau'$ خواهد بود در نتیجه حداکثر توان عملیاتی برابر است با :

$$\rho_{\max} = \frac{ME[X]}{ME[X]+\tau'} = \frac{1}{1+\tau'/ME[X]} = \frac{1}{1+a'/M} \quad for\, multitoken$$

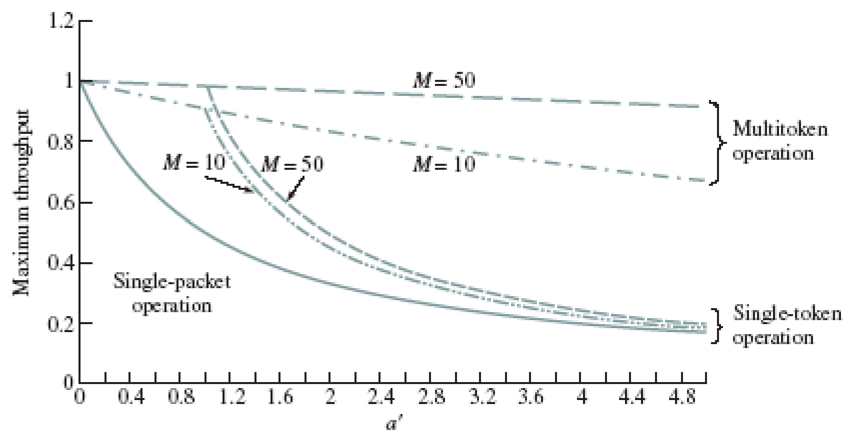
حال فرض کنید از روش Singletoken استفاده شده است. طول هر بسته ثابت L باشد و زمان انتقال نیز X=L/R است. زمان موثر انتقال بسته ماکزیمم بین X و M خواهد بود و حداکثر توان عملیاتی برابر است با:

$$\rho_{\max} = \frac{MX}{M\max\{X,\tau'\}+\tau'} = \frac{1}{\max\{1,a'\}+\tau'/MX} = \frac{1}{\max\{1,a'\}+a'/M} \quad for\, Singletoken$$

و بالاخره اگر از روش Singlepacket استفاده شود زمان موثر انتقال بسته $E[X]+\tau'$ خواهد بود و حداکثر توان عملیاتی برابر است با :

$$\rho_{\max} = \frac{ME[X]}{M(E[X]+\tau')+\tau} = \frac{1}{1+a'(1+\frac{1}{M})} \quad for\, \sin glepacket$$

در زیر نموداری به نقل از Leon Garcia 2000 برای مقایسه توان عملیاتی سه حالت مختلف ارائه شده است.



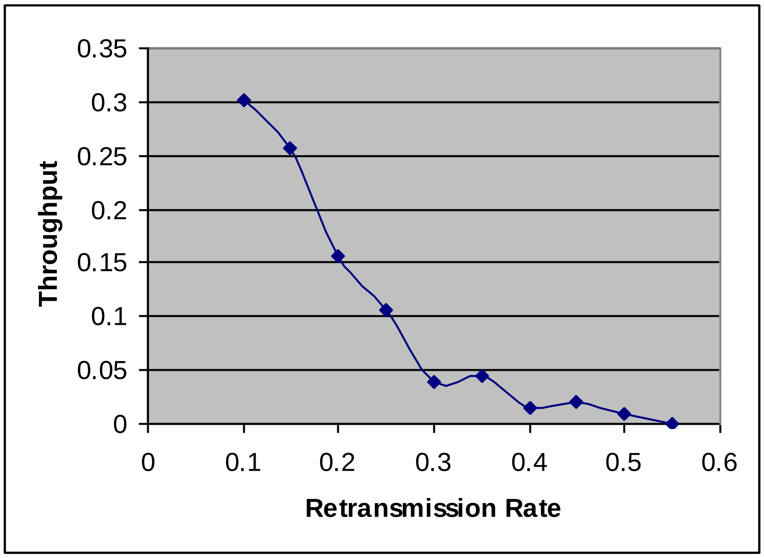
شکل 4 – 6: مقایسه توان عملیاتی

- ✓ **نتایج شبیه‌سازی:** شبیه‌سازی به کمک Applet موجود در http://egnatia.ee.auth.gr/~dviz/cn2/simulation/applet/aloha.htm انجام شده است.

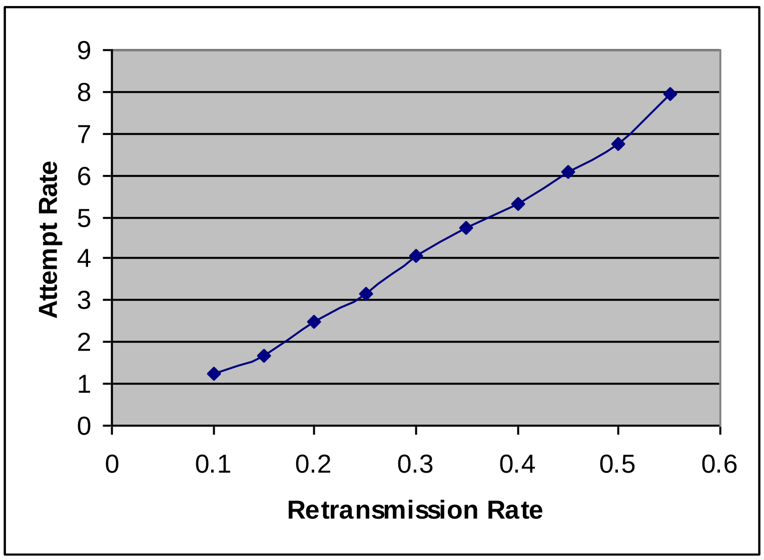
جدول 4 – 1: نتایج شبیه سازی Slotted ALOHA برای اجرای اول

Slots	200	Slot Duration (ms)	500
Hosts	15	Arrival Rate	0.1
Persistency	1.0	Retransmission Policy	Geometric

Retransmission Rate	Throughput	Attempt Rate	Mean Packet Delay
0.1	0.302	1.251	31.048
0.15	0.256	1.693	34.619
0.2	0.156	2.472	53.993
0.25	0.106	3.176	70.6
0.3	0.04	4.06	126.816
0.35	0.045	4.744	113.118
0.4	0.015	5.291	155.422
0.45	0.02	6.085	143.096
0.5	0.01	6.729	161.406
0.55	0	7.955	192.22



الف



ب

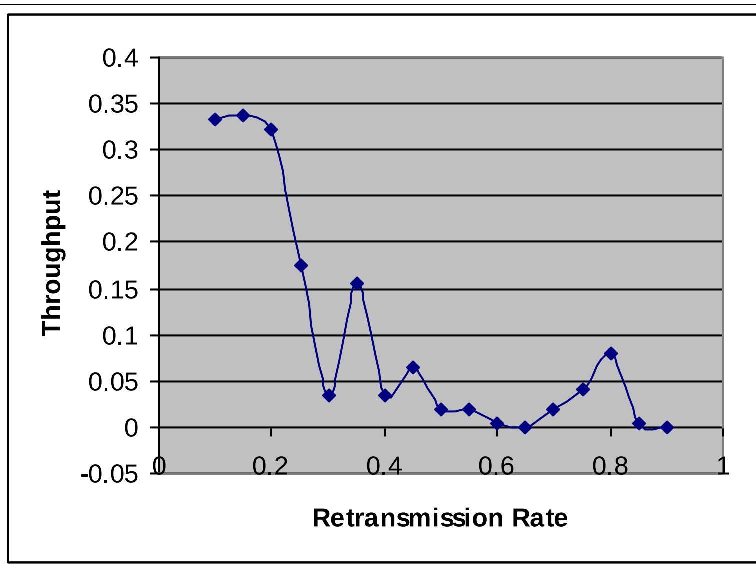
شکل 4 – 7: الف) توان عملیاتی و ب) نرخ تلاش دوباره بر حسب نرخ ارسال دوباره

جدول 4 – 2: نتایج شبیه سازی Slotted ALOHA برای اجرای دوم

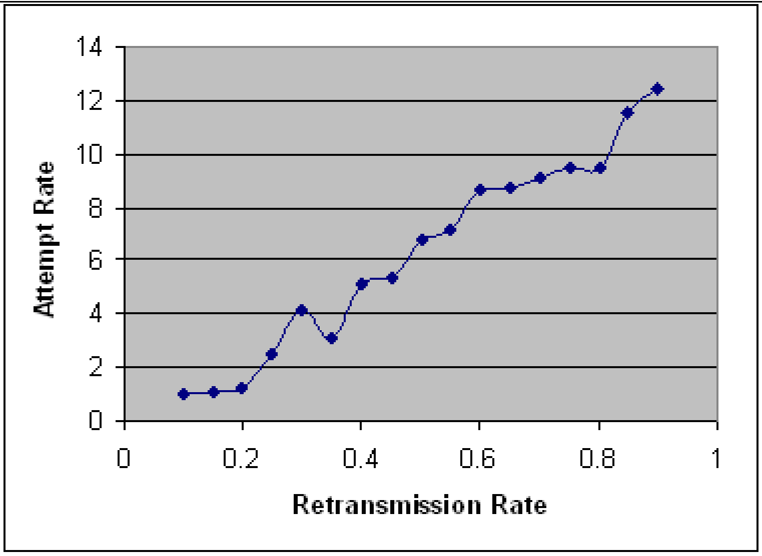
Slots	200	Slot Duration (ms)	500

Hosts	15	Arrival Rate	0.05
Persistency	1.0	Retransmission Policy	Geometric

Retransmission Rate	Throughput	Attempt Rate	Mean Packet Delay
0.1	0.332	1.055	21.798
0.15	0.337	1.101	17.883
0.2	0.322	1.266	18.548
0.25	0.176	2.523	41.567
0.3	0.035	4.181	116.24
0.35	0.156	3.151	40.56
0.4	0.035	5.171	123.779
0.45	0.065	5.372	85.148
0.5	0.02	6.774	138.781
0.55	0.02	7.186	136.095
0.6	0.005	8.623	175.646
0.65	0	8.749	178.453
0.7	0.02	9.09	137.31
0.75	0.04	9.462	106.644
0.8	0.08	9.472	76.995
0.85	0.005	11.558	167.838
0.9	0	12.472	182.65



الف



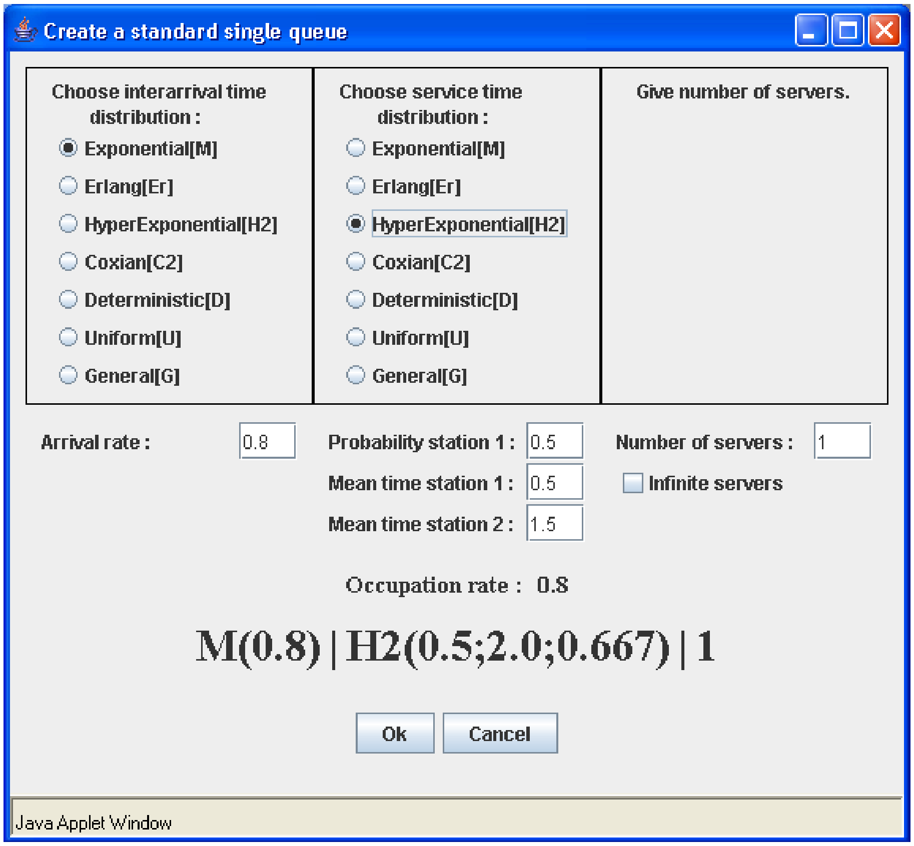
ب

شکل 7 – 5: الف) توان عملیاتی و ب) نرخ تلاش

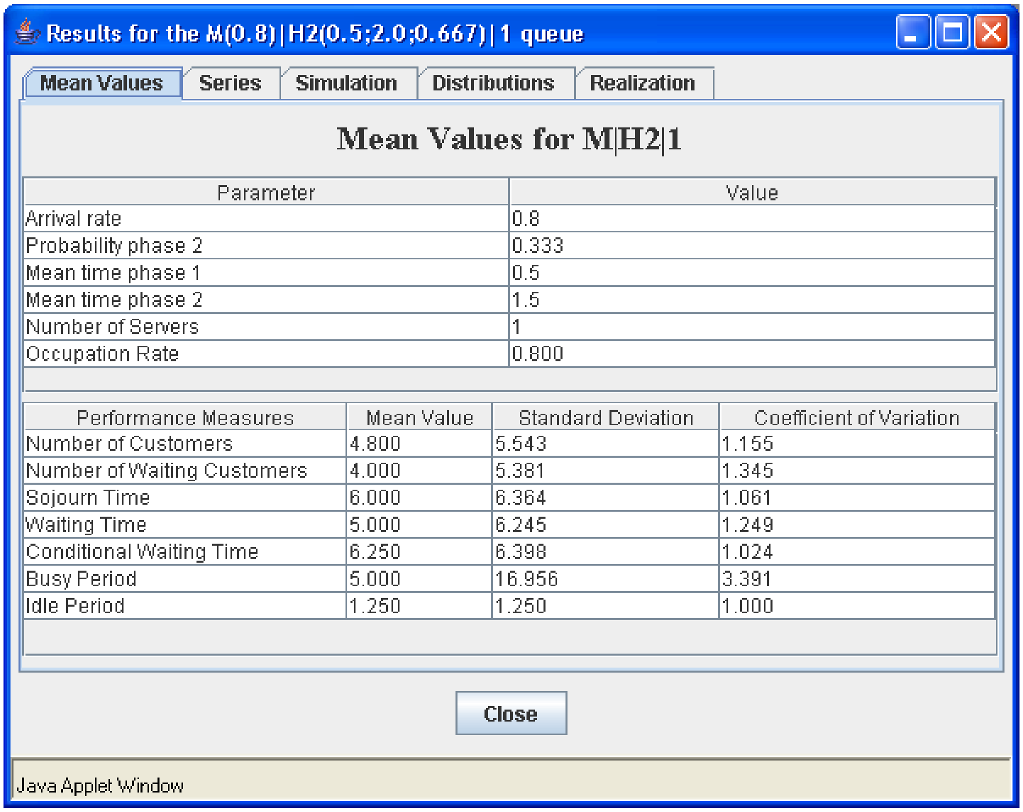
تمرین 5

برای حل کردن قسمت های مربوط به صف M/H2/1 و M/Cox2/1 ، از یک اپلت جاوا به نام Queue 2.0 که توسط Herman Verschuren و Hans de Swart نوشته شده است و در آدرس http://www.win.tue.nl/cow/Q2/ موجود می باشد استفاده شده است. در ادامه با استفاده از این برنامه، بطور گرافیکی، مراحل محاسبه مقادیر خواسته شده را تشریح می کنیم.

5 – 1 صف M/H2/1



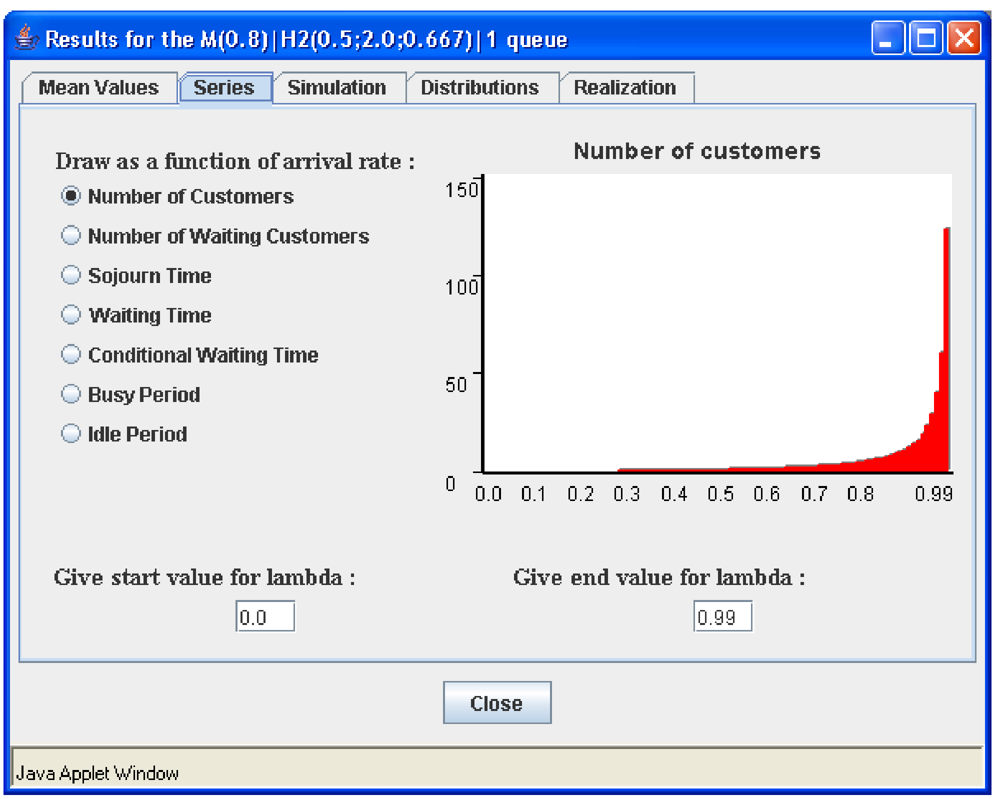
شکل 5 – 1: ابتدا یک صف M/H2/1 با مقادیر مشخص برای پارامترهای آن می سازیم. مقادیر در شکل مشخص شده اند.



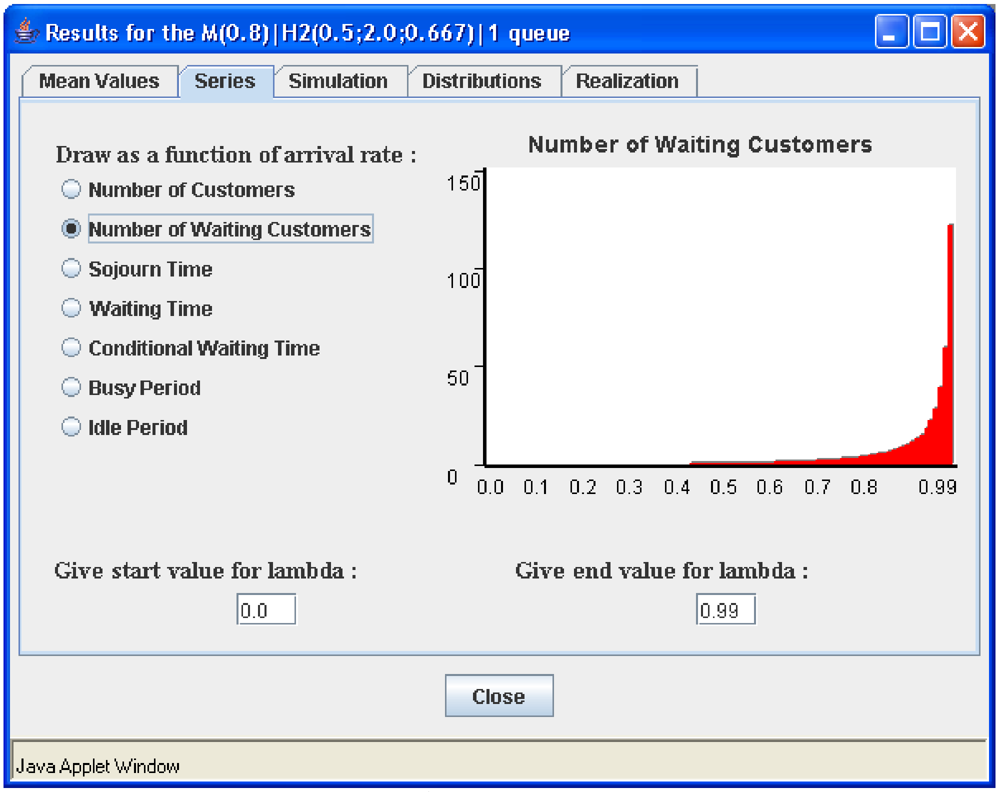
Parameter	Value
Arrival rate	0.8
Probability phase 2	0.333
Mean time phase 1	0.5
Mean time phase 2	1.5
Number of Servers	1
Occupation Rate	0.800

Performance Measures	Mean Value	Standard Deviation	Coefficient of Variation
Number of Customers	4.800	5.543	1.155
Number of Waiting Customers	4.000	5.381	1.345
Sojourn Time	6.000	6.364	1.061
Waiting Time	5.000	6.245	1.249
Conditional Waiting Time	6.250	6.398	1.024
Busy Period	5.000	16.956	3.391
Idle Period	1.250	1.250	1.000

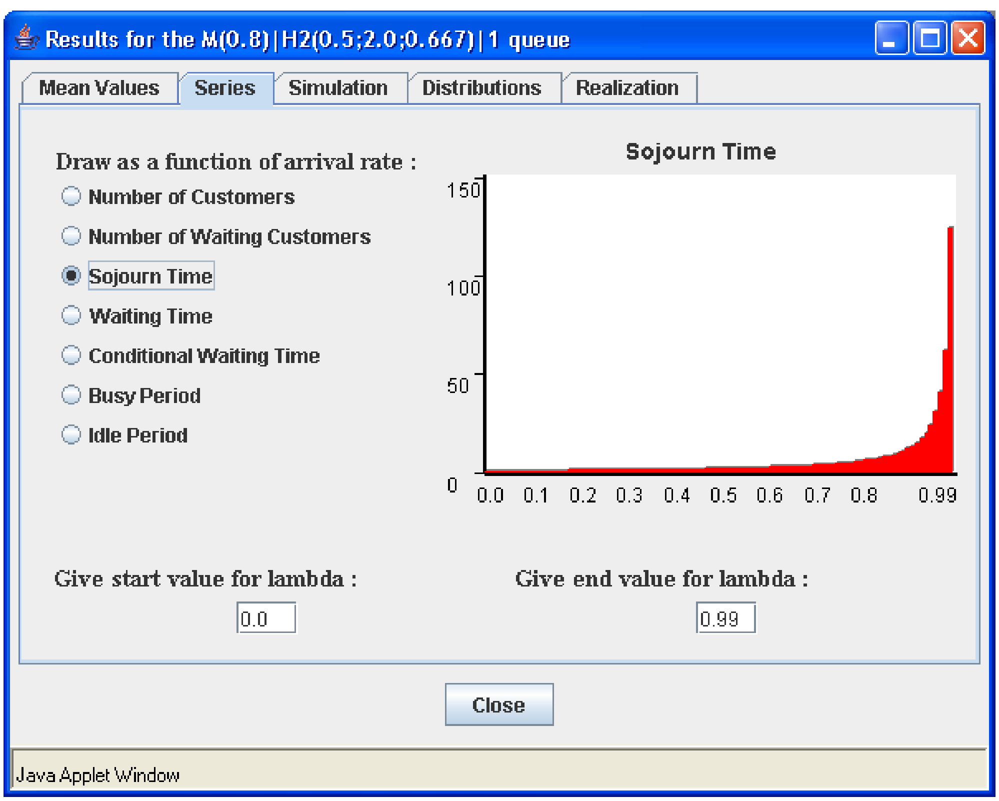
شکل 5 – 2: مقادیر متوسط برای M/H2/1



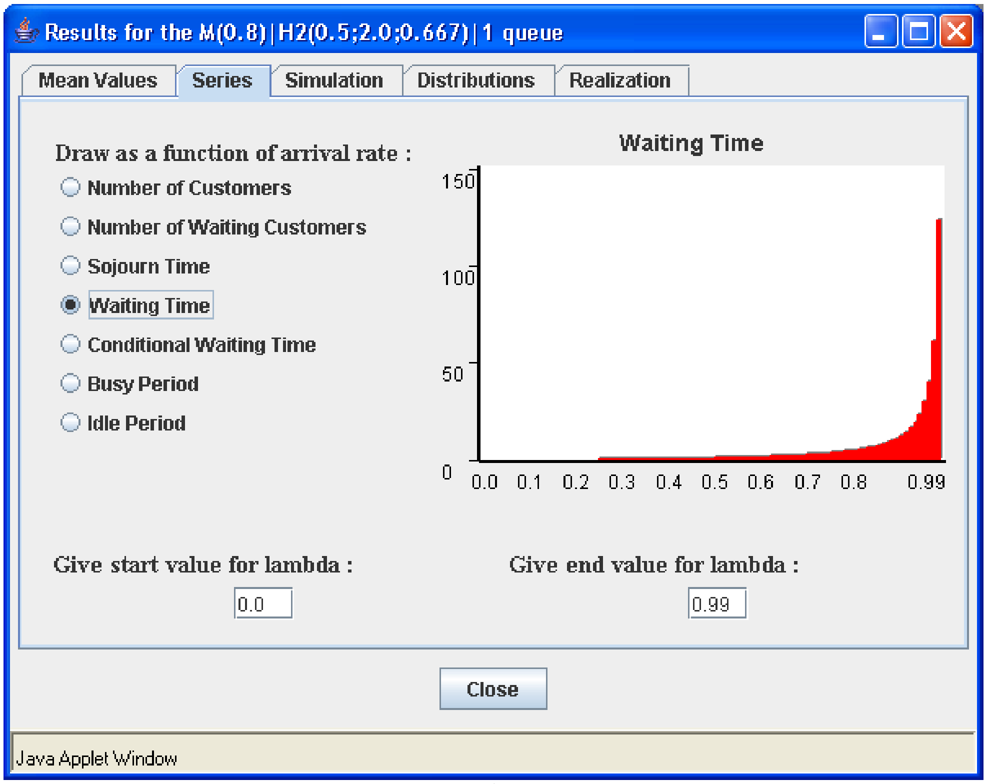
شکل 5 – 3: تعداد مشتری ها بصورت تابعی از λ



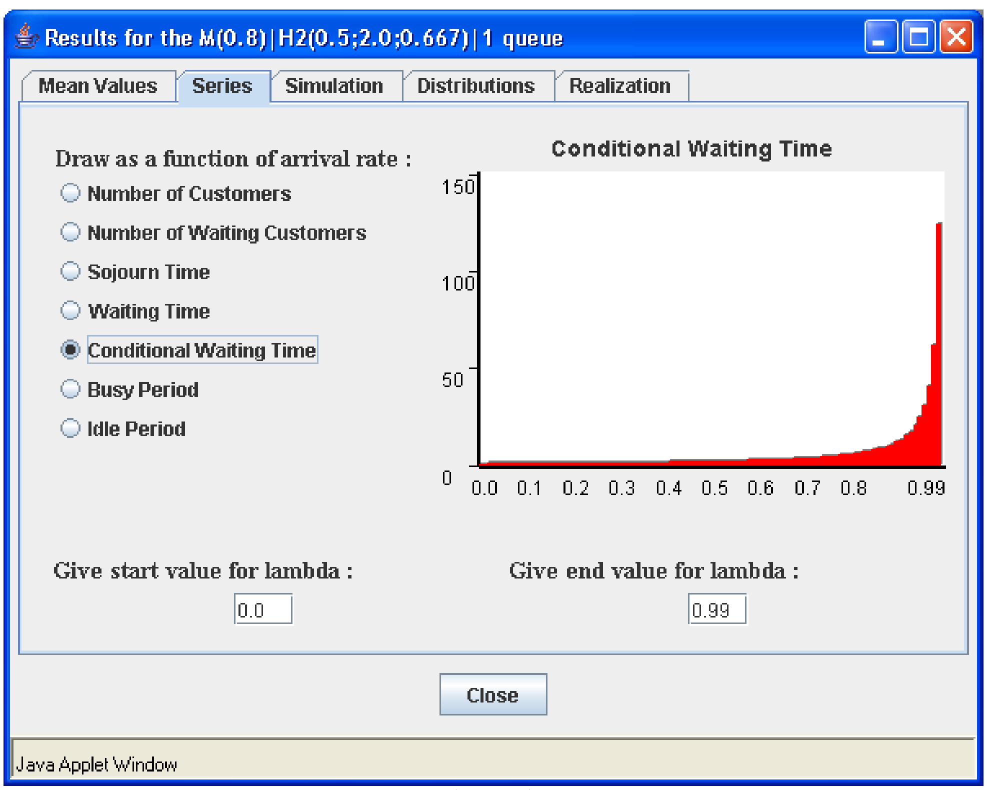
شکل 5 – 4: تعداد مشتری های منتظر بصورت تابعی از λ



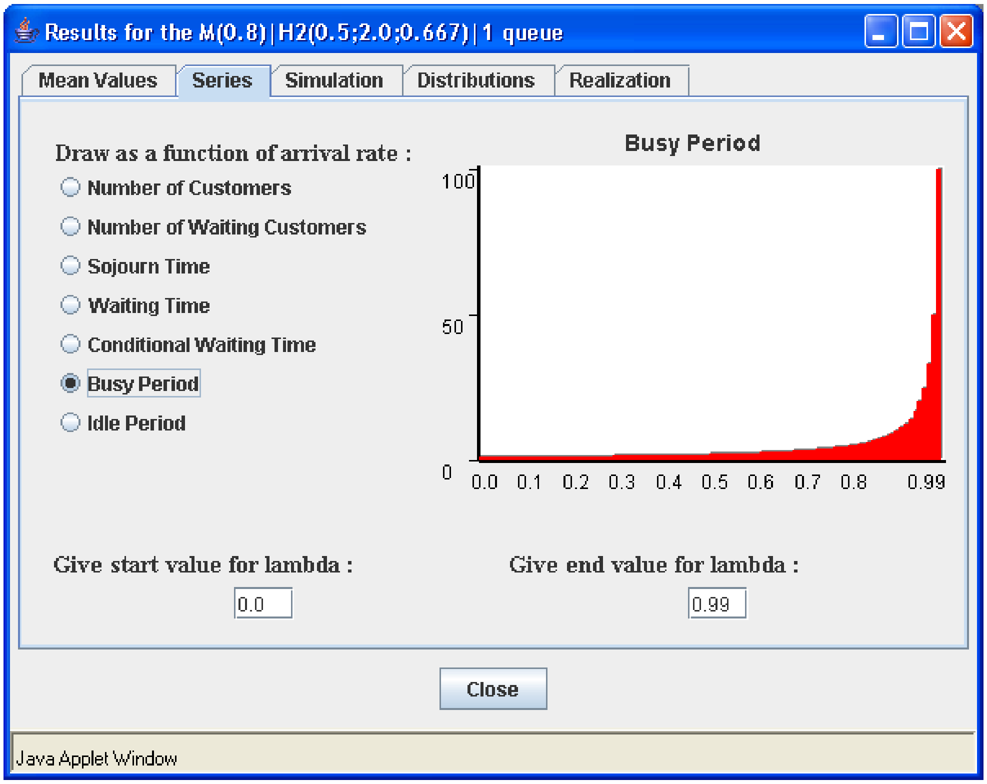
شکل 5 – 5: زمان اقامت بصورت تابعی از λ



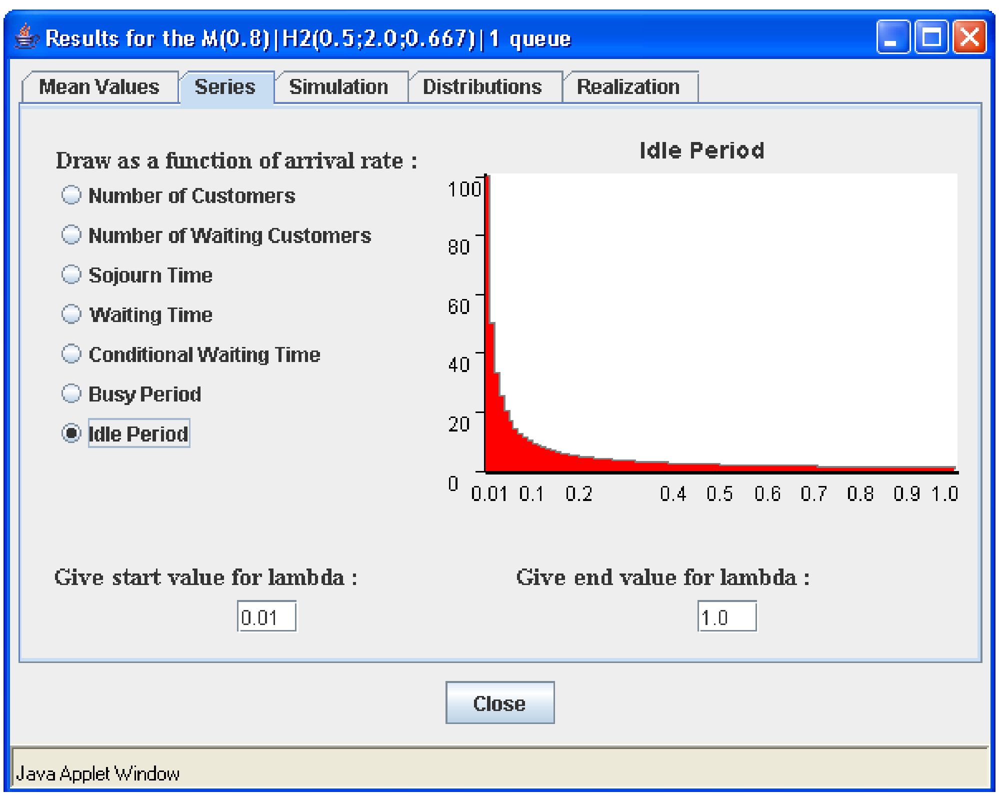
شکل 5 – 6: زمان انتظار بصورت تابعی از λ



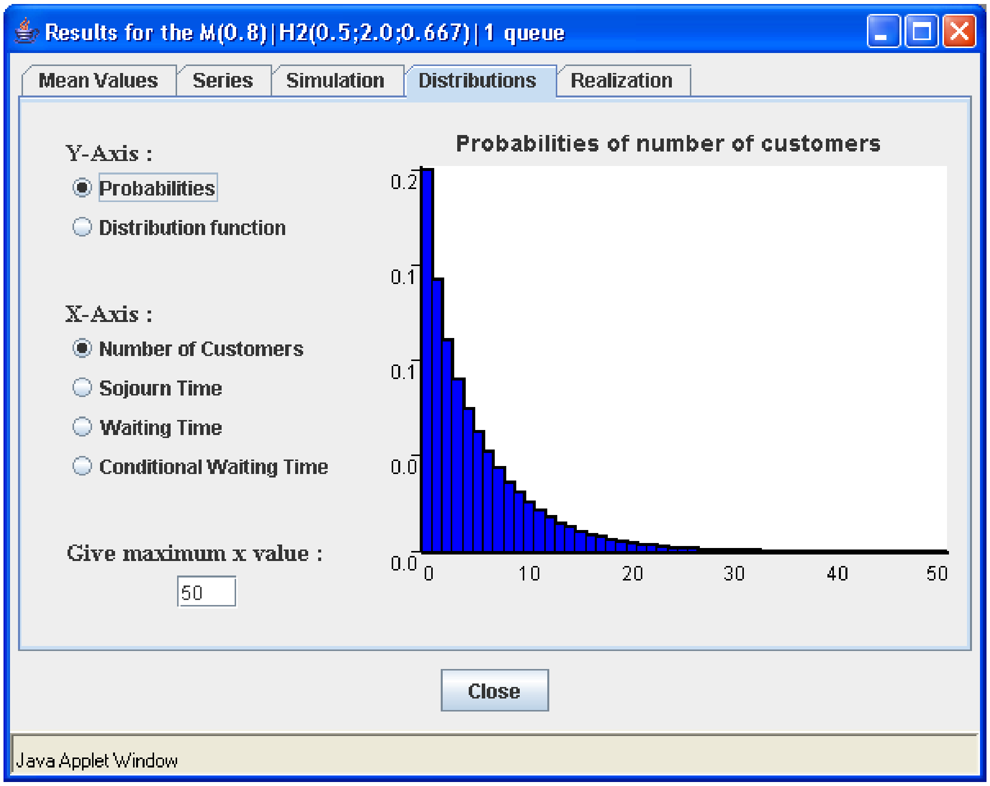
شکل 5 – 7: زمان انتظارشرطی بصورت تابعی از λ



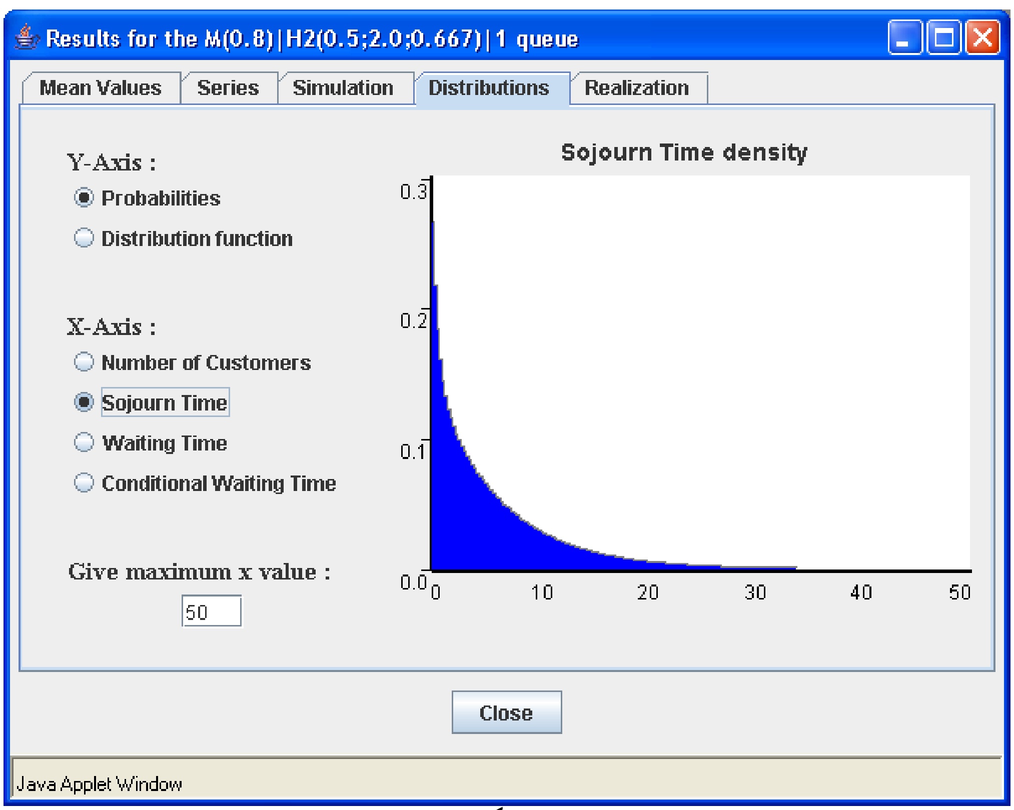
شکل 5 – 8: مدت زمان فعالیت بصورت تابعی از λ



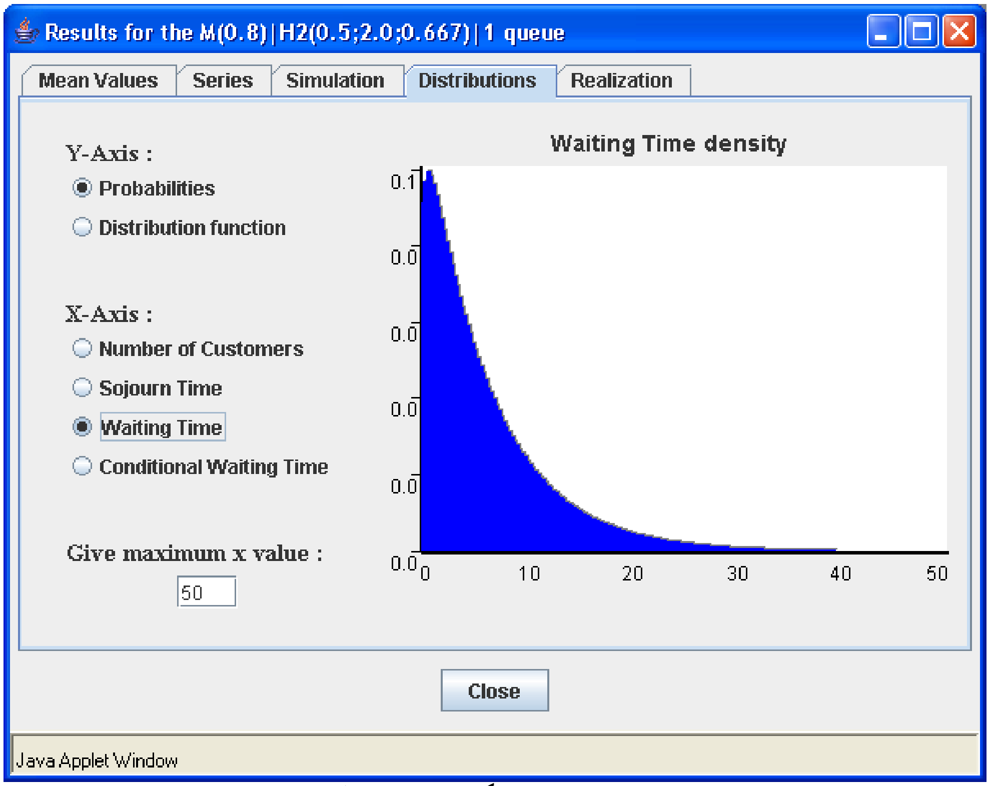
شکل 5 – 9: مدت زمان بیکاری بصورت تابعی از λ



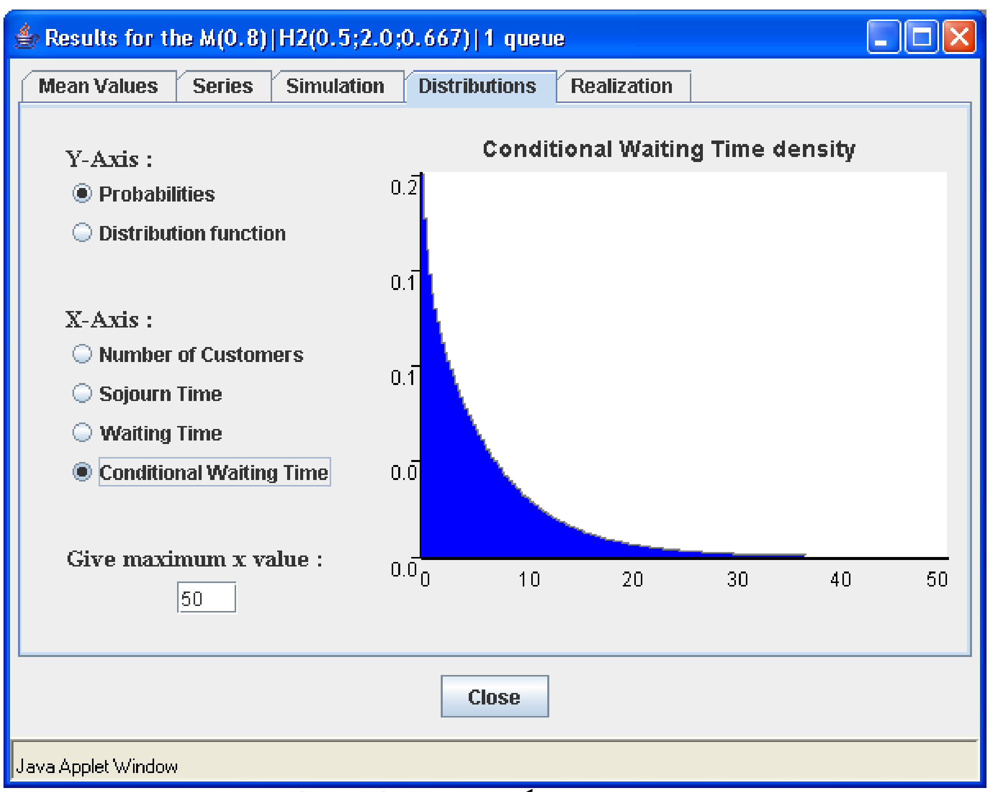
شکل 5 – 10: احتمال تعداد مشتری ها



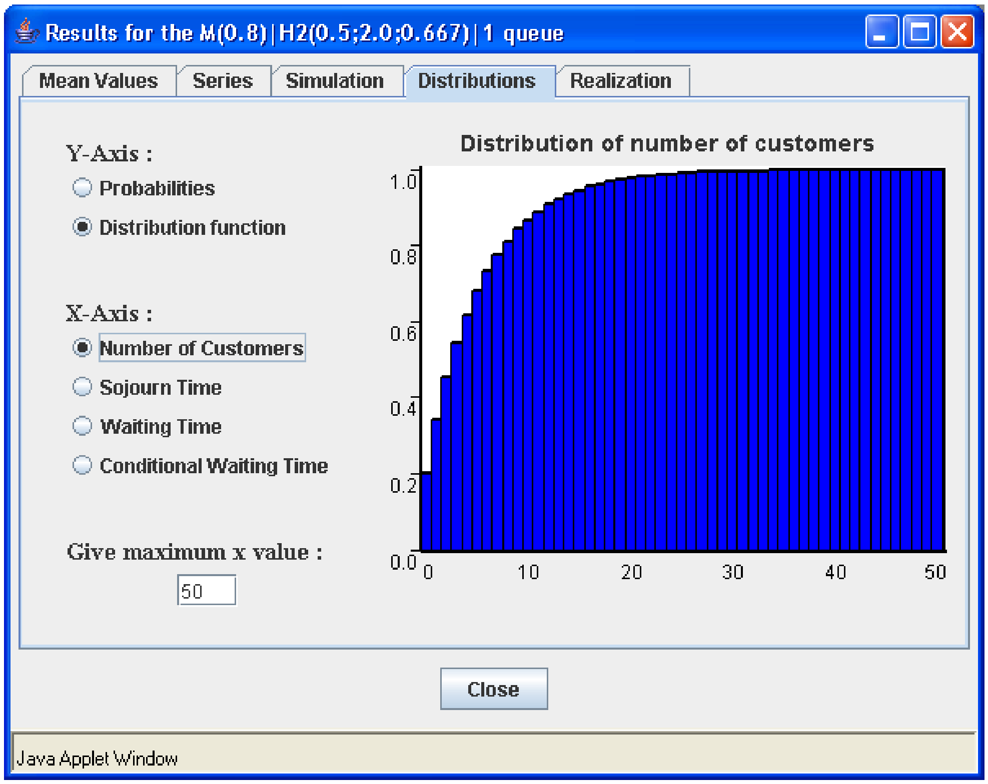
شکل 5 – 11: چگالی زمان اقامت



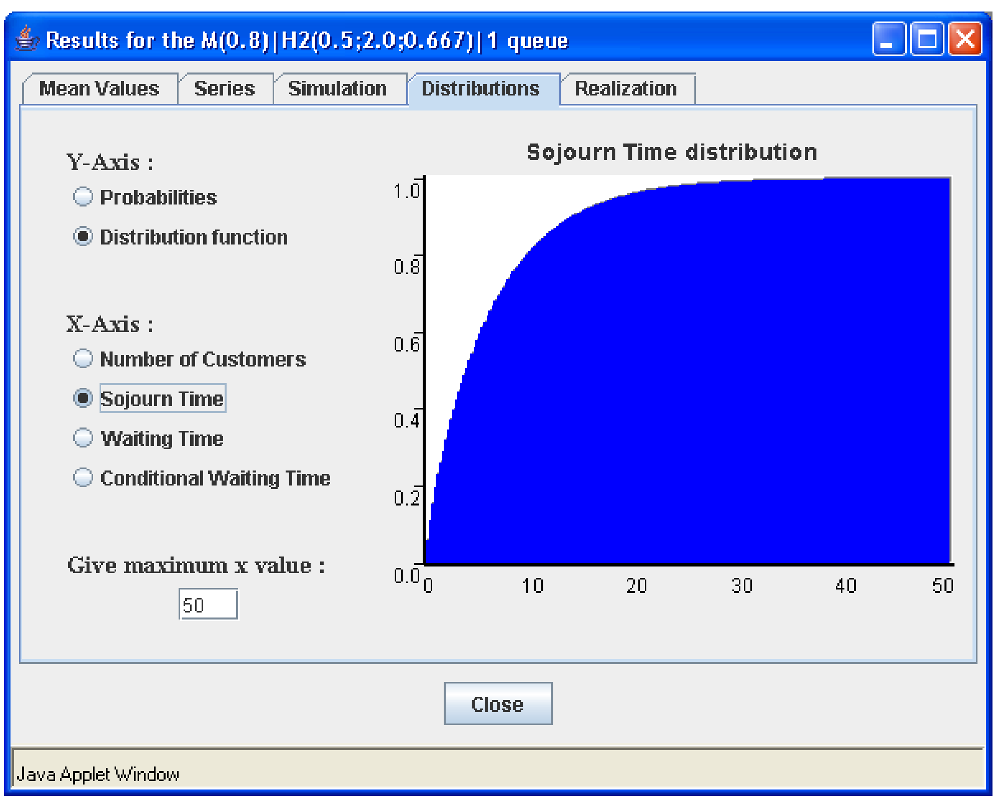
شکل 5 – 12: چگالی زمان انتظار



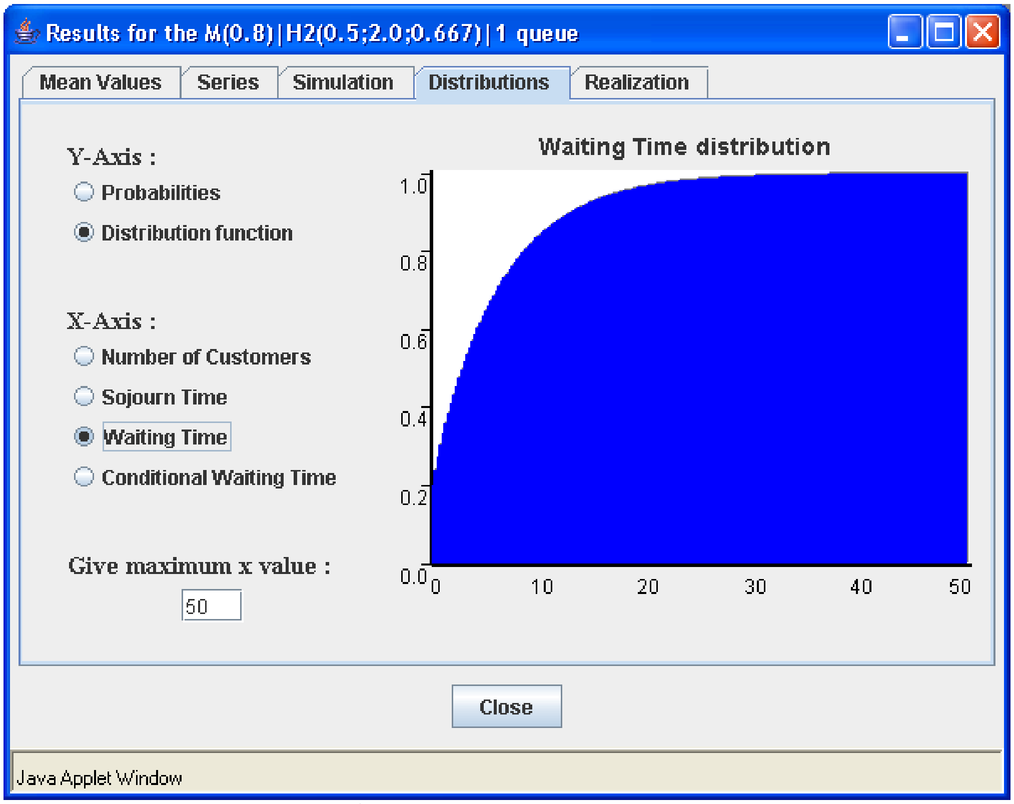
شکل 5 – 13: چگالی زمان انتظار شرطی



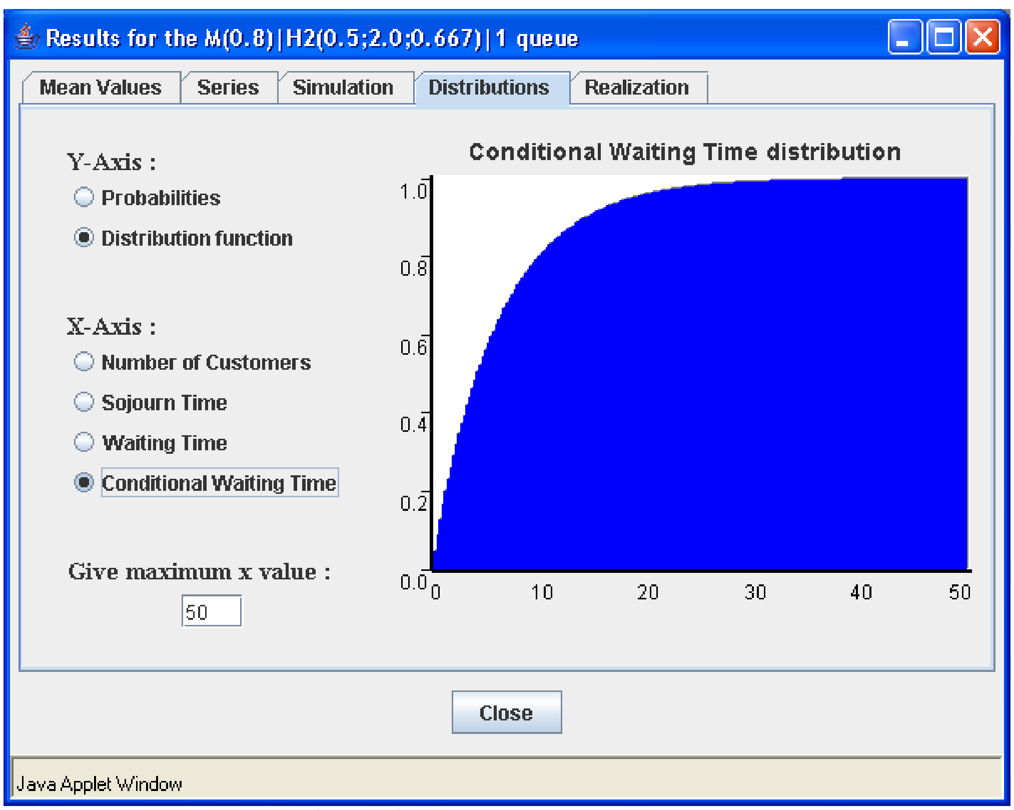
شکل 5 – 14: توزیع تعداد مشتری ها



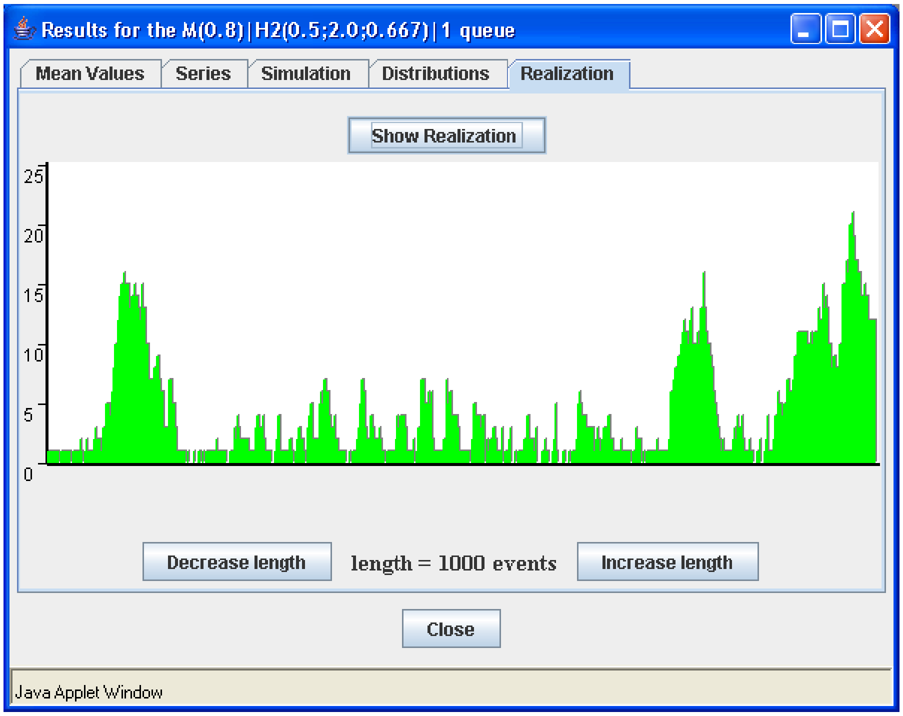
شکل 5 – 15: توزیع زمان اقامت



شکل 5 – 16: توزیع زمان انتظار

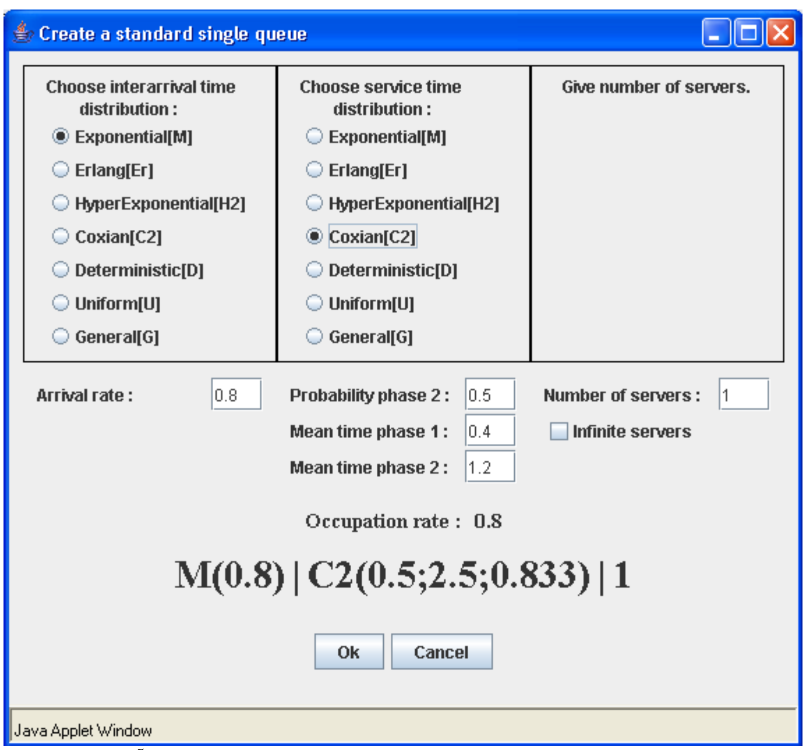


شکل 5 – 17: توزیع زمان انتظار شرطی

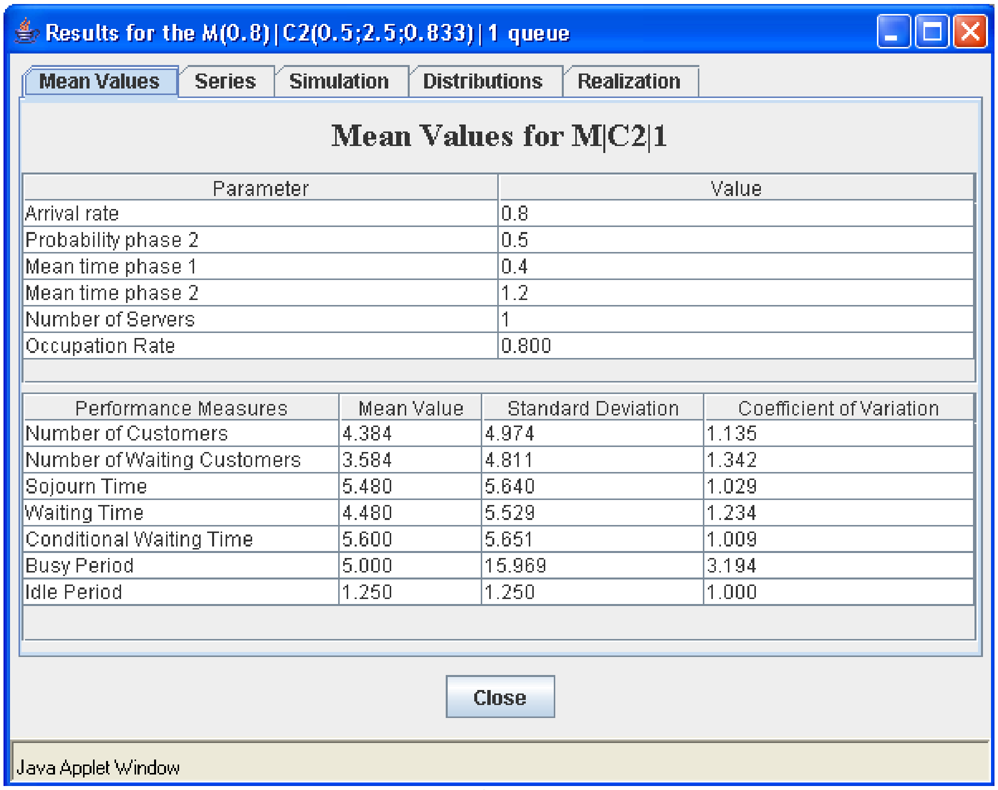


شکل 5 – 18: نمودار تحقق صف M/H2/1

5 – 2 صف M/Cox2/1



شکل 5 – 19: ابتدا یک صف M/Cox2/1 با مقادیر مشخص برای پارامترهای آن می سازیم. مقادیر در شکل مشخص شده اند.



Results for the M(0.8)|C2(0.5;2.5;0.833)|1 queue

Mean Values | Series | Simulation | Distributions | Realization

Mean Values for M|C2|1

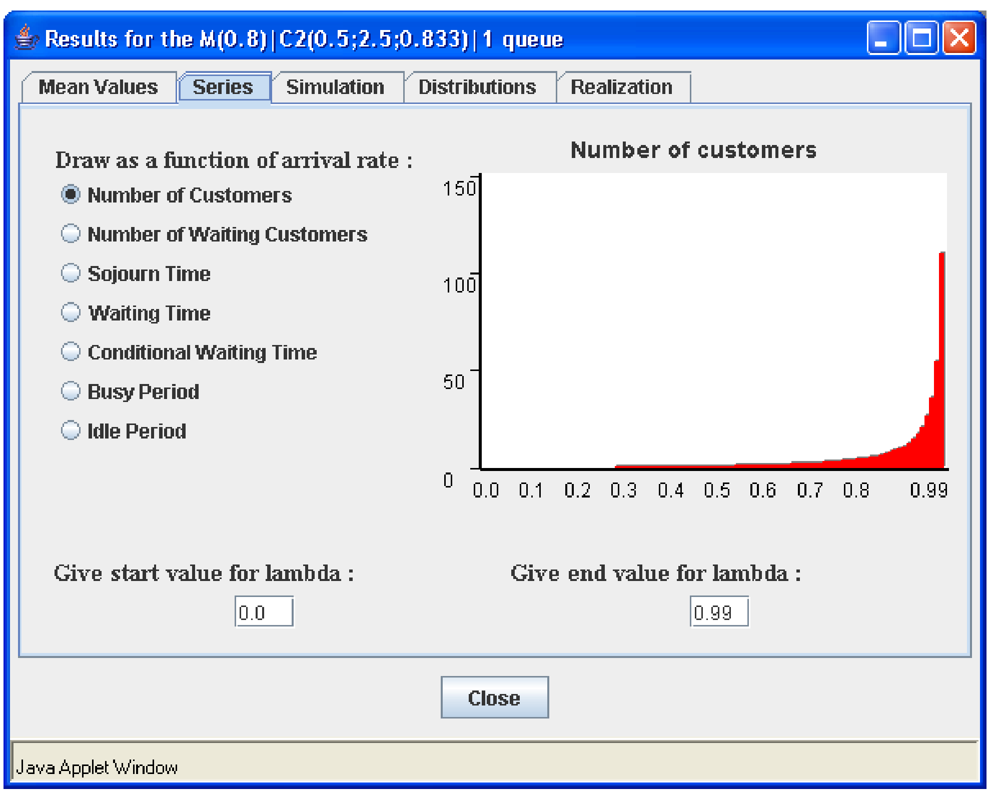
Parameter	Value
Arrival rate	0.8
Probability phase 2	0.5
Mean time phase 1	0.4
Mean time phase 2	1.2
Number of Servers	1
Occupation Rate	0.800

Performance Measures	Mean Value	Standard Deviation	Coefficient of Variation
Number of Customers	4.384	4.974	1.135
Number of Waiting Customers	3.584	4.811	1.342
Sojourn Time	5.480	5.640	1.029
Waiting Time	4.480	5.529	1.234
Conditional Waiting Time	5.600	5.651	1.009
Busy Period	5.000	15.969	3.194
Idle Period	1.250	1.250	1.000

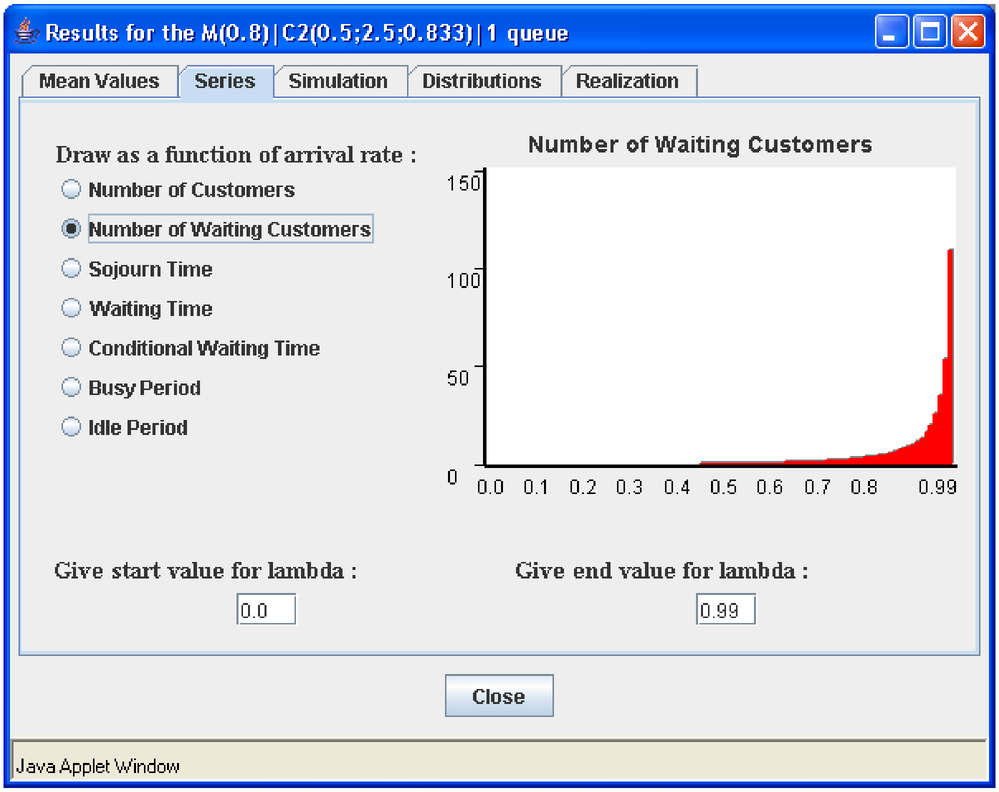
Close

Java Applet Window

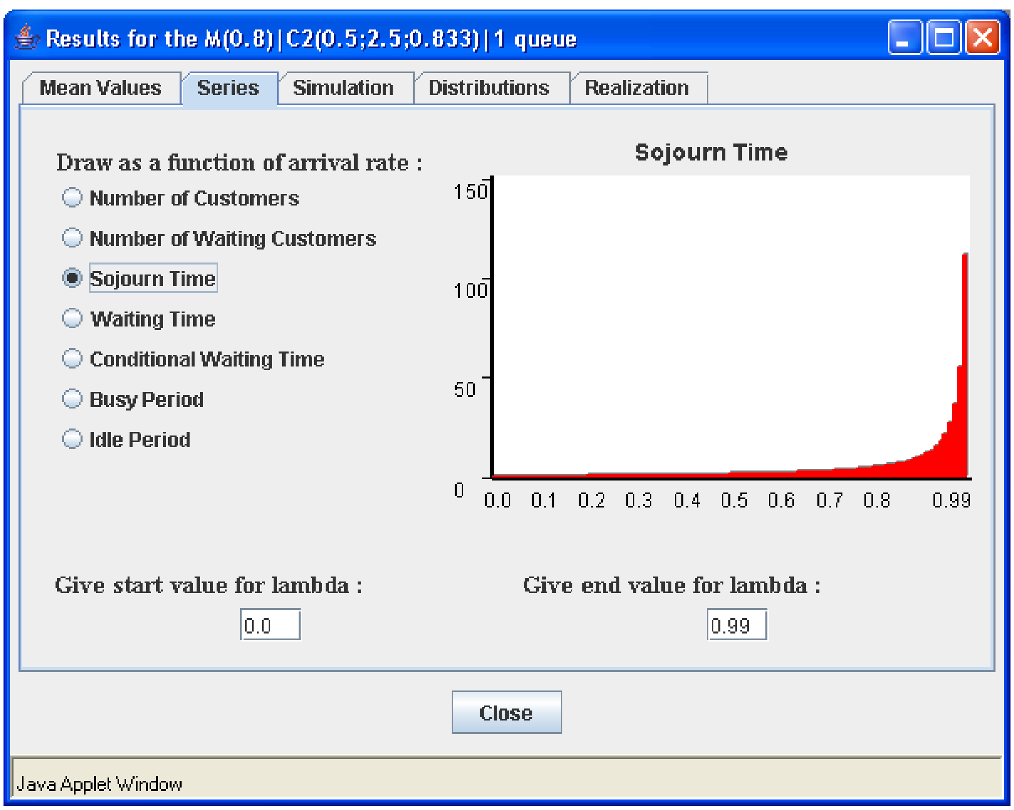
شکل 5 – 20: مقادیر متوسط برای صف M/Cox2/1



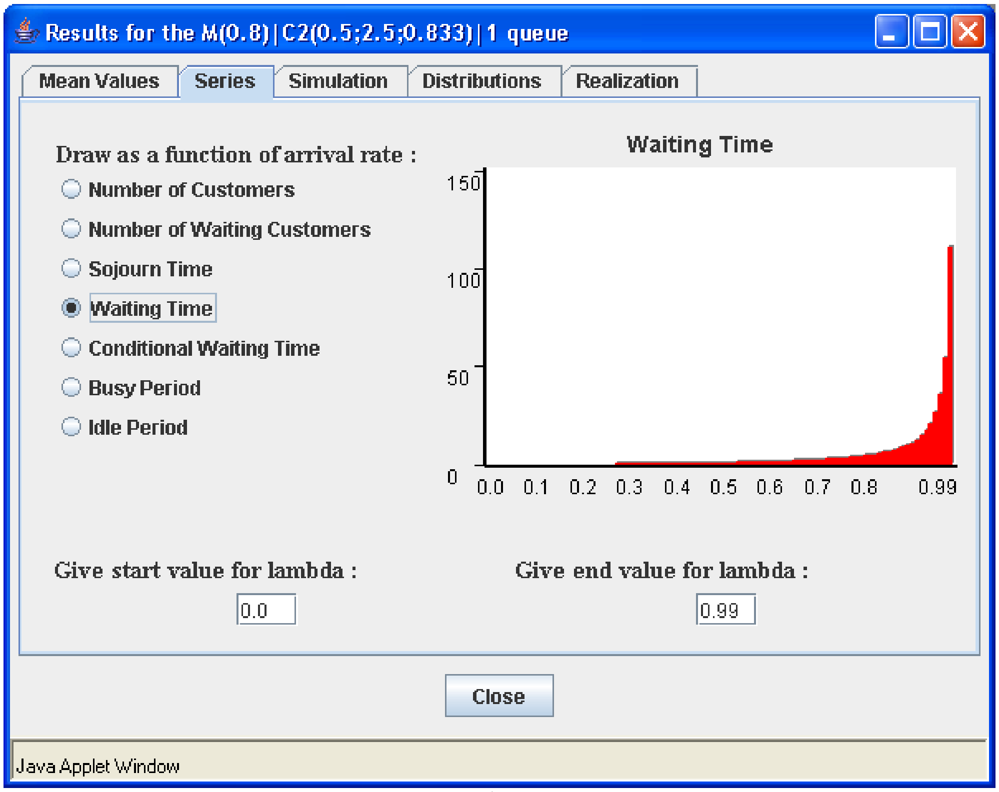
شکل 5 – 21: تعداد مشتری ها بصورت تابعی از λ



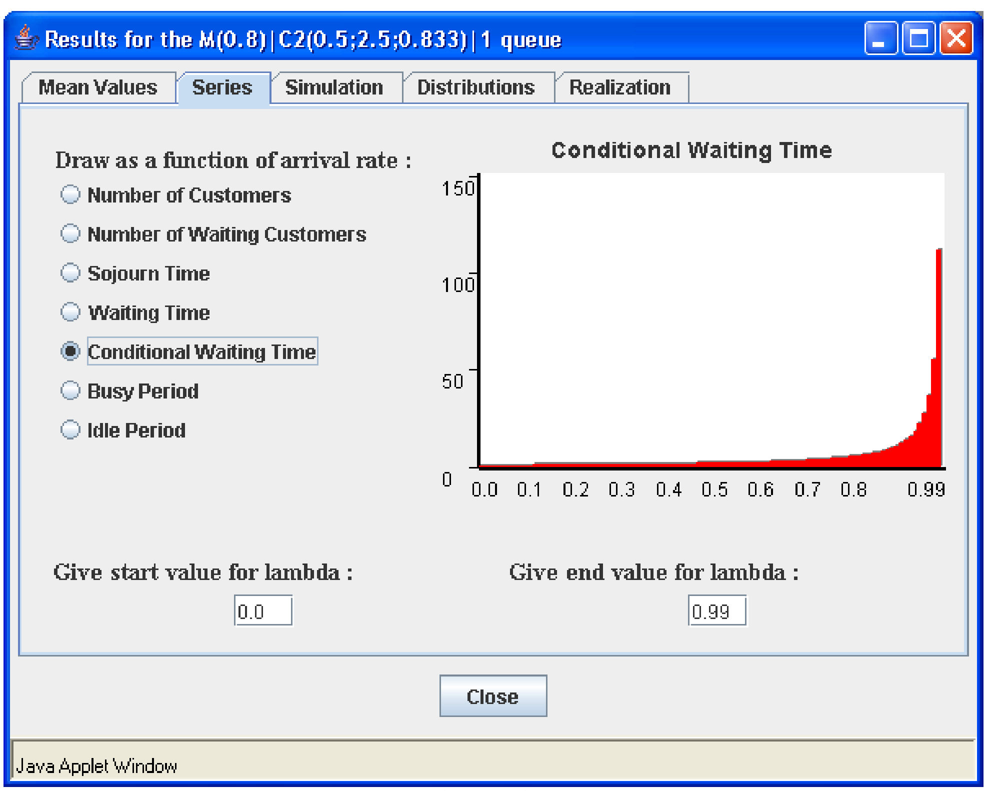
شکل 5 – 22: تعداد مشتری های منتظر بصورت تابعی از λ



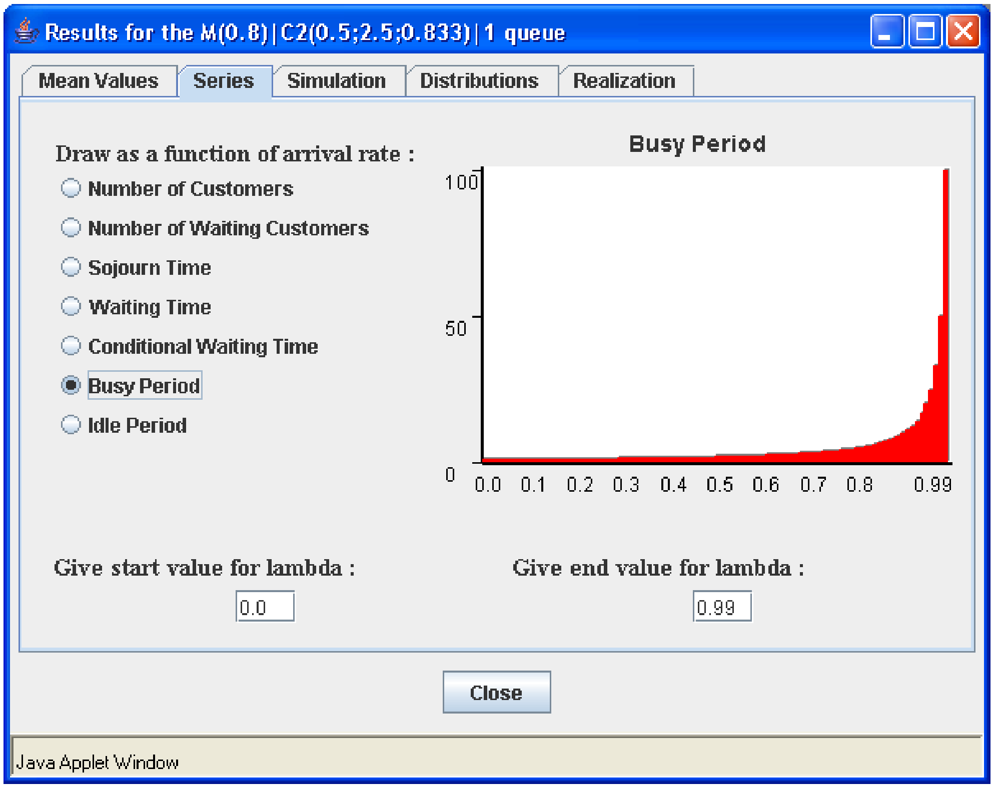
شکل 5 – 23: زمان اقامت بصورت تابعی از λ



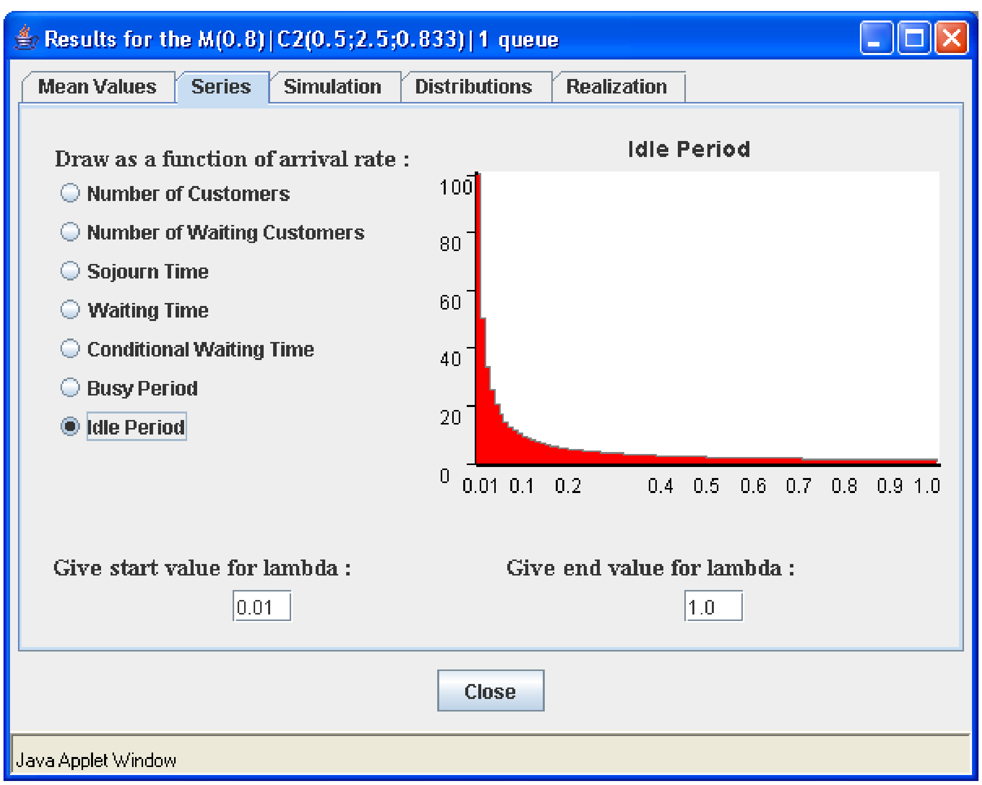
شکل 5 – 24: زمان انتظار بصورت تابعی از λ



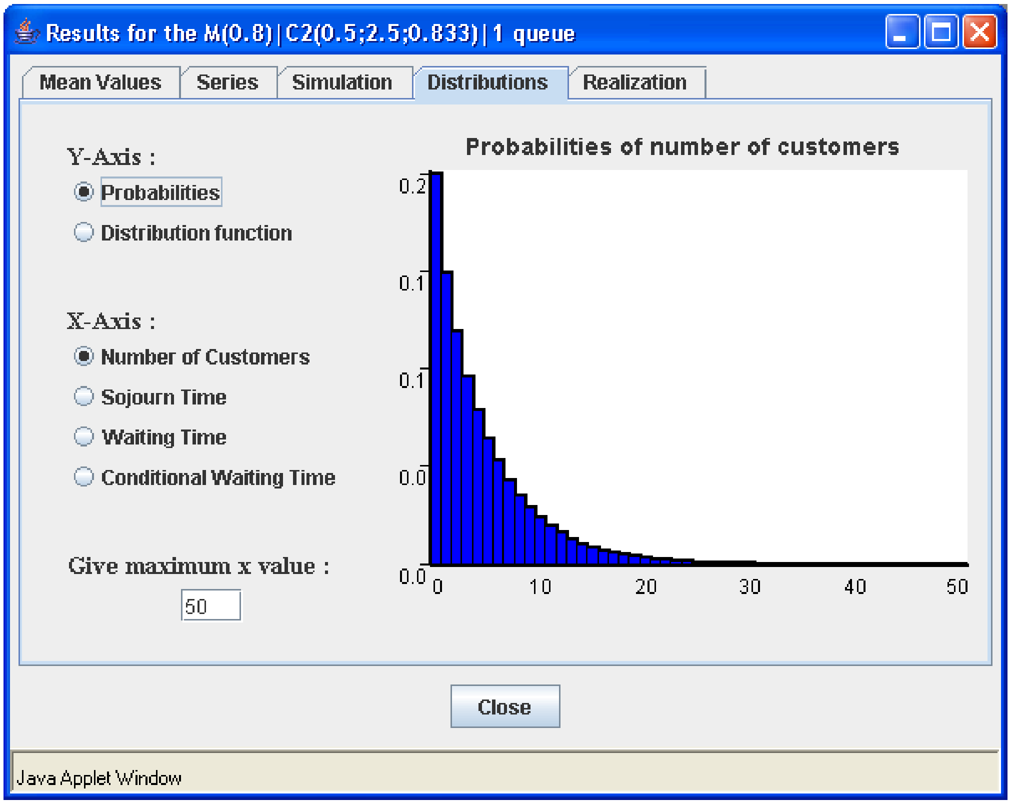
شکل 5 – 25: زمان انتظار شرطی بصورت تابعی از λ



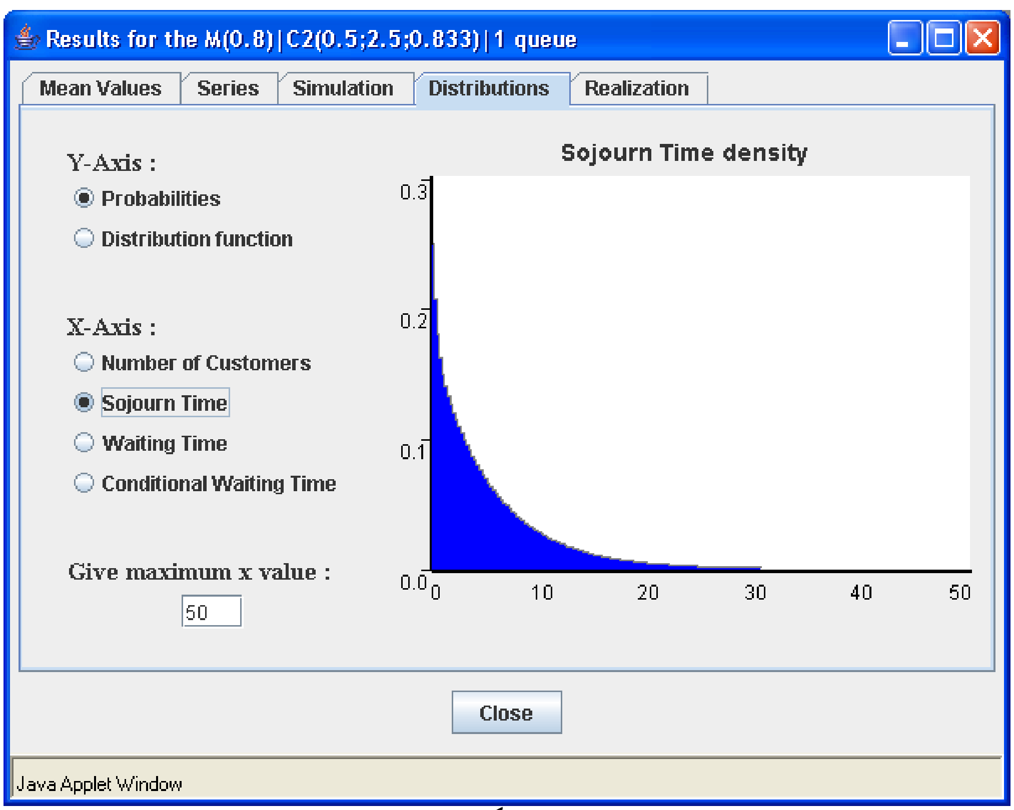
شکل 5 – 26: مدت زمان فعالیت بصورت تابعی از λ



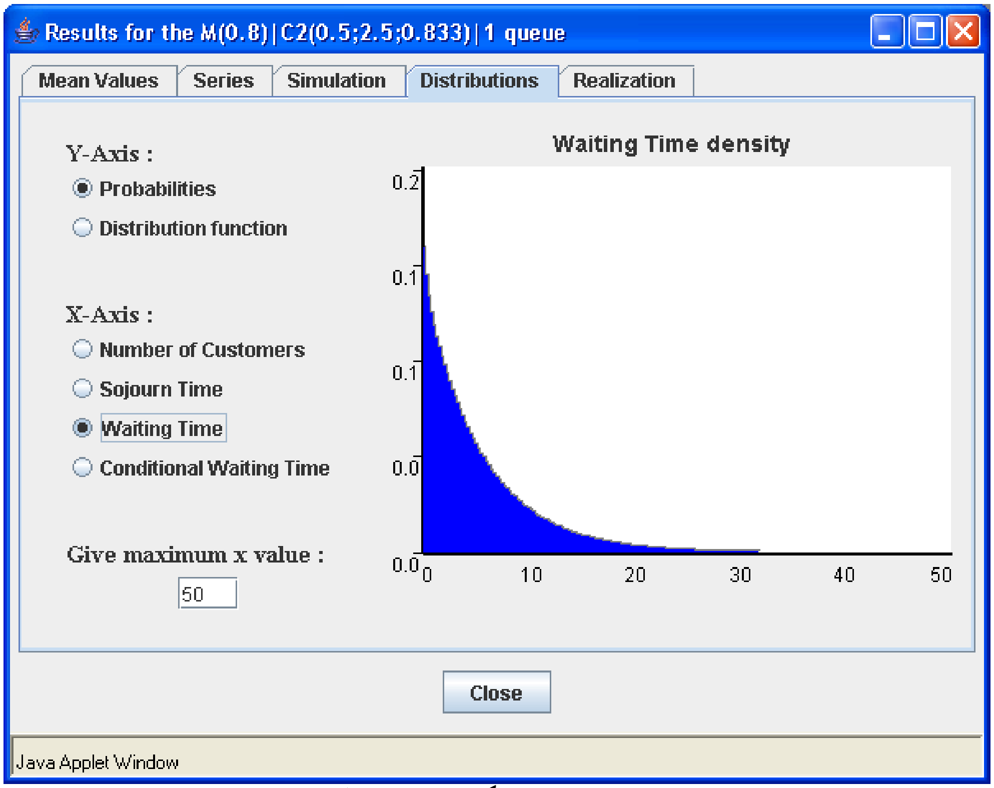
شکل 5 – 27: مدت زمان بیکاری بصورت تابعی از λ



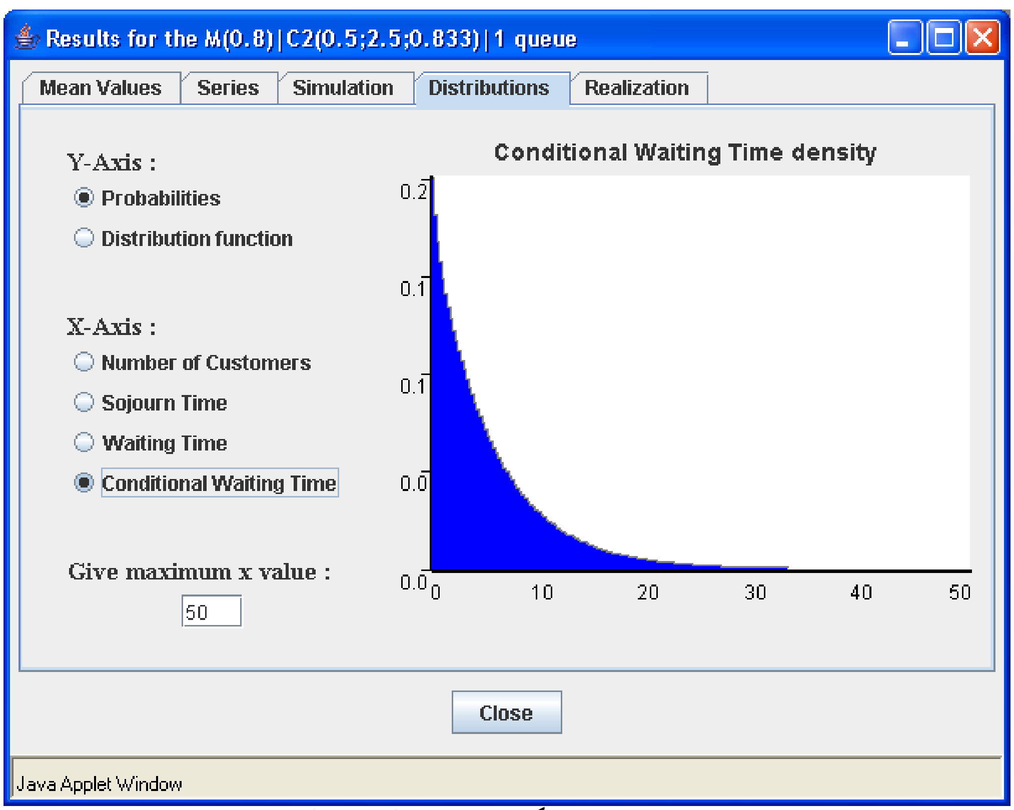
شکل 5 – 28: احتمال تعداد مشتری ها



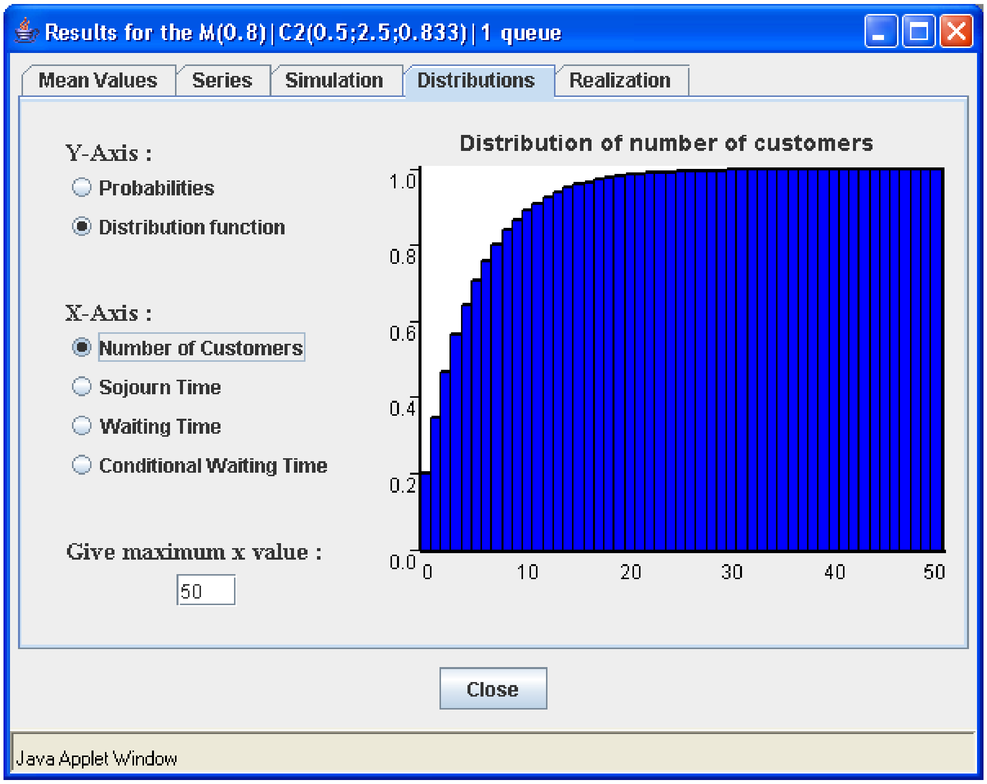
شکل 5 – 29: چگالی زمان اقامت



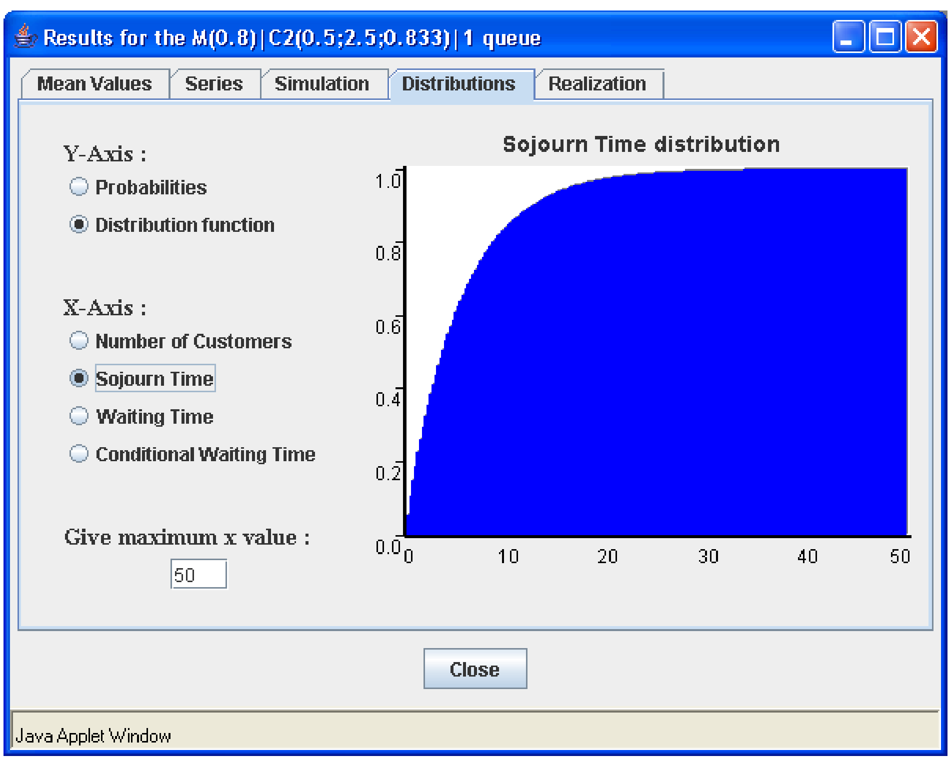
شکل 5 – 30: چگالی زمان انتظار



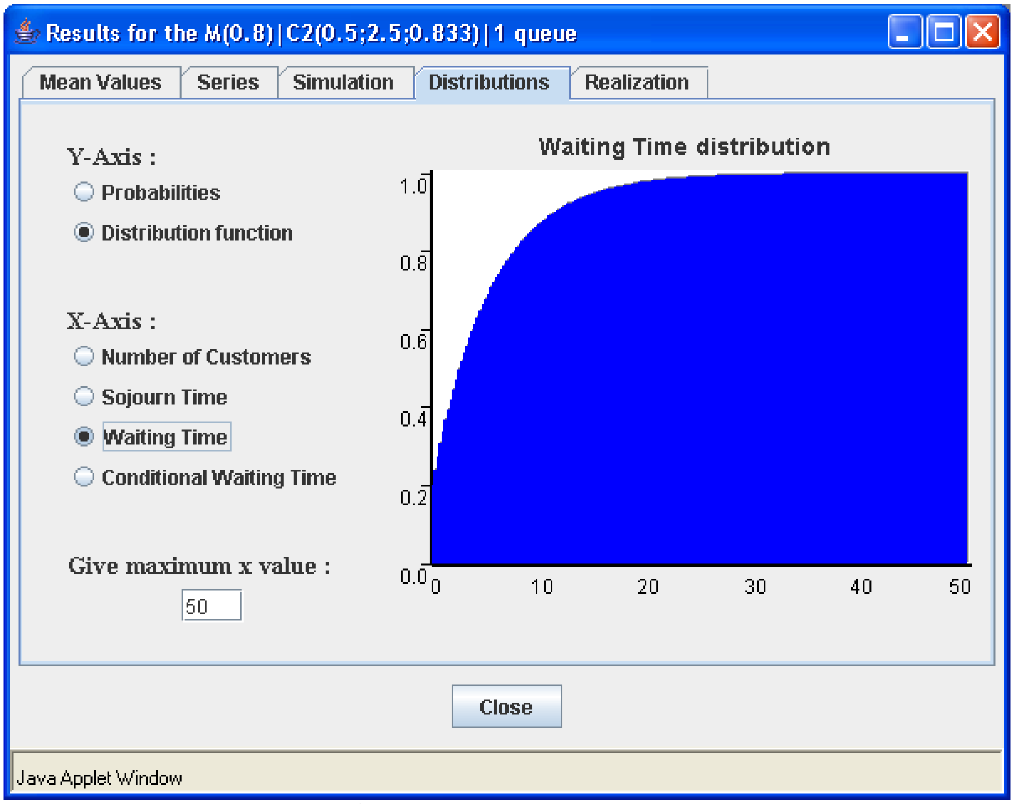
شکل 5 – 31: چگالی زمان انتظار شرطی



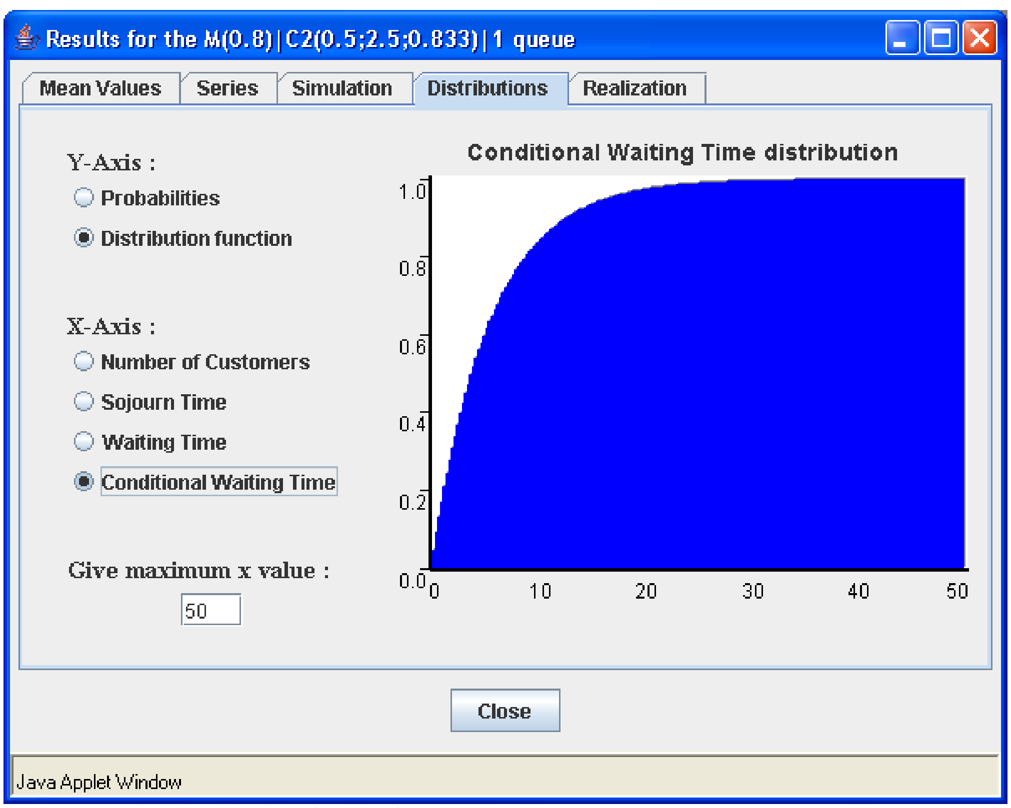
شکل 5 – 32: توزیع تعداد مشتری ها



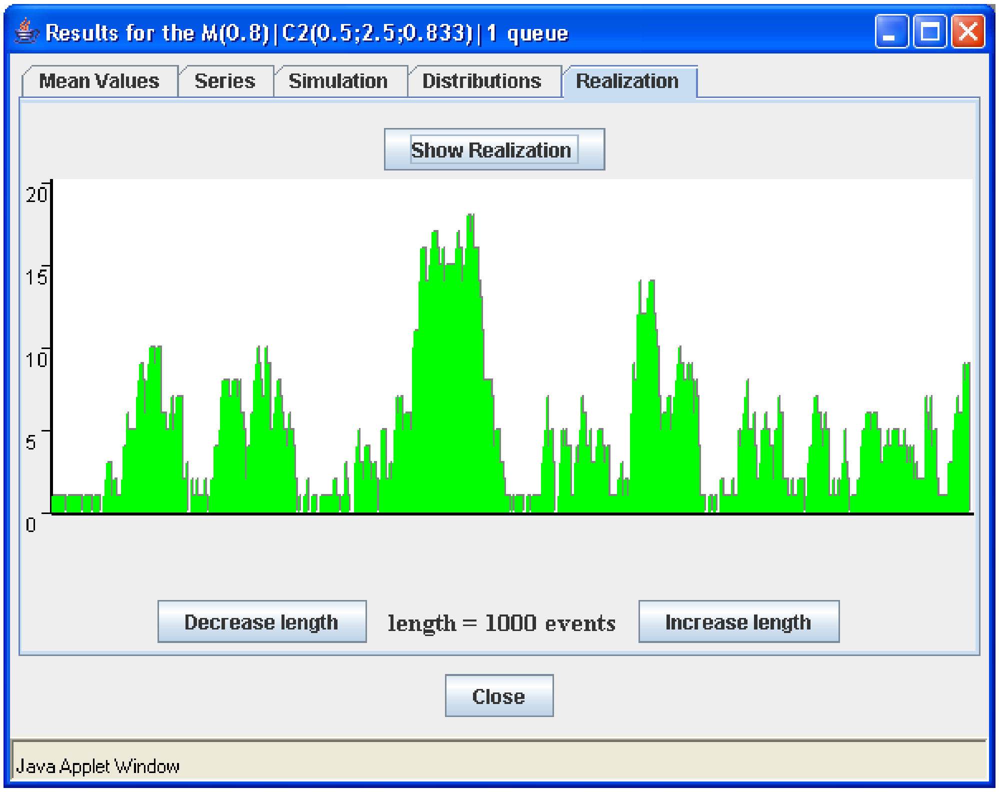
شکل 5 – 33: توزیع زمان اقامت



شکل 5 – 34: توزیع زمان انتظار



شکل 5 – 35: توزیع زمان انتظار شرطی



شکل 5 – 36: نمودار تحقق صف M/Cox2/1

5 – 3 صف MMPP/M/1

برای شبیه سازی صف MMPP/M/1 از یک برنامه MATLAB استفاده کرده ایم و آنرا به ازای مقادیر مختلف N و λ اجرا نموده ایم. متن برنامه و نمودار شبیه سازی و خروجی برنامه در زیر آورده شده است.

```
%%%% (MMPP/M/1 Queue)

N=30;
lambda_1=2;
ON=10; Z=20;


K=N+1;
L=diag([0:N]*lambda_1);

%%% Q matrix has Birth-Death Structure
Q=zeros(K,K);

for i=0:N-1,
        Q(i+1,i+2)=(N-i)/Z;   % new source becomes active
end;

for i=1:N,
        Q(i+1,i+1-1)=i/ON;     % active source enters OFF period
end;

for i=0:N,
        Q(i+1,i+1) = - sum(Q(i+1,:));  % diagonal elements of Q
end;

eps=ones(1,K);
```

```
pi=[zeros(1,K), 1] / [Q, eps'];  % solce pi*Q=0 and Pi*eps'=1
Lambda=pi*L*eps';


% validate pi
disp(pi*Q); disp(pi*eps');  % should be zero vector respectively 1

%%%%%% MMPP/M/1 Queue
nu=30;
I=eye(K);

A1=Q-L-nu*I; A0=L; A2=nu*I;

t2=cputime;   % time measurement for obtaining R via simple iteration
Ri=zeros(K,K);  % initial value
converged=1e-12;   % threshold for convergence
actual_value=1;
X1=-A0/A1;
X2=-A2/A1;

i=0;   % count iterations
while actual_value > converged,   % iterate
  Ri=X1+Ri^2*X2;
actual_value=norm(A0+Ri*A1+Ri^2*A2,1);
     i=i+1;
     end;
R2=Ri;
t2=cputime-t2;
disp(t2);  % computation duration
disp(i);   % number of iterations
disp(norm(A0+R2*A1+R2^2*A2));

%%%%%%% Spectral Expansion
t3=cputime;

[U0,l0]=eig([zeros(K,K), -(A0/A2); ...
                   eye(K), -(A1/A2)].');
% use transposed of matrix, since MATLAB function computes the Evectors
% when multiplied by the matrix from the left, X*v'=lambda v'
% (and we ant the others, u*X = lambda u)

l0=diag(l0);

% use K smallest EValues (<1) and first half of Evectors
[dummy,i]=sort(abs(l0));
lam=l0(i(1:K)).';

U=U0(1:K,i(1:K)).';  % first half of columns of X
R3=(U\diag(lam))*U;

t3=cputime-t3;
disp(t3);  % computation duration
disp(norm(A0+R3*A1+R3^2*A2));


%%% Queue-Length Probabilities (using R3)
%%% Note: using the spectral decomposition of R3 would be more efficient
%%% (but not done here)

R=R3; nmax=100;
r=zeros(1,nmax+1);  % scalar queue-length probabilities

n=0:nmax;
pi_k=pi*(I-R);
r(1)=sum(pi_k);

for i=1:nmax,
   pi_k=pi_k*R;    % matrix geometric factors to compute pi_{i+1} from pi_i
   r(i+1)=sum(pi_k);
end;

qbar=(pi*R)*((I-R)\eps');
```

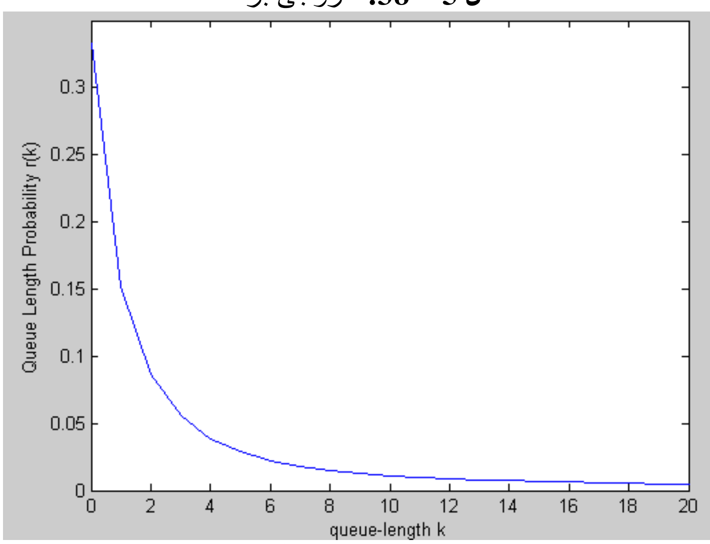
```
disp(qbar);

figure;
plot(n,r,'b-');
axis([0,20,0,0.35]);
xlabel('queue-length k');
ylabel('Queue Length Probability r(k)');
title(MMPP/M/1 Queue');
```

شکل 5 – 37: برنامه مطلب برای شبیه سازی صف MMPP/M/1

5 – 3 – 1 نتیجه اجرا برای N = 10, λ = 6

```
1.0e-016 *

 Columns 1 through 6

  0.0520    0.1388   -0.2776    0.1388   -0.8327    0.6245

 Columns 7 through 11

  0.5031    0.2082    0.2380   -0.7128    0.0975

  1.0000

  0.1719

     2684

 1.0044e-012

  0.0313

 1.1099e-013

 13.2788
```

شکل 5 – 38: خروجی برنامه

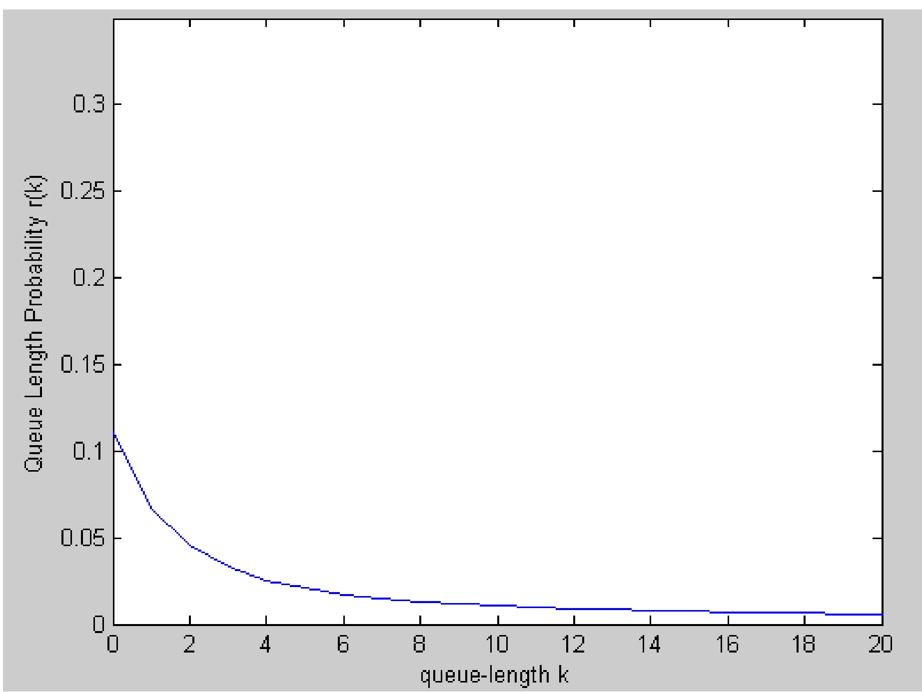


شکل 5 – 39: احتمال طول صف

5 – 3 – 2 نتیجه اجرا برای N = 20, λ = 4

```
1.0e-015 *

 Columns 1 through 6

 -0.0249   -0.0199   -0.0382   -0.0069   -0.0139    0.1527

 Columns 7 through 12

  0.0278   -0.1804   -0.0971    0.2012    0.0971    0.0919

 Columns 13 through 18

 -0.0087   -0.0169   -0.0674   -0.0360   -0.0533   -0.0199

 Columns 19 through 21

  0.0172    0.0171    0.0072

   1

  1.3906

     7187

 6.8005e-013

  0.0313

 3.9496e-011

  93.6558
```

شکل 5 – 40: خروجی برنامه

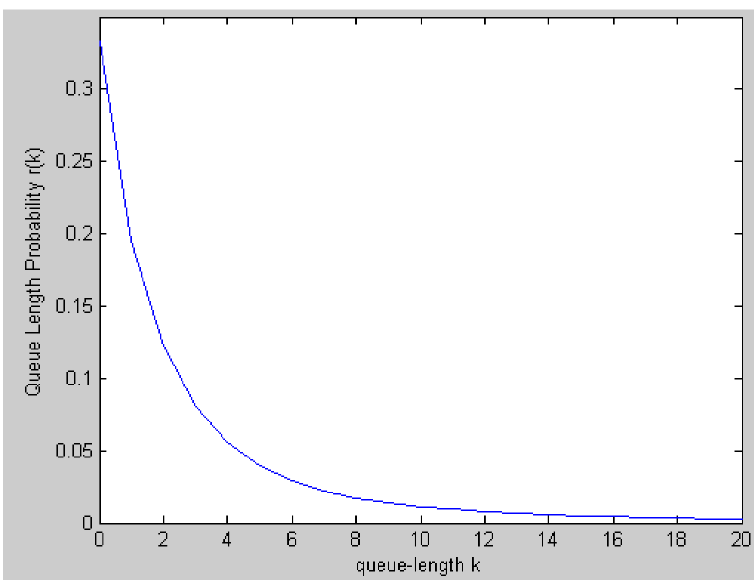


شکل 5 – 41: احتمال طول صف

5 – 3 – 3 نتیجه اجرا برای $N = 30, \lambda = 2$

```
1.0e-015 *

 Columns 1 through 6

  -0.0442   -0.0012    0.0360    0.0139    0.0798   -0.0104

 Columns 7 through 12

   0.0833   -0.0555    0.0278   -0.0833    0.0555   -0.1388

 Columns 13 through 18

   0.0971    0.0694    0.0416   -0.1284    0.0173   -0.0139

 Columns 19 through 24

  -0.0301    0.0156   -0.0143   -0.0064   -0.0121   -0.0079

 Columns 25 through 30

  -0.0191   -0.0456   -0.0319    0.0123    0.0248   -0.0017

 Column 31

   0.0067

   1.0000

   0.6094

       1306

 1.1204e-012

   0.0156

 5.5698e-009

   3.5978
```

شکل 5 – 42: خروجی برنامه

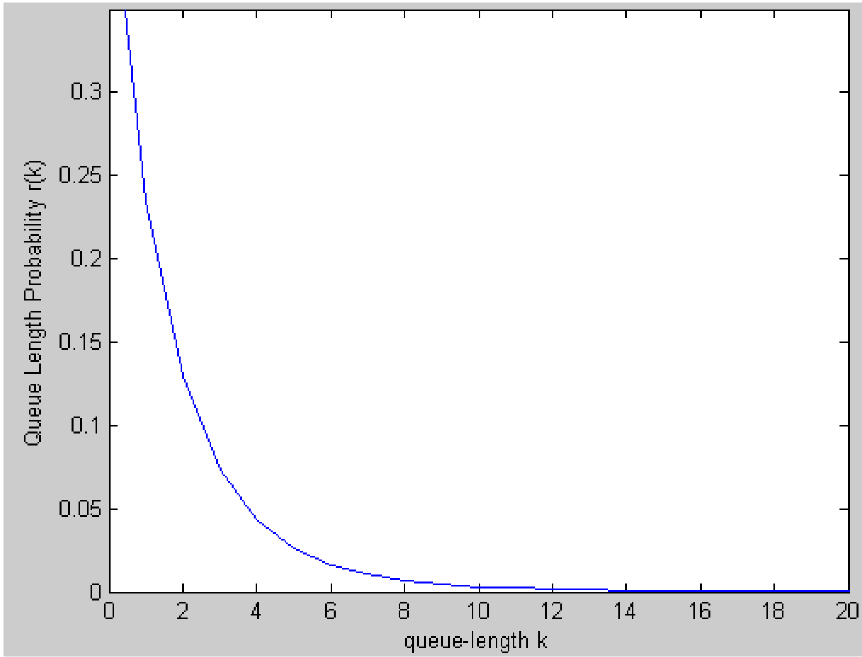


شکل 5 – 43: احتمال طول صف

5 – 3 – 4 نتیجه اجرا برای $N = 50, \lambda = 1$

```
1.0e-015 *

  Columns 1 through 6

  -0.0402    0.0119    0.0282   -0.1361   -0.0272   -0.0441

  Columns 7 through 12

  -0.0556    0.0061   -0.0789    0.0278    0.0416    0.1388

  Columns 13 through 18

  -0.0278    0.0833    0.3053   -0.0555    0.2776   -0.0833

  Columns 19 through 24

        0    0.2220   -0.1804   -0.2776   -0.0069   -0.0035

  Columns 25 through 30

   0.1422   -0.0893   -0.0199   -0.1316    0.0503    0.1331

  Columns 31 through 36

   0.0705   -0.1425   -0.1023   -0.0838   -0.0137    0.0812

  Columns 37 through 42

   0.0854   -0.0558   -0.0174    0.1841    0.1583   -0.1058

  Columns 43 through 48

  -0.0836   -0.0086    0.0136   -0.0186   -0.0260   -0.0278

  Columns 49 through 51

  -0.0169   -0.0257   -0.0125

   1.0000

   1.2344

  591

  1.4531e-012

   0.0469

  1.0498e-004

   1.4172
```

شکل 5 – 44: خروجی برنامه



شکل 5 – 45: احتمال طول صف