DESEMPENHO E DIMENSIONAMENTO DE REDES

ASSIGNMENT GUIDE NO. 4

PERFORMANCE OF A PACKAGE SWITCHING CONNECTION

Work done by:

- Fábio Alves, Nmec: 84734
- Sérgio de Aguiar, Nmec: 84831

a) Data gathering:

```
Result = zeros(8,4);
Conf = zeros(8,4);

Case = zeros(10,4);

C=2;

for f = [10 1]
    for lambda = [1 2]
         for k = 1:10
            [PL , APD , MPD , TT] =
Simulator1(lambda*100,C,f*10000,1000);
            Case(k,1) = PL;
            Case(k,2) = APD;
            Case(k,3) = MPD;
            Case(k,4) = TT;
         end

         if f== 10
            Result(lambda , 1) = mean(Case(:,1));
            Conf(lambda , 1) = 1.645*std(Case(:, 1))/sqrt(10);
         else
            Result(lambda + 2, 1) = mean(Case(:,1));
            Conf(lambda + 2, 1) = 1.645*std(Case(:, 1))/sqrt(10);
         end

         if f== 10
            Result(lambda , 2) = mean(Case(:,2));
            Conf(lambda , 2) = 1.645*std(Case(:, 2))/sqrt(10);
         else
            Result(lambda + 2, 2) = mean(Case(:,2));
            Conf(lambda + 2, 2) = 1.645*std(Case(:, 2))/sqrt(10);
         end

         if f== 10
            Result(lambda , 3) = mean(Case(:,3));
            Conf(lambda , 3) = 1.645*std(Case(:, 3))/sqrt(10);
         else
            Result(lambda + 2, 3) = mean(Case(:,3));
            Conf(lambda + 2, 3) = 1.645*std(Case(:, 3))/sqrt(10);
         end

         if f== 10
            Result(lambda , 4) = mean(Case(:,4));
            Conf(lambda , 4) = 1.645*std(Case(:, 4))/sqrt(10);
         else
            Result(lambda + 2, 4) = mean(Case(:,4));
            Conf(lambda + 2, 4) = 1.645*std(Case(:, 4))/sqrt(10);
         end

        disp(Result);
        disp(Conf);
     end
end
```

```
C=10;

for f = [10 1]
    for lambda = [1 2]
         for k = 1:10
            [PL , APD , MPD , TT] =
Simulator1(lambda*500,C,f*10000,1000);
            Case(k,1) = PL;
            Case(k,2) = APD;
            Case(k,3) = MPD;
            Case(k,4) = TT;
         end

         if f== 10
            Result(4+lambda , 1) = mean(Case(:,1));
            Conf(4+lambda , 1) = 1.645*std(Case(:, 1))/sqrt(10);
         else
            Result(4+lambda + 2, 1) = mean(Case(:,1));
            Conf(4+lambda + 2, 1) = 1.645*std(Case(:, 1))/sqrt(10);
         end

         if f== 10
            Result(4+lambda , 2) = mean(Case(:,2));
            Conf(4+lambda , 2) = 1.645*std(Case(:, 2))/sqrt(10);
         else
            Result(4+lambda + 2, 2) = mean(Case(:,2));
            Conf(4+lambda + 2, 2) = 1.645*std(Case(:, 2))/sqrt(10);
         end

         if f== 10
            Result(4+lambda , 3) = mean(Case(:,3));
            Conf(4+lambda , 3) = 1.645*std(Case(:, 3))/sqrt(10);
         else
            Result(4+lambda + 2, 3) = mean(Case(:,3));
            Conf(4+lambda + 2, 3) = 1.645*std(Case(:, 3))/sqrt(10);
         end

         if f== 10
            Result(4+lambda , 4) = mean(Case(:,4));
            Conf(4+lambda , 4) = 1.645*std(Case(:, 4))/sqrt(10);
         else
            Result(4+lambda + 2, 4) = mean(Case(:,4));
            Conf(4+lambda + 2, 4) = 1.645*std(Case(:, 4))/sqrt(10);
         end

        disp(Result);
        disp(Conf);
     end
 end
```

Case	λ data (pps)	C (Mbps)	f (Bytes)	Packet Loss (%)	Average Delay (msec)	Maximum Delay (msec)	Transm. Through. (Mbps)
A	100	2	100000	0 +/- 0	4.5336 +/- 0.1062	21.2186 +/- 1.6331	0.6615 +/- 0.0126
B	200	2	100000	0 +/- 0	7.7374 +/- 0.5869	34.7860 +/- 4.9402	1.3241 +/- 0.0253
C	100	2	10000	0 +/- 0	4.5520 +/- 0.0826	20.9207 +/- 1.6136	0.6733 +/- 0.0135
D	200	2	10000	0.1193 +/- 0.0801	8.2376 +/- 0.4210	38.6340 +/- 2.7430	1.3359 +/- 0.0304
E	500	10	100000	0 +/- 0	0.8899 +/- 0.0089	4.0070 +/- 0.2884	3.3335 +/- 0.0508
F	1000	10	100000	0 +/- 0	1.5997 +/- 0.0709	7.3567 +/- 0.6594	6.7372 +/- 0.1493
G	500	10	10000	0 +/- 0	0.9035 +/- 0.0224	3.8494 +/- 0.2783	3.3501 +/- 0.0834
H	1000	10	10000	0.0599 +/- 0.0363	1.6091 +/- 0.0930	7.4520 +/- 0.4553	6.6840 +/- 0.0984

b)

- **packet rate:** More packets can lead to a bigger packet loss, especially when the queue size reduces, as well as an increased packet delay due to this increased loss. By increasing the delays overall, it also leads to an increased maximum delay. Overall throughput though seems to double with the packet rate doubling.

- **connection capacity:** By increasing the connection capacity, the number of packets lost decreases by being able to route more packets. Average delay times, and maximum delay times, also drastically decrease due to the same reason. It also leads to a higher throughput, especially when conjoined with a higher packet rate (by itself, roughly increases throughput by 5 times when also increasing the capacity by said amount).

- **queue size:** Bigger queue sizes lead to less packet loss due to being able to queue more packets to be sent. They do, meanwhile, not seem to affect average delay times. Queue sizes also dont seem to be affecting average throughput.

c) Data gathering:

```
Result = zeros(8,4);
Conf = zeros(8,4);

Case = zeros(10,4);

C=2;

for f = [10 1]
    for lambda = [1 2]
         for k = 1:10
            [PL , APD , MPD , TT] =
Simulator1(lambda*100,C,f*10000,100000);
            Case(k,1) = PL;
            Case(k,2) = APD;
            Case(k,3) = MPD;
            Case(k,4) = TT;
         end

         if f== 10
            Result(lambda , 1) = mean(Case(:,1));
            Conf(lambda , 1) = 1.645*std(Case(:, 1))/sqrt(10);
         else
            Result(lambda + 2, 1) = mean(Case(:,1));
            Conf(lambda + 2, 1) = 1.645*std(Case(:, 1))/sqrt(10);
         end

         if f== 10
            Result(lambda , 2) = mean(Case(:,2));
            Conf(lambda , 2) = 1.645*std(Case(:, 2))/sqrt(10);
         else
            Result(lambda + 2, 2) = mean(Case(:,2));
            Conf(lambda + 2, 2) = 1.645*std(Case(:, 2))/sqrt(10);
         end

         if f== 10
            Result(lambda , 3) = mean(Case(:,3));
            Conf(lambda , 3) = 1.645*std(Case(:, 3))/sqrt(10);
         else
            Result(lambda + 2, 3) = mean(Case(:,3));
            Conf(lambda + 2, 3) = 1.645*std(Case(:, 3))/sqrt(10);
         end

         if f== 10
            Result(lambda , 4) = mean(Case(:,4));
            Conf(lambda , 4) = 1.645*std(Case(:, 4))/sqrt(10);
         else
            Result(lambda + 2, 4) = mean(Case(:,4));
            Conf(lambda + 2, 4) = 1.645*std(Case(:, 4))/sqrt(10);
         end

        disp(Result);
        disp(Conf);
     end
end
```

```
C=10;

for f = [10 1]
    for lambda = [1 2]
         for k = 1:10
            [PL , APD , MPD , TT] =
Simulator1(lambda*500,C,f*10000,100000);
            Case(k,1) = PL;
            Case(k,2) = APD;
            Case(k,3) = MPD;
            Case(k,4) = TT;
         end

         if f== 10
            Result(4+lambda , 1) = mean(Case(:,1));
            Conf(4+lambda , 1) = 1.645*std(Case(:, 1))/sqrt(10);
         else
            Result(4+lambda + 2, 1) = mean(Case(:,1));
            Conf(4+lambda + 2, 1) = 1.645*std(Case(:, 1))/sqrt(10);
         end

         if f== 10
            Result(4+lambda , 2) = mean(Case(:,2));
            Conf(4+lambda , 2) = 1.645*std(Case(:, 2))/sqrt(10);
         else
            Result(4+lambda + 2, 2) = mean(Case(:,2));
            Conf(4+lambda + 2, 2) = 1.645*std(Case(:, 2))/sqrt(10);
         end

         if f== 10
            Result(4+lambda , 3) = mean(Case(:,3));
            Conf(4+lambda , 3) = 1.645*std(Case(:, 3))/sqrt(10);
         else
            Result(4+lambda + 2, 3) = mean(Case(:,3));
            Conf(4+lambda + 2, 3) = 1.645*std(Case(:, 3))/sqrt(10);
         end

         if f== 10
            Result(4+lambda , 4) = mean(Case(:,4));
            Conf(4+lambda , 4) = 1.645*std(Case(:, 4))/sqrt(10);
         else
            Result(4+lambda + 2, 4) = mean(Case(:,4));
            Conf(4+lambda + 2, 4) = 1.645*std(Case(:, 4))/sqrt(10);
         end

        disp(Result);
        disp(Conf);
     end
 end
```

Case	λ data (pps)	C (Mbps)	f (Bytes)	Packet Loss (%)	Average Delay (msec)	Maximum Delay (msec)	Transm. Through. (Mbps)
A	100	2	100000	0 +/- 0	4.5427 +/- 0.0089	33.1476 +/- 1.3032	0.6672 +/- 0.0012
B	200	2	100000	0 +/- 0	8.1624 +/- 0.0394	66.2101 +/- 4.7449	1.3368 +/- 0.0039
C	100	2	10000	0.0001 +/- 0.0002	4.5365 +/- 0.0111	32.8349 +/- 1.6498	0.6664 +/- 0.0016
D	200	2	10000	0.0566 +/- 0.0066	8.0859 +/- 0.0573	44.5696 +/- 0.2826	1.3328 +/- 0.0016
E	500	10	100000	0 +/- 0	0.9092 +/- 0.0017	6.4303 +/- 0.2974	3.3398 +/- 0.0032
F	1000	10	100000	0 +/- 0	1.6454 +/- 0.0104	13.6000 +/- 0.5910	6.6935 +/- 0.0112
G	500	10	10000	0 +/- 0	0.9091 +/- 0.0014	6.4043 +/- 0.2382	3.3365 +/- 0.0062
H	1000	10	10000	0.0570 +/- 0.0037	1.6218 +/- 0.00572	8.8432 +/- 0.0928	6.6747 +/- 0.0085

Analysis:

- packet loss: Not many conclusions can be taken from these values due to most being 0 (unable to know of decreases/increases that easily). It does seem like overall confidence interval values are decreasing though.

- average delay: By increasing the stopping criteria value, the confidence intervals are decreasing, possibly converging towards 0 as the value keeps being increased.

- maximum delay: While the results from **a)** have the confidence intervals increasing with the packet rates for every value of queue size, the same does not verify for **c)**. For **c)**, when the queue size decreases, the confidence intervals seem to decrease for packet rate increases, rather than increase, such as for higher queue sizes.

- transmission throughput: Just like what happened for average delay values, the increase in the stopping criteria value seems to be influencing the confidence intervals by having them converge towards 0.

d) Data gathering:

```
Result = zeros(8,4);
Conf = zeros(8,4);

Case = zeros(1000,4);

C=2;

for f = [10 1]
    for lambda = [1 2]
         for k = 1:1000
            [PL , APD , MPD , TT] =
Simulator1(lambda*100,C,f*10000,1000);
            Case(k,1) = PL;
            Case(k,2) = APD;
            Case(k,3) = MPD;
            Case(k,4) = TT;
         end

         if f== 10
            Result(lambda , 1) = mean(Case(:,1));
            Conf(lambda , 1) = 1.645*std(Case(:, 1))/sqrt(1000);
         else
            Result(lambda + 2, 1) = mean(Case(:,1));
            Conf(lambda + 2, 1) = 1.645*std(Case(:, 1))/sqrt(1000);
         end

         if f== 10
            Result(lambda , 2) = mean(Case(:,2));
            Conf(lambda , 2) = 1.645*std(Case(:, 2))/sqrt(1000);
         else
            Result(lambda + 2, 2) = mean(Case(:,2));
            Conf(lambda + 2, 2) = 1.645*std(Case(:, 2))/sqrt(1000);
         end

         if f== 10
            Result(lambda , 3) = mean(Case(:,3));
            Conf(lambda , 3) = 1.645*std(Case(:, 3))/sqrt(1000);
         else
            Result(lambda + 2, 3) = mean(Case(:,3));
            Conf(lambda + 2, 3) = 1.645*std(Case(:, 3))/sqrt(1000);
         end

         if f== 10
            Result(lambda , 4) = mean(Case(:,4));
            Conf(lambda , 4) = 1.645*std(Case(:, 4))/sqrt(1000);
         else
            Result(lambda + 2, 4) = mean(Case(:,4));
            Conf(lambda + 2, 4) = 1.645*std(Case(:, 4))/sqrt(1000);
         end

        disp(Result);
        disp(Conf);
     end
end
```

```
C=10;

for f = [10 1]
    for lambda = [1 2]
         for k = 1:1000
            [PL , APD , MPD , TT] =
Simulator1(lambda*500,C,f*10000,1000);
            Case(k,1) = PL;
            Case(k,2) = APD;
            Case(k,3) = MPD;
            Case(k,4) = TT;
         end

         if f== 10
            Result(4+lambda , 1) = mean(Case(:,1));
            Conf(4+lambda , 1) = 1.645*std(Case(:, 1))/sqrt(1000);
         else
            Result(4+lambda + 2, 1) = mean(Case(:,1));
            Conf(4+lambda + 2, 1) = 1.645*std(Case(:, 1))/sqrt(1000);
         end

         if f== 10
            Result(4+lambda , 2) = mean(Case(:,2));
            Conf(4+lambda , 2) = 1.645*std(Case(:, 2))/sqrt(1000);
         else
            Result(4+lambda + 2, 2) = mean(Case(:,2));
            Conf(4+lambda + 2, 2) = 1.645*std(Case(:, 2))/sqrt(1000);
         end

         if f== 10
            Result(4+lambda , 3) = mean(Case(:,3));
            Conf(4+lambda , 3) = 1.645*std(Case(:, 3))/sqrt(1000);
         else
            Result(4+lambda + 2, 3) = mean(Case(:,3));
            Conf(4+lambda + 2, 3) = 1.645*std(Case(:, 3))/sqrt(1000);
         end

         if f== 10
            Result(4+lambda , 4) = mean(Case(:,4));
            Conf(4+lambda , 4) = 1.645*std(Case(:, 4))/sqrt(1000);
         else
            Result(4+lambda + 2, 4) = mean(Case(:,4));
            Conf(4+lambda + 2, 4) = 1.645*std(Case(:, 4))/sqrt(1000);
         end

        disp(Result);
        disp(Conf);
     end
 end
```

Case	λ data (pps)	C (Mbps)	f (Bytes)	Packet Loss (%)	Average Delay (msec)	Maximum Delay (msec)	Transm. Through. (Mbps)
A	100	2	100000	0 +/- 0	4.5509 +/- 0.0089	20.7031 +/- 0.1688	0.6687 +/- 0.0013
B	200	2	100000	0 +/- 0	8.1785 +/- 0.0478	38.5879 +/- 0.4426	1.3368 +/- 0.0026
C	100	2	10000	0 +/- 0	4.5465 +/- 0.0090	20.9580 +/- 0.1807	0.6683 +/- 0.0013
D	200	2	10000	0.0551 +/- 0.0060	8.0629 +/- 0.0413	36.2143 +/- 0.2553	1.3338 +/- 0.0027
E	500	10	100000	0 +/- 0	0.9082 +/- 0.0018	4.1749 +/- 0.0381	3.3401 +/- 0.0066
F	1000	10	100000	0 +/- 0	1.6297 +/- 0.0100	7.6590 +/- 0.0881	6.6695 +/- 0.0137
G	500	10	10000	0 +/- 0	0.9086 +/- 0.0018	4.1323 +/- 0.0351	3.3421 +/- 0.0064
H	1000	10	10000	0.0578 +/- 0.0063	1.6170 +/- 0.0086	7.2386 +/- 0.0521	6.6633 +/- 0.0133

Analysis:

- packet loss: Like stated in **c)**, not many conclusions can be taken from this, asides from the possibility of the values being near stable, with a possible increase in confidence interval values for higher connection capacity values.

- average delay: For lower connection capacity values, average delay confidence intervals seem to have decreased when using larger queue sizes and increased when using smaller queue sizes. For higher connection capacities though, the values seem a lot closer to those obtained in **b)**, despite there still being an increase in the confidence interval for larger connection capacities and queue sizes.

- maximum delay: When comparing with the previous results, these confidence intervals are much smaller, seemingly converging towards 0 and displaying a similar behavior to that explained in **c)** for variation in **a)**.

- transmission throughput: Unlike what was experienced in **c)**, the confidence intervals seem to be increasing in most cases.

e) Data gathering:

For M/M/1:

```
% Every 100 packets, we can expect avg_byte_100_packets bytes.
avg_byte_100_packets = 64*16+1518*22+(65+1517)/2*(100-16-22);

% Given input for each case (A through H).
inputs = [100  ,  2 ;
          200  ,  2 ;
          100  ,  2 ;
          200  ,  2 ;
          500  , 10 ;
          1000 , 10 ;
          500  , 10 ;
          1000 , 10 ];

packets_per_second = zeros(size(inputs, 1), 1);

for i = 1:size(packets_per_second, 1)
   % For each case, calculating the packets that can be sent per second.
   packets_per_second(i,1) = (inputs(i,1) * (inputs(i,2) * 1000000)/8)
...
/ (avg_byte_100_packets * inputs(i,1)/100);
end

avg_packet_delay = zeros(size(packets_per_second, 1), 1);

for j = 1:size(avg_packet_delay, 1)
   % For each case, calculating the average packet delay in M/M/1.
   avg_packet_delay(j,1) = 1000/(packets_per_second(j,1) - inputs(j, 1));
end

% Average packet delay for each case (A through H), in milliseconds.
disp(avg_packet_delay);
```

Case	Theorical Average Packet Delay (msec)
A	5.0116
B	10.0465
C	5.0116
D	10.0465
E	1.0023
F	2.0093
G	1.0023
H	2.0093

For M/G/1:

```
function [result] = func_mg1(lambda, c)

% Calculating the probability of each remaining case.
aux_prob = (1 - 0.16 - 0.22)/(1517 - 65 + 1);

% Calculating E[S] known values.
e_s = (64 * 8 / (c * 1e6)) * 0.16 + (1518 * 8 / (c * 1e6)) * 0.22;

for i = 65:1517
    % Calculating, for each remaining case, their value towards E[S].
    e_s = e_s + (i * 8 / (c * 1e6)) * aux_prob;
end

% Calculating E[S^2] known values.
e_s_2 = (64 * 8 / (c * 1e6))^2 * 0.16 + (1518 * 8 / (c * 1e6))^2 * 0.22;

for i = 65:1517
    % Calculating, for each remaining case, their value towards E[S].
    e_s_2 = e_s_2 + (i * 8 / (c * 1e6))^2 * aux_prob;
end

% packet_rate = [100 ; 200 ; 100 ; 200 ; 500 ; 1000 ; 500 ; 1000];

result = 1000 * ((lambda * e_s_2) / (2 * (1 - lambda * e_s)) + e_s);
```

```
result = [func_mg1(100, 2);
          func_mg1(200, 2);
          func_mg1(500, 10);
          func_mg1(1000, 10)];

disp(result);
```

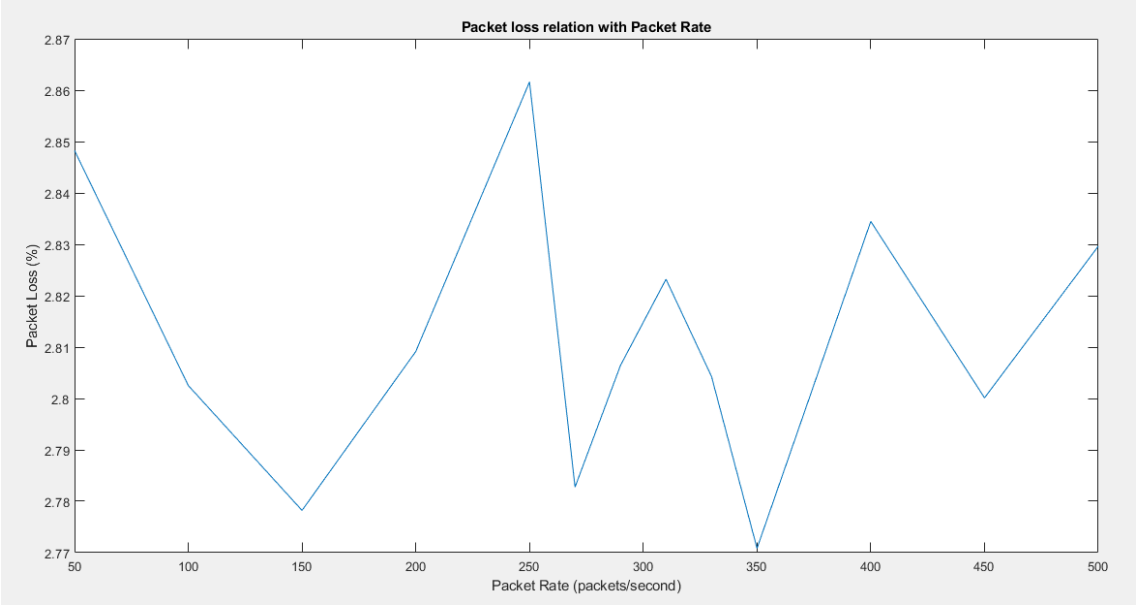
Case	Theorical Average Packet Delay (msec)
A	4.5449
B	8.1755
C	4.5449
D	8.1755
E	0.9090
F	1.6351
G	0.9090
H	1.6351

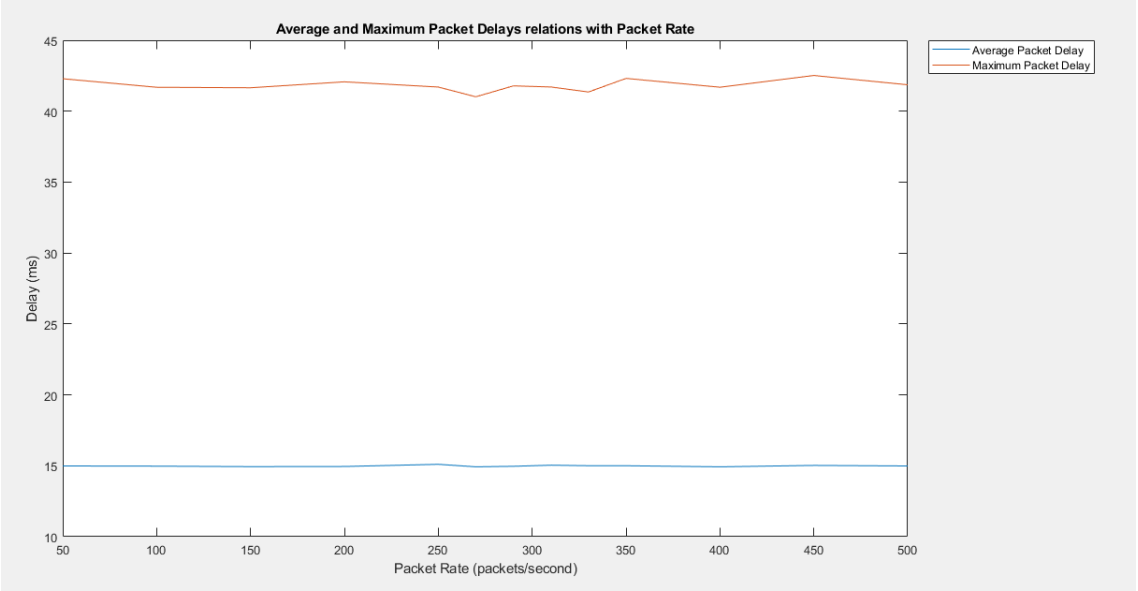
Analysis:

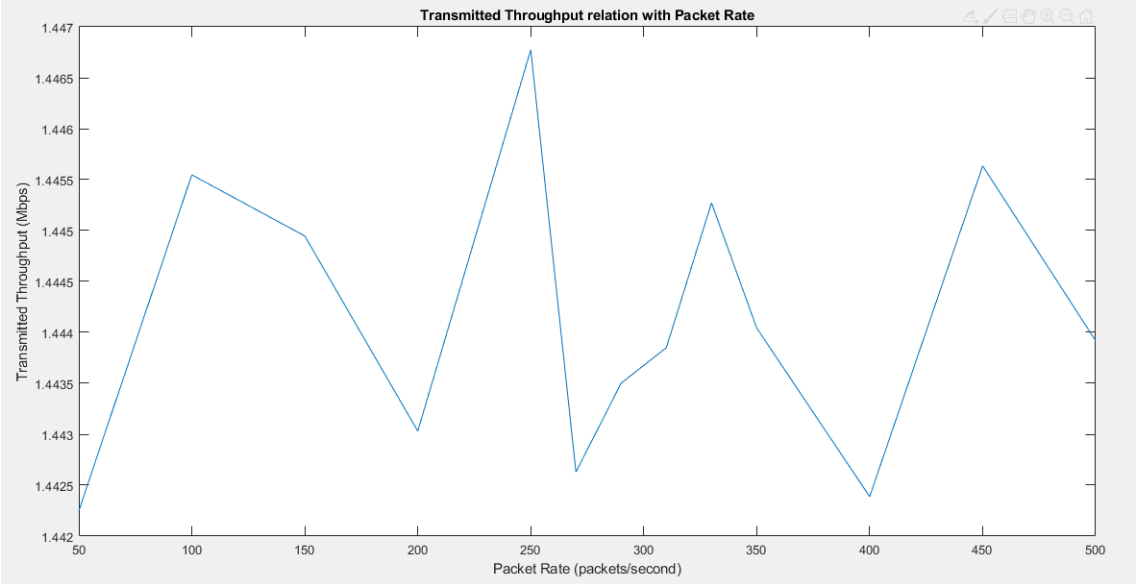
Given the results from both queueing model's theorical calculations, we can conclude that M/G/1 better approximates the result obtained.

f) Data gathering/Plot generation:

```
Case = zeros(10,4);
Result = zeros(13,4);
Conf = zeros(1,4);

pps = [50, 100, 150, 200, 250, 270, 290, 310, 330, 350, 400, 450, 500];
c = 2;
f = 10000;

for i = 1:size(pps, 2)
     for k = 1:10
        [PL , APD , MPD , TT] = Simulator1(pps(1,k), c, f, 100000);
        Case(k,1) = PL;
        Case(k,2) = APD;
        Case(k,3) = MPD;
        Case(k,4) = TT;
     end

     for j = 1:size(Case, 2)
        Result(i, j) = mean(Case(:,j));
        Conf(i , j) = 1.645*std(Case(:,j))/sqrt(10);
     end
end

disp(Result);
disp(Conf);

figure(1)
plot(pps, Result(:,1))

title('Packet loss relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Packet Loss (%)')

figure(2)
plot(pps, Result(:,2), pps, Result(:,3))

title('Average and Maximum Packet Delays relations with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Delay (ms)')
legend({'Average Packet Delay', 'Maximum Packet Delay'} , 'Location',
'bestoutside')

figure(3)
plot(pps, Result(:,4))

title('Transmitted Throughput relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Transmitted Throughput (Mbps)')
```

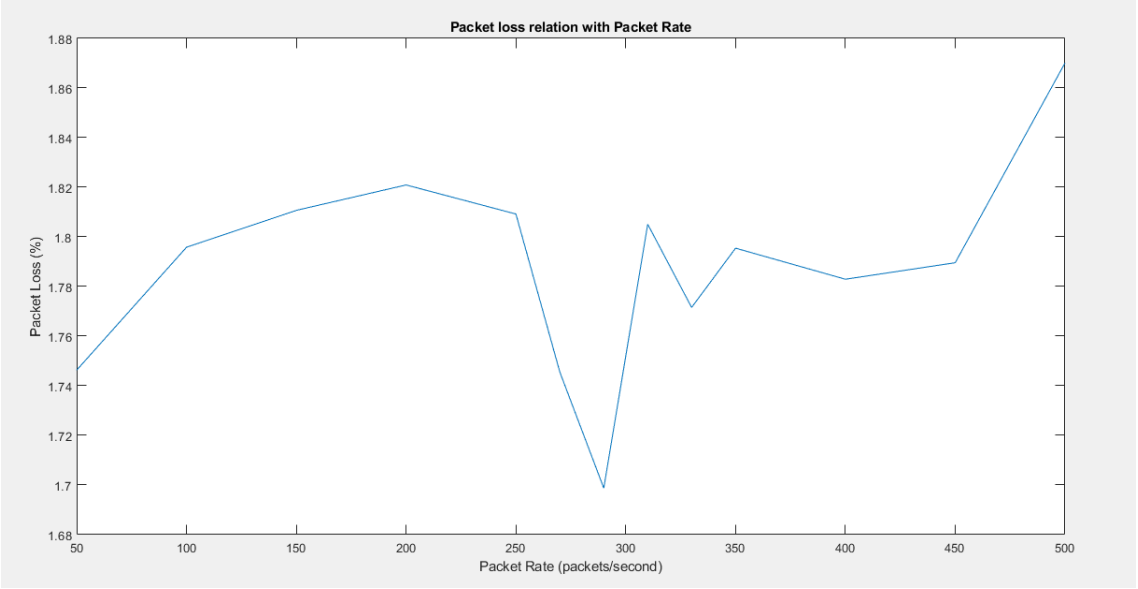

Packet loss relation with Packet Rate
Packet Loss (%)
Packet Rate (packets/second)
2.87
2.86
2.85
2.84
2.83
2.82
2.81
2.8
2.79
2.78
2.77
50
100
150
200
250
300
350
400
450
500

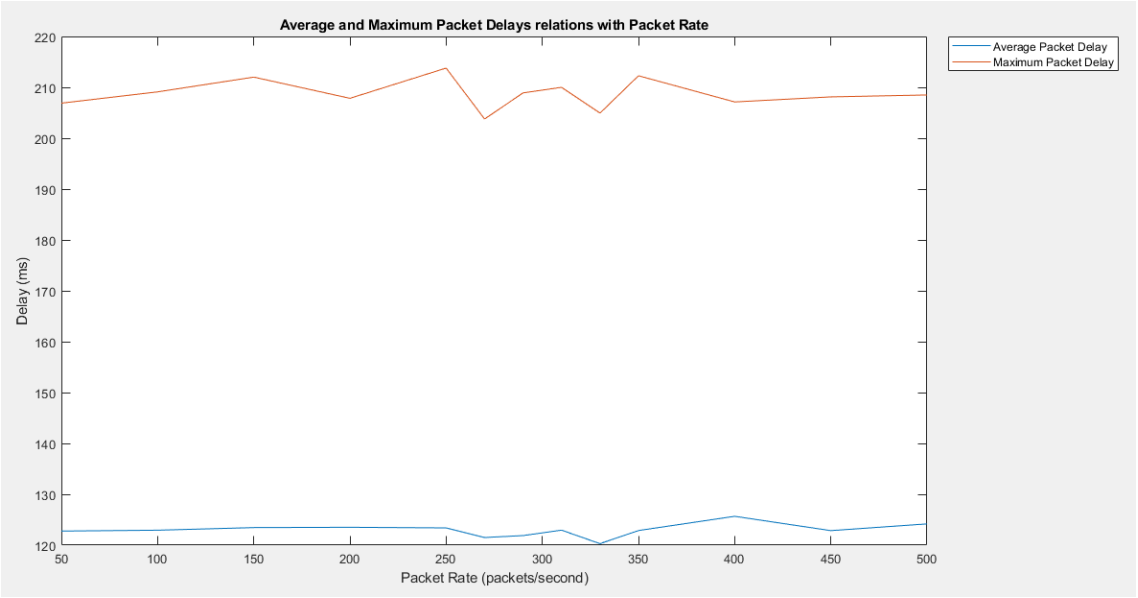

Average and Maximum Packet Delays relations with Packet Rate
Average Packet Delay
Maximum Packet Delay
Delay (ms)
Packet Rate (packets/second)
45
40
35
30
25
20
15
10
50
100
150
200
250
300
350
400
450
500

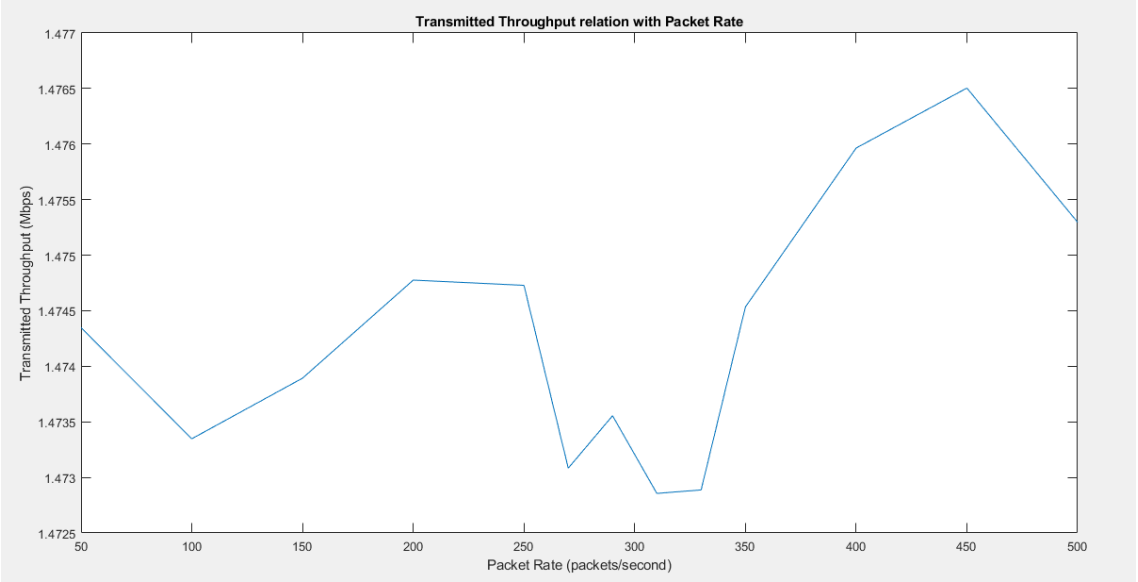

Transmitted Throughput relation with Packet Rate
Transmitted Throughput (Mbps)
Packet Rate (packets/second)
1.447
1.4465
1.446
1.4455
1.445
1.4445
1.444
1.4435
1.443
1.4425
1.442
50
100
150
200
250
300
350
400
450
500

g) Data gathering/Plot generation:

```
Case = zeros(10,4);
Result = zeros(13,4);
Conf = zeros(1,4);

pps = [50, 100, 150, 200, 250, 270, 290, 310, 330, 350, 400, 450, 500];
c = 2;
f = 100000;

for i = 1:size(pps, 2)
     for k = 1:10
        [PL , APD , MPD , TT] = Simulator1(pps(1,k), c, f, 100000);
        Case(k,1) = PL;
        Case(k,2) = APD;
        Case(k,3) = MPD;
        Case(k,4) = TT;
     end

     for j = 1:size(Case, 2)
        Result(i, j) = mean(Case(:,j));
        Conf(i , j) = 1.645*std(Case(:,j))/sqrt(10);
     end
end

disp(Result);
disp(Conf);

figure(1)
plot(pps, Result(:,1))

title('Packet loss relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Packet Loss (%)')

figure(2)
plot(pps, Result(:,2), pps, Result(:,3))

title('Average and Maximum Packet Delays relations with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Delay (ms)')
legend({'Average Packet Delay', 'Maximum Packet Delay'} , 'Location',
'bestoutside')

figure(3)
plot(pps, Result(:,4))

title('Transmitted Throughput relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Transmitted Throughput (Mbps)')
```


Packet loss relation with Packet Rate
Packet Loss (%)
Packet Rate (packets/second)
1.88
1.86
1.84
1.82
1.8
1.78
1.76
1.74
1.72
1.7
1.68
50
100
150
200
250
300
350
400
450
500


Average and Maximum Packet Delays relations with Packet Rate
Average Packet Delay
Maximum Packet Delay
Delay (ms)
Packet Rate (packets/second)
220
210
200
190
180
170
160
150
140
130
120
50
100
150
200
250
300
350
400
450
500


Transmitted Throughput relation with Packet Rate
Transmitted Throughput (Mbps)
Packet Rate (packets/second)
1.477
1.4765
1.476
1.4755
1.475
1.4745
1.474
1.4735
1.473
1.4725
50
100
150
200
250
300
350
400
450
500

Analysis:

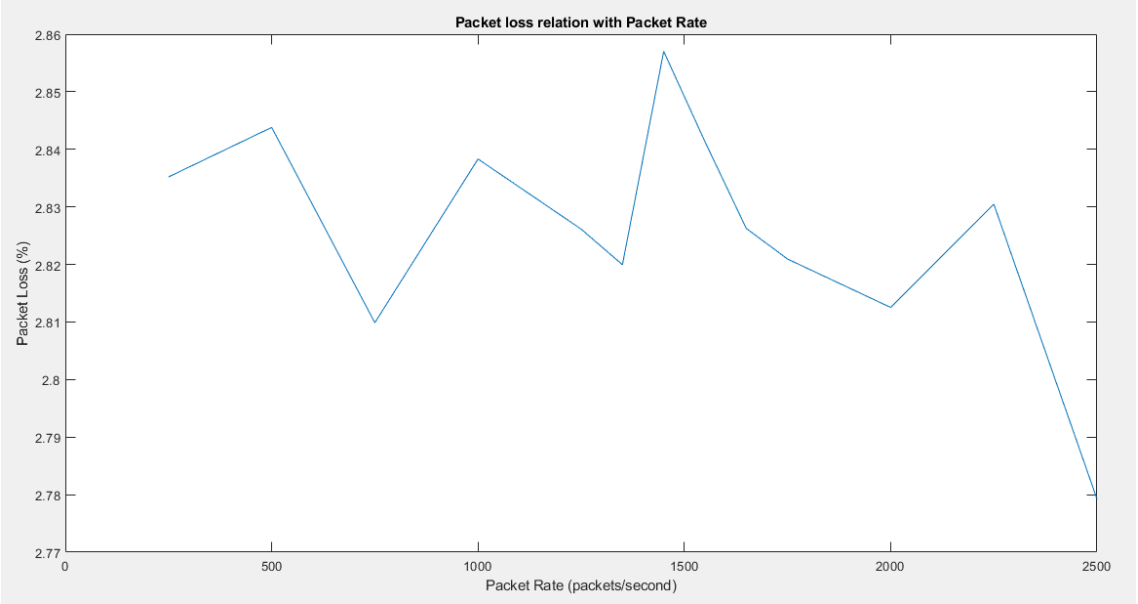
- Packet loss relation with Packet rate: When comparing **f)** and **g)**'s graphs, we noticed that they only shared behavior in certain intervals. Packet loss saw an increase in both graphs in the intervals [150, 200], [290, 310] and [450, 500]; and a decrease in the intervals [250, 270] and [310, 330]. In **f)** we also see a much higher fluctuation of packet loss values throughout the increase of the packet rate, which contrasts the somewhat more stable graph of **g)** which only had a pretty significant dip in the interval [250, 310] and a peak at [450, 500]. The values in **g)** are also smaller.

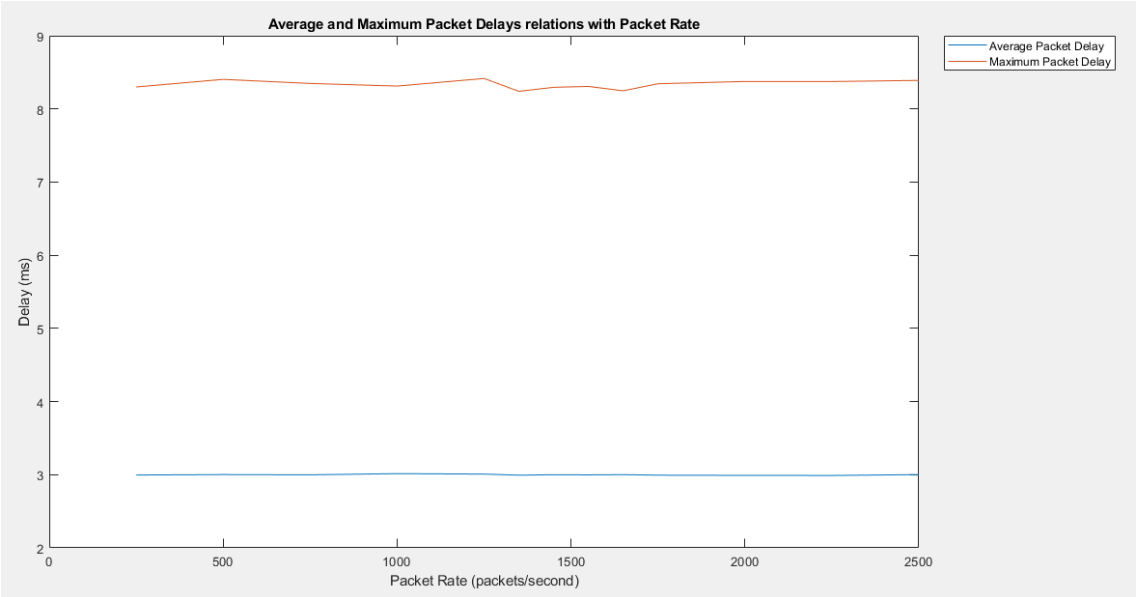
- Average and Maximum Packet Delays relation with Packet Rate: When comparing **f)** and **g)**'s graphs, we noticed that they only shared behavior in certain intervals. Average Packet Delays saw an increase in both graphs in the interval [290, 310]; and a decrease in the intervals [250, 270] and [310, 330]. Maximum Packet Delays saw an increase in both graphs in the intervals [270, 290], [330, 350] and [400, 450]; and a decrease in the intervals [250, 270], [310, 330] and [350, 400]. As for Average Packet Delays, **g)**'s graph shows both higher values and less stability, while **f)**'s is somewhat close to linear. As for Maximum Packet Delays, the same applies.

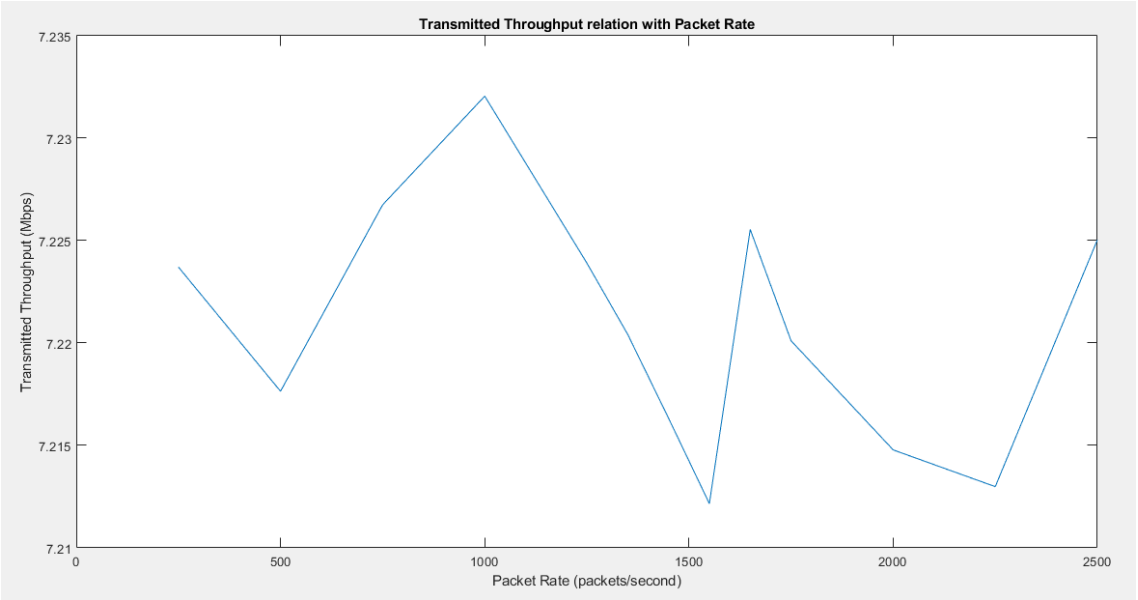
- Transmitted Throughput relation with Packet Rate: When comparing **f)** and **g)**'s graphs, we noticed that they only shared behavior in certain intervals. Transmitted Throughput saw an increase in both graphs in the intervals [270, 290], [310, 330] and [400, 450]; and a decrease in the intervals [250, 270] and [450, 500]. Both graphs seem somewhat unstable and display similar high and low end values, despite it being at different packet rate values, with f) having peaks and dips well distributed throughout the spectrum, while g) seems a bit more concentrated, with a significant dip on the low end of the spectrum, followed by a slight increase, then followed by another two back-to-back dips in the center of the spectrum, to then end with a peak that covers most of the high end values (despite starting to drop off at the actual end of the visible area).

h) Data gathering/Plot generation:

```
Case = zeros(10,4);
Result = zeros(13,4);
Conf = zeros(1,4);

pps = [250, 500, 750, 1000, 1250, 1350, 1450, 1550, 1650, 1750, 2000,
2250 ,2500];
c = 10;
f = 10000;

for i = 1:size(pps, 2)
     for k = 1:10
        [PL , APD , MPD , TT] = Simulator1(pps(1,k), c, f, 100000);
        Case(k,1) = PL;
        Case(k,2) = APD;
        Case(k,3) = MPD;
        Case(k,4) = TT;
     end

     for j = 1:size(Case, 2)
        Result(i, j) = mean(Case(:,j));
        Conf(i , j) = 1.645*std(Case(:,j))/sqrt(10);
     end
end

disp(Result);
disp(Conf);

figure(1)
plot(pps, Result(:,1))

title('Packet loss relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Packet Loss (%)')

figure(2)
plot(pps, Result(:,2), pps, Result(:,3))

title('Average and Maximum Packet Delays relations with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Delay (ms)')
legend({'Average Packet Delay', 'Maximum Packet Delay'} , 'Location',
'bestoutside')

figure(3)
plot(pps, Result(:,4))

title('Transmitted Throughput relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Transmitted Throughput (Mbps)')
```


Packet loss relation with Packet Rate
Packet Loss (%)
Packet Rate (packets/second)
2.86
2.85
2.84
2.83
2.82
2.81
2.8
2.79
2.78
2.77
0
500
1000
1500
2000
2500


Average and Maximum Packet Delays relations with Packet Rate
Average Packet Delay
Maximum Packet Delay
Delay (ms)
Packet Rate (packets/second)
9
8
7
6
5
4
3
2
0
500
1000
1500
2000
2500


Transmitted Throughput relation with Packet Rate
Transmitted Throughput (Mbps)
Packet Rate (packets/second)
7.235
7.23
7.225
7.22
7.215
7.21
0
500
1000
1500
2000
2500

Analysis:

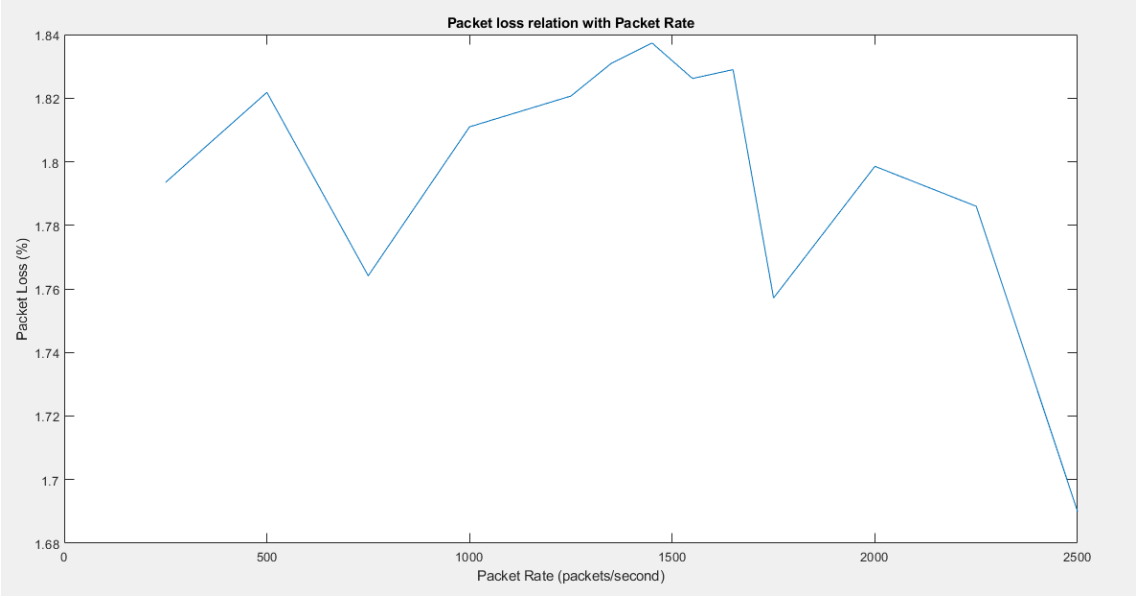
- Packet loss relation with Packet rate: When comparing **f)**, **g)** and **h)**, we noticed that in the interval [50, 500], both **g)** and **h)** end with a higher packet loss value, while **f)** ends with a lower one. The graph that describes **h)** is also very unstable, with a high peak at 1450 packets per second and a big dip at the end of it.

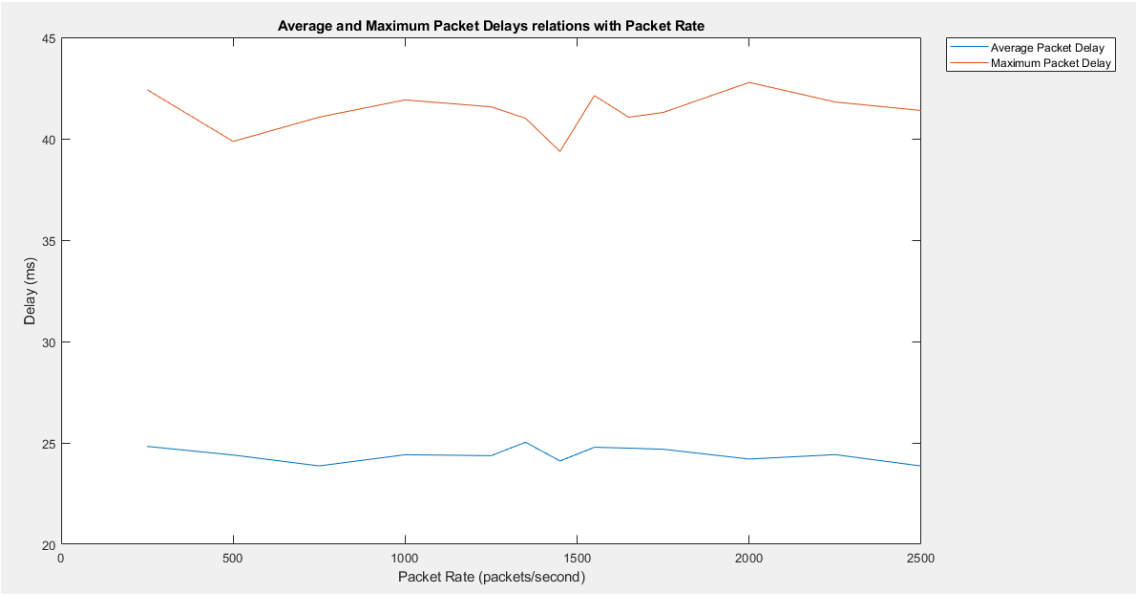
- Average and Maximum Packet Delays relation with Packet Rate: When comparing **f)**, **g)** and **h)**, we instantly noticed the vastly inferior values for both Average and Maximum Packet Delays. Both Average and Maximum Packet Delays also seem to have a somewhat linear representation, only with the Maximum being a bit more irregular.

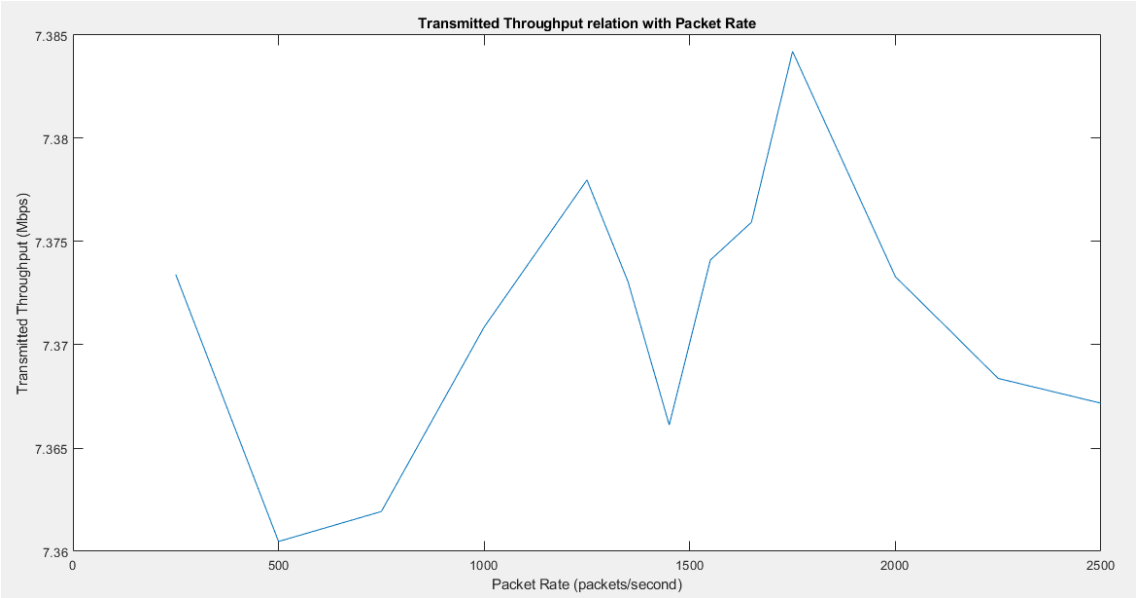
- Transmitted Throughput relation with Packet Rate: When comparing **f)**, **g)** and **h)**, we noticed that the transmitted throughput values are vastly superior in **h)**, when comparing to **f)** and **g)**. It is also worth noting that in the interval [50, 500] both **f)** and **g)** increased their throughput while it decreased in **h)**.

i) Data gathering/Plot generation:

```
Case = zeros(10,4);
Result = zeros(13,4);
Conf = zeros(1,4);

pps = [250, 500, 750, 1000, 1250, 1350, 1450, 1550, 1650, 1750, 2000,
2250 ,2500];
c = 10;
f = 100000;

for i = 1:size(pps, 2)
     for k = 1:10
        [PL , APD , MPD , TT] = Simulator1(pps(1,k), c, f, 100000);
        Case(k,1) = PL;
        Case(k,2) = APD;
        Case(k,3) = MPD;
        Case(k,4) = TT;
     end

     for j = 1:size(Case, 2)
        Result(i, j) = mean(Case(:,j));
        Conf(i , j) = 1.645*std(Case(:,j))/sqrt(10);
     end
end

disp(Result);
disp(Conf);

figure(1)
plot(pps, Result(:,1))

title('Packet loss relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Packet Loss (%)')

figure(2)
plot(pps, Result(:,2), pps, Result(:,3))

title('Average and Maximum Packet Delays relations with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Delay (ms)')
legend({'Average Packet Delay', 'Maximum Packet Delay'} , 'Location',
'bestoutside')

figure(3)
plot(pps, Result(:,4))

title('Transmitted Throughput relation with Packet Rate')
xlabel('Packet Rate (packets/second)')
ylabel('Transmitted Throughput (Mbps)')
```


Packet loss relation with Packet Rate
Packet Loss (%)
Packet Rate (packets/second)
1.84
1.82
1.8
1.78
1.76
1.74
1.72
1.7
1.68
0
500
1000
1500
2000
2500


Average and Maximum Packet Delays relations with Packet Rate
Delay (ms)
Packet Rate (packets/second)
Average Packet Delay
Maximum Packet Delay
45
40
35
30
25
20
0
500
1000
1500
2000
2500


Transmitted Throughput relation with Packet Rate
Transmitted Throughput (Mbps)
Packet Rate (packets/second)
7.385
7.38
7.375
7.37
7.365
7.36
0
500
1000
1500
2000
2500

Analysis:

- Packet loss relation with Packet rate: When comparing **f)**, **g)**, **h)** and **i)**, we noticed that in the interval [50, 500], both **g)** and **h)** end with a higher packet loss value, while **f)** and **i)** end with a lower one. The graphs that describe **h)** and **i)** are also very unstable, with a high peak at 1450 packets per second and a big dip at the end of it. Both graphs are very similar in terms of form, with **i)**'s displaying lower values in relation to **h)**, just as **g)** does to **f)**.

- Average and Maximum Packet Delays relation with Packet Rate: : When comparing **f)**, **g)**, **h)** and **i)**, we noticed that just like the delay values increase between **f)** and **g)**, they also do from **h)** to **i)**, as well as share similarities when it comes to the graph forms.

- Transmitted Throughput relation with Packet Rate: When comparing **f)**, **g)**, **h)** and **i)**, we noticed that **h)** and **i)** have both very similar forms and values. There were only 4 intervals where h) and i) did not behave in the same way, those being [1000, 1250], [1450, 1550], [1650, 1750] and [2250, 2500].

Simulator 2:

```
function [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim2(lambda,C,f,P,n)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
% OUTPUT PARAMETERS:
%  PL   - packet loss (%)
%  APD  - average packet delay (milliseconds)
%  MPD  - maximum packet delay (milliseconds)
%  TT   - transmitted throughput (Mbps)

%Events:
ARRIVAL= 0;       % Arrival of a packet
DEPARTURE= 1;     % Departure of a packet

DATA= 1;
VOIP= 2;

%State variables:
State = 0;          % 0 - connection free; 1 - connection bysy
QueueOccupation= 0; % Occupation of the queue (in Bytes)
Queue= [];          % Size and arriving time instant of each packet in
the queue

%Statistical Counters:
TotalPackets= 0;       % No. of packets arrived to the system
LostPackets= 0;        % No. of packets dropped due to buffer overflow
TransmittedPackets= 0; % No. of transmitted packets
TransmittedBytes= 0;   % Sum of the Bytes of transmitted packets
Delays= 0;             % Sum of the delays of transmitted packets
MaxDelay= 0;           % Maximum delay among all transmitted packets
TotalPacketsVoip= 0;
LostPacketsVoip= 0;
TransmittedPacketsVoip= 0;
DelaysVoip= 0;
MaxDelayVoip= 0;

%Auxiliary variables:
% Initializing the simulation clock:
Clock= 0;

% Initializing the List of Events with the first ARRIVAL:
EventList = [ARRIVAL , Clock + exprnd(1/lambda) , GeneratePacketSize() ,
0, DATA];
```

```
for i=1:n
    EventList= [EventList; ARRIVAL , Clock +  0.02*rand(),
GeneratePacketSizeVoip() , 0, VOIP];
end

%Similation loop:
while TransmittedPackets<P                % Stopping criterium
    EventList= sortrows(EventList,2);     % Order EentList by time
    Event= EventList(1,1);                % Get first event and
    Clock= EventList(1,2);                %   and
    PacketSize= EventList(1,3);           %   associated
    ArrivalInstant= EventList(1,4);       %   parameters.
    Type = EventList(1,5);
    EventList(1,:)= [];                   % Eliminate first event
    switch Event
        case ARRIVAL                      % If first event is an ARRIVAL
            if Type == DATA
                TotalPackets= TotalPackets+1;
                EventList = [EventList; ARRIVAL , Clock +
exprnd(1/lambda) , GeneratePacketSize() , 0, DATA];
            else
                TotalPacketsVoip= TotalPacketsVoip +1;
                EventList= [EventList; ARRIVAL , Clock +
0.001*randi([16,24]), GeneratePacketSizeVoip() , 0, VOIP];
            end

            if State==0
                State= 1;
                EventList = [EventList; DEPARTURE , Clock +
8*PacketSize/(C*10^6) , PacketSize , Clock, Type];
            else
                if QueueOccupation + PacketSize <= f
                    Queue= [Queue;PacketSize , Clock, Type];
                    QueueOccupation= QueueOccupation + PacketSize;
                else
                    if Type == DATA
                        LostPackets= LostPackets + 1;
                    else
                        LostPacketsVoip= LostPacketsVoip +1;
                    end
                end
            end
        case DEPARTURE                    % If first event is a
DEPARTURE
            TransmittedBytes= TransmittedBytes + PacketSize;

            if Type == DATA
                Delays= Delays + (Clock - ArrivalInstant);
            else
                DelaysVoip= DelaysVoip + (Clock - ArrivalInstant);
            end
```

```
            if Clock - ArrivalInstant > MaxDelay
                if Type == DATA
                    MaxDelay= Clock - ArrivalInstant;
                end
            end
            if Clock - ArrivalInstant > MaxDelayVoip
                if Type == VOIP
                    MaxDelayVoip= Clock - ArrivalInstant;
                end
            end

            if Type == DATA
                TransmittedPackets= TransmittedPackets + 1;
            else
                TransmittedPacketsVoip= TransmittedPacketsVoip + 1;
            end

            if QueueOccupation > 0
                EventList = [EventList; DEPARTURE , Clock +
8*Queue(1,1)/(C*10^6) , Queue(1,1) , Queue(1,2), Queue(1,3)];
                QueueOccupation= QueueOccupation - Queue(1,1);
                Queue(1,:)= [];
            else
                State= 0;
            end
    end
end

%Performance parameters determination:
PL= 100*LostPackets/TotalPackets      % in %
PLvoip= 100*LostPacketsVoip/TotalPacketsVoip
APD= 1000*Delays/TransmittedPackets   % in milliseconds
APDvoip= 1000*DelaysVoip/TransmittedPacketsVoip
MPD= 1000*MaxDelay                    % in milliseconds
MPDvoip= 1000*MaxDelayVoip
TT= 10^(-6)*TransmittedBytes*8/Clock  % in Mbps

end

function out= GeneratePacketSize()
    aux= rand();
    if aux <= 0.16
        out= 64;
    elseif aux >= 0.78
        out= 1518;
    else
        out = randi([65 1517]);
    end
end

function out= GeneratePacketSizeVoip()
    out = randi([110 130]);
end
```

j) Data gathering:

```
Result = zeros(12,4);
Conf = zeros(12,4);

ResultVoip = zeros(12,4);
ConfVoip = zeros(12,4);

Case = zeros(10,7);

lambda = 2;

for f = [10 1]
    for n = [5 10 15]

         for k = 1:10
            [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim2(lambda*100,2,f*10000,100000,n);
            Case(k,1) = PL;
            Case(k,2) = PLvoip;
            Case(k,3) = APD;
            Case(k,4) = APDvoip;
            Case(k,5) = MPD;
            Case(k,6) = MPDvoip;
            Case(k,7) = TT
         end

         %PL

         if f== 10
            Result(n/5 + 3*0 + 7*0, 1) = mean(Case(:,1));
            ResultVoip(n/5 + 3*0 + 7*0 , 1) = mean(Case(:,2));
            Conf(n/5 + 3*0 + 7*0 , 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/5 + 3*0 + 7*0 , 1) = 1.645*std(Case(:,
2))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 1) = mean(Case(:,1));
            ResultVoip(n/5 + 3*1 + 7*0, 1) = mean(Case(:,2));
            Conf(n/5 + 3*1 + 7*0, 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/5 + 3*1 + 7*0, 1) = 1.645*std(Case(:,
2))/sqrt(10);
         end
```

```
        %ADD

         if f== 10
            Result(n/5 + 3*0 + 7*0, 2) = mean(Case(:,3));
            ResultVoip(n/5 + 3*0 + 7*0 , 2) = mean(Case(:,4));
            Conf(n/5 + 3*0 + 7*0 , 2) = 1.645*std(Case(:,
3))/sqrt(10);
            ConfVoip(n/5 + 3*0 + 7*0 , 2) = 1.645*std(Case(:,
4))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 2) = mean(Case(:,3));
            ResultVoip(n/5 + 3*1 + 7*0, 2) = mean(Case(:,4));
            Conf(n/5 + 3*1 + 7*0, 2) = 1.645*std(Case(:, 3))/sqrt(10);
            ConfVoip(n/5 + 3*1 + 7*0, 2) = 1.645*std(Case(:,
4))/sqrt(10);
         end

         %MDA

         if f== 10
            Result(n/5 + 3*0 + 7*0, 3) = mean(Case(:,5));
            ResultVoip(n/5 + 3*0 + 7*0 , 3) = mean(Case(:,6));
            Conf(n/5 + 3*0 + 7*0 , 3) = 1.645*std(Case(:,
5))/sqrt(10);
            ConfVoip(n/5 + 3*0 + 7*0 , 3) = 1.645*std(Case(:,
6))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 3) = mean(Case(:,5));
            ResultVoip(n/5 + 3*1 + 7*0, 3) = mean(Case(:,6));
            Conf(n/5 + 3*1 + 7*0, 3) = 1.645*std(Case(:, 5))/sqrt(10);
            ConfVoip(n/5 + 3*1 + 7*0, 3) = 1.645*std(Case(:,
6))/sqrt(10);
         end

         %TT

         if f== 10
            Result(n/5 + 3*0 + 7*0, 4) = mean(Case(:,7));
            Conf(n/5 + 3*0 + 7*0 , 4) = 1.645*std(Case(:,
7))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 4) = mean(Case(:,7));
            Conf(n/5 + 3*1 + 7*0, 4) = 1.645*std(Case(:, 7))/sqrt(10);
         end

        Result
        Conf
        ResultVoip
        ConfVoip
     end
end
```

```
lambda = 10 ;

for f = [10 1]
    for n = [25 50 75]

         for k = 1:10
            [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim2(lambda*100,10,f*10000,100000,n);
            Case(k,1) = PL;
            Case(k,2) = PLvoip;
            Case(k,3) = APD;
            Case(k,4) = APDvoip;
            Case(k,5) = MPD;
            Case(k,6) = MPDvoip;
            Case(k,7) = TT;
         end

         %PL

         if f== 10
            Result(n/25 + 3*0 + 7*1, 1) = mean(Case(:,1));
            ResultVoip(n/25 + 3*1 + 7*0 , 1) = mean(Case(:,2));
            Conf(n/25 + 3*0 + 7*1 , 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/25 + 3*1 + 7*0 , 1) = 1.645*std(Case(:,
2))/sqrt(10);
         else
            Result(n/25 + 3*1 + 7*1, 1) = mean(Case(:,1));
            ResultVoip(n/25 + 3*1 + 7*1, 1) = mean(Case(:,2));
            Conf(n/25 + 3*1 + 7*1, 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/25 + 3*1 + 7*1, 1) = 1.645*std(Case(:,
2))/sqrt(10);
         end

        %ADD

         if f== 10
            Result(n/25 + 3*0 + 7*1, 2) = mean(Case(:,3));
            ResultVoip(n/25 + 3*0 + 7*1 , 2) = mean(Case(:,4));
            Conf(n/25 + 3*0 + 7*1 , 2) = 1.645*std(Case(:, 3))/sqrt(10);
            ConfVoip(n/25 + 3*0 + 7*1 , 2) = 1.645*std(Case(:,
4))/sqrt(10);
         else
            Result(n/25 + 3*1 + 7*1, 2) = mean(Case(:,3));
            ResultVoip(n/25 + 3*1 + 7*1, 2) = mean(Case(:,4));
            Conf(n/25 + 3*1 + 7*1, 2) = 1.645*std(Case(:, 3))/sqrt(10);
            ConfVoip(n/25 + 3*1 + 7*1, 2) = 1.645*std(Case(:,
4))/sqrt(10);
         end
```

```
          %MDA

          if f== 10
             Result(n/25 + 3*0 + 7*1, 3) = mean(Case(:,5));
             ResultVoip(n/25 + 3*0 + 7*1 , 3) = mean(Case(:,6));
             Conf(n/25 + 3*0 + 7*1 , 3) = 1.645*std(Case(:,
5))/sqrt(10);
             ConfVoip(n/25 + 3*0 + 7*1 , 3) = 1.645*std(Case(:,
6))/sqrt(10);
          else
             Result(n/25 + 3*1 + 7*1, 3) = mean(Case(:,5));
             ResultVoip(n/25 + 3*1 + 7*1, 3) = mean(Case(:,6));
             Conf(n/25 + 3*1 + 7*1, 3) = 1.645*std(Case(:,
5))/sqrt(10);
             ConfVoip(n/25 + 3*1 + 7*1, 3) = 1.645*std(Case(:,
6))/sqrt(10);
          end

          %TT

          if f== 10
             Result(n/25 + 3*0 + 7*1, 4) = mean(Case(:,7));
             Conf(n/25 + 3*0 + 7*1 , 4) = 1.645*std(Case(:,
7))/sqrt(10);
          else
             Result(n/25 + 3*1 + 7*1, 4) = mean(Case(:,7));
             Conf(n/25 + 3*1 + 7*1, 4) = 1.645*std(Case(:,
7))/sqrt(10);
          end

         Result
         Conf
         ResultVoip
         ConfVoip
      end
end
```

Case	λ data (pps)	C (Mbps)	f (Bytes)	n	Packet Loss Data / VoIP (%)	Avg. Delay Data / VoIP (msec)	Max. Delay data / VoIP (msec)	Transm. Through. (Mbps)
A	200	2	100000	5	0 +/- 0 / 0 +/- 0	10.9385 +/- 0.0927 / 7.9137 +/- 0.0801	86.4779 +/- 3.4647 / 84.3790 +/- 3.3209	1.5721 +/- 0.0034
B	200	2	100000	10	0 +/- 0 / 0 +/- 0	20.2543 +/- 0.3625 / 17.2314 +/- 0.3657	144.1893 +/- 6.0430 / 143.7091 +/- 5.9528	1.8113 +/- 0.0032
C	200	2	100000	15	2.5693 +/- 0.1030 / 0.2011 +/- 0.0070	339.3675 +/- 4.2604 / 337.8070 +/- 4.2558	405.5139 +/- 0.1130 / 405.1651 +/- 0.1846	1.9997 +/- 0.0002
D	200	2	10000	5	0.2547 +/- 0.0150 / 0.0187 +/- 0.0030	10.3513 +/- 0.0585 / 7.4227 +/- 0.0589	45.3936 +/- 0.1697 / 44.3054 +/- 0.2363	1.5693 +/- 0.0034
E	200	2	10000	10	1.1940 +/- 0.0421 / 0.0978 +/- 0.0058	14.9640 +/- 0.0851 / 12.2125 +/- 0.0819	45.4802 +/- 0.1361 / 44.9990 +/- 0.1829	1.7873 +/- 0.0021
F	200	2	10000	15	5.2004 +/- 0.0694 / 0.4377 +/- 0.0091	23.2083 +/- 0.1300 / 21.0858 +/- 0.1375	45.8014 +/- 0.0737 / 45.4616 +/- 0.1373	1.9470 +/- 0.0010

Case	λ data (pps)	C (Mbps)	f (Bytes)	n	Packet Loss Data / VoIP (%)	Avg. Delay Data / VoIP (msec)	Max. Delay data / VoIP (msec)	Transm. Through. (Mbps)
G	1000	10	100000	25	0 +/- 0 / 0 +/- 0	2.1786 +/- 0.0141 / 1.5900 +/- 0.0133	18.8575 +/- 1.6507 / 18.3416 +/- 1.6405	7.8738 +/- 0.0104
H	1000	10	100000	50	0 +/- 0 / 0 +/- 0	4.1238 +/- 0.0870 / 3.5223 +/- 0.0851	29.7285 +/- 2.2927 / 29.5308 +/- 2.2452	9.0690 +/- 0.0141
I	1000	10	100000	75	2.5915 +/- 0.2107 / 0 +/- 0	67.7451 +/- 1.1398 / 67.4281 +/- 1.1437	81.1453 +/- 0.0196 / 81.0540 +/- 0.0305	9.9989 +/- 0.0005
J	1000	10	10000	25	0.2531 +/- 0.0155 / 0.0214 +/- 0.0030	2.0650 +/- 0.0136 / 1.4901 +/- 0.0128	9.0575 +/- 0.0246 / 8.8844 +/- 0.0784	7.8401 +/- 0.0140
K	1000	10	10000	50	1.2242 +/- 0.0441 / 0.1101 +/- 0.0054	3.0050 +/- 0.0176 / 2.4710 +/- 0.0176	9.1190 +/- 0.0284 / 9.0611 +/- 0.0519	8.9477 +/- 0.0085
L	1000	10	10000	75	5.1861 +/- 0.0808 / 0.4752 +/- 0.0112	4.6270 +/- 0.0237 / 4.2101 +/- 0.0251	9.1426 +/- 0.0195 / 9.0923 +/- 0.0202	9.7275 +/- 0.0062

k) Analysis:

When analyzing the data pertaining to packet loss, we noticed that both the packet rate and connection capacity do not seem to affect the values when incremented at the same rate. Decreasing the queue size though seems to cause a higher packet loss to happen. Packet loss also increases with VoIP flow increases. The VoIP flows also displayed much smaller values than the data flow, as well as smaller confidence intervals. The previously enunciated behavior was similar in both data and VoIP flows.

When analyzing the data pertaining to average packet delay, we noticed that when both the packet rate and the connection capacity increased, the average packet delay decreased. For the same packet rate and connection capacity values, as the number of packet flows increases the average packet delay increases as well. This increase is less impactful when the queue size is decreased. Both the values obtained and the observed behaviors were similar for both data flows when looking at average packet delays.

When analyzing the data pertaining to the maximum packet delay, we noticed that, similarly to the average packet delay, the maximum packet delay seems to decrease as the values of both the packet rate and the connection capacity increase. The conclusions taken regarding queue size and VoIP flows also stand. Once again, values obtained and behavior were similar in both data flows.

When analyzing the data pertaining transmitted throughput, we noticed that when increasing both the packet rate and the connection capacity, both the transmitted throughput and its confidence interval seem to increase. When the queue size decreases though, transmitted throughput also decreases, maintaining similar incrementations for the same packet rate and connection capacity values as VoIP flows increase. Meanwhile, while increasing VoIP flows, the transmitted throughput also seems to increase, bolstering higher increments for larger packet rate and connection capacity values.

Simulator 3:

```
function [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim3(lambda,C,f,P,n)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
% OUTPUT PARAMETERS:
%  PL   - packet loss (%)
%  APD  - average packet delay (milliseconds)
%  MPD  - maximum packet delay (milliseconds)
%  TT   - transmitted throughput (Mbps)

%Events:
ARRIVAL= 0;       % Arrival of a packet
DEPARTURE= 1;     % Departure of a packet

DATA= 1;
VOIP= 0;

%State variables:
State = 0;          % 0 - connection free; 1 - connection bysy
QueueOccupation= 0; % Occupation of the queue (in Bytes)
Queue= [];          % Size and arriving time instant of each packet in
the queue

%Statistical Counters:
TotalPackets= 0;       % No. of packets arrived to the system
LostPackets= 0;        % No. of packets dropped due to buffer overflow
TransmittedPackets= 0; % No. of transmitted packets
TransmittedBytes= 0;   % Sum of the Bytes of transmitted packets
Delays= 0;             % Sum of the delays of transmitted packets
MaxDelay= 0;           % Maximum delay among all transmitted packets
TotalPacketsVoip= 0;
LostPacketsVoip= 0;
TransmittedPacketsVoip= 0;
DelaysVoip= 0;
MaxDelayVoip= 0;

%Auxiliary variables:
% Initializing the simulation clock:
Clock= 0;

% Initializing the List of Events with the first ARRIVAL:
EventList = [ARRIVAL , Clock + exprnd(1/lambda) , GeneratePacketSize() ,
0, DATA];
```

```
for i=1:n
    EventList= [EventList; ARRIVAL , Clock +  0.02*rand(),
GeneratePacketSizeVoip() , 0, VOIP];
end

%Similation loop:
while TransmittedPackets<P                % Stopping criterium
    EventList= sortrows(EventList,2);     % Order EentList by time
    Event= EventList(1,1);                % Get first event and
    Clock= EventList(1,2);                %   and
    PacketSize= EventList(1,3);           %   associated
    ArrivalInstant= EventList(1,4);       %   parameters.
    Type = EventList(1,5);
    EventList(1,:)= [];                   % Eliminate first event
    switch Event
        case ARRIVAL                      % If first event is an ARRIVAL
            if Type == DATA
                TotalPackets= TotalPackets+1;
                   EventList = [EventList; ARRIVAL , Clock +
exprnd(1/lambda) , GeneratePacketSize() , 0, DATA];
            else
                TotalPacketsVoip= TotalPacketsVoip +1;
                   EventList= [EventList; ARRIVAL , Clock +
0.001*randi([16,24]), GeneratePacketSizeVoip() , 0, VOIP];
            end

            if State==0
                State= 1;
                EventList = [EventList; DEPARTURE , Clock +
8*PacketSize/(C*10^6) , PacketSize , Clock, Type];
            else
                if QueueOccupation + PacketSize <= f
                    Queue= [Queue;PacketSize , Clock, Type];
                    Queue= sortrows(Queue,3);

                    QueueOccupation= QueueOccupation + PacketSize;
                else
                    if Type == DATA
                        LostPackets= LostPackets + 1;
                    else
                        LostPacketsVoip= LostPacketsVoip +1;
                    end
                end
            end
        case DEPARTURE                    % If first event is a
DEPARTURE
            TransmittedBytes= TransmittedBytes + PacketSize;

            if Type == DATA
                Delays= Delays + (Clock - ArrivalInstant);
            else
                DelaysVoip= DelaysVoip + (Clock - ArrivalInstant);
            end
```

```
            if Clock - ArrivalInstant > MaxDelay
                if Type == DATA
                    MaxDelay= Clock - ArrivalInstant;
                end
            end
            if Clock - ArrivalInstant > MaxDelayVoip
                if Type == VOIP
                    MaxDelayVoip= Clock - ArrivalInstant;
                end
            end

            if Type == DATA
                TransmittedPackets= TransmittedPackets + 1;
            else
                TransmittedPacketsVoip= TransmittedPacketsVoip + 1;
            end

            if QueueOccupation > 0
                EventList = [EventList; DEPARTURE , Clock +
8*Queue(1,1)/(C*10^6) , Queue(1,1) , Queue(1,2), Queue(1,3)];
                QueueOccupation= QueueOccupation - Queue(1,1);
                Queue(1,:)= [];
            else
                State= 0;
            end
    end
end

%Performance parameters determination:
PL= 100*LostPackets/TotalPackets      % in %
PLvoip= 100*LostPacketsVoip/TotalPacketsVoip
APD= 1000*Delays/TransmittedPackets   % in milliseconds
APDvoip= 1000*DelaysVoip/TransmittedPacketsVoip
MPD= 1000*MaxDelay                    % in milliseconds
MPDvoip= 1000*MaxDelayVoip
TT= 10^(-6)*TransmittedBytes*8/Clock  % in Mbps

end

function out= GeneratePacketSize()
    aux= rand();
    if aux <= 0.16
        out= 64;
    elseif aux >= 0.78
        out= 1518;
    else
        out = randi([65 1517]);
    end
end

function out= GeneratePacketSizeVoip()
    out = randi([110 130]);
end
```

I) Data gathering:

```
Result = zeros(12,4);
Conf = zeros(12,4);

ResultVoip = zeros(12,4);
ConfVoip = zeros(12,4);

Case = zeros(10,7);

lambda = 2;

for f = [10 1]
    for n = [5 10 15]
         for k = 1:10
            [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim3(lambda*100,2,f*10000,100000,n);
            Case(k,1) = PL;
            Case(k,2) = PLvoip;
            Case(k,3) = APD;
            Case(k,4) = APDvoip;
            Case(k,5) = MPD;
            Case(k,6) = MPDvoip;
            Case(k,7) = TT;
         end

         %PL

         if f== 10
            Result(n/5 + 3*0 + 7*0, 1) = mean(Case(:,1));
            ResultVoip(n/5 + 3*0 + 7*0 , 1) = mean(Case(:,2));
            Conf(n/5 + 3*0 + 7*0 , 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/5 + 3*0 + 7*0 , 1) = 1.645*std(Case(:,
2))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 1) = mean(Case(:,1));
            ResultVoip(n/5 + 3*1 + 7*0, 1) = mean(Case(:,2));
            Conf(n/5 + 3*1 + 7*0, 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/5 + 3*1 + 7*0, 1) = 1.645*std(Case(:,
2))/sqrt(10);
         end
```

```
        %ADD

         if f== 10
            Result(n/5 + 3*0 + 7*0, 2) = mean(Case(:,3));
            ResultVoip(n/5 + 3*0 + 7*0 , 2) = mean(Case(:,4));
            Conf(n/5 + 3*0 + 7*0 , 2) = 1.645*std(Case(:,
3))/sqrt(10);
            ConfVoip(n/5 + 3*0 + 7*0 , 2) = 1.645*std(Case(:,
4))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 2) = mean(Case(:,3));
            ResultVoip(n/5 + 3*1 + 7*0, 2) = mean(Case(:,4));
            Conf(n/5 + 3*1 + 7*0, 2) = 1.645*std(Case(:, 3))/sqrt(10);
            ConfVoip(n/5 + 3*1 + 7*0, 2) = 1.645*std(Case(:,
4))/sqrt(10);
         end

         %MDA

         if f== 10
            Result(n/5 + 3*0 + 7*0, 3) = mean(Case(:,5));
            ResultVoip(n/5 + 3*0 + 7*0 , 3) = mean(Case(:,6));
            Conf(n/5 + 3*0 + 7*0 , 3) = 1.645*std(Case(:,
5))/sqrt(10);
            ConfVoip(n/5 + 3*0 + 7*0 , 3) = 1.645*std(Case(:,
6))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 3) = mean(Case(:,5));
            ResultVoip(n/5 + 3*1 + 7*0, 3) = mean(Case(:,6));
            Conf(n/5 + 3*1 + 7*0, 3) = 1.645*std(Case(:, 5))/sqrt(10);
            ConfVoip(n/5 + 3*1 + 7*0, 3) = 1.645*std(Case(:,
6))/sqrt(10);
         end

         %TT

         if f== 10
            Result(n/5 + 3*0 + 7*0, 4) = mean(Case(:,7));
            Conf(n/5 + 3*0 + 7*0 , 4) = 1.645*std(Case(:,
7))/sqrt(10);
         else
            Result(n/5 + 3*1 + 7*0, 4) = mean(Case(:,7));
            Conf(n/5 + 3*1 + 7*0, 4) = 1.645*std(Case(:, 7))/sqrt(10);
         end

        disp(Result);
        disp(Conf);
        disp(ResultVoip);
        disp(ConfVoip);
     end
end
```

```
lambda = 10 ;

for f = [10 1]
    for n = [25 50 75]
         for k = 1:10
            [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim3(lambda*100,10,f*10000,100000,n);
            Case(k,1) = PL;
            Case(k,2) = PLvoip;
            Case(k,3) = APD;
            Case(k,4) = APDvoip;
            Case(k,5) = MPD;
            Case(k,6) = MPDvoip;
            Case(k,7) = TT;
         end

         %PL

         if f== 10
            Result(n/25 + 30 + 71, 1) = mean(Case(:,1));
            ResultVoip(n/25 + 31 + 71 , 1) = mean(Case(:,2));
            Conf(n/25 + 30 + 71 , 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/25 + 31 + 71 , 1) = 1.645*std(Case(:,
2))/sqrt(10);
         else
            Result(n/25 + 3*1 + 7*1, 1) = mean(Case(:,1));
            ResultVoip(n/25 + 3*1 + 7*1, 1) = mean(Case(:,2));
            Conf(n/25 + 3*1 + 7*1, 1) = 1.645*std(Case(:, 1))/sqrt(10);
            ConfVoip(n/25 + 3*1 + 7*1, 1) = 1.645*std(Case(:,
2))/sqrt(10);
         end

        %ADD

         if f== 10
            Result(n/25 + 3*0 + 7*1, 2) = mean(Case(:,3));
            ResultVoip(n/25 + 3*0 + 7*1 , 2) = mean(Case(:,4));
            Conf(n/25 + 3*0 + 7*1 , 2) = 1.645*std(Case(:, 3))/sqrt(10);
            ConfVoip(n/25 + 3*0 + 7*1 , 2) = 1.645*std(Case(:,
4))/sqrt(10);
         else
            Result(n/25 + 3*1 + 7*1, 2) = mean(Case(:,3));
            ResultVoip(n/25 + 3*1 + 7*1, 2) = mean(Case(:,4));
            Conf(n/25 + 3*1 + 7*1, 2) = 1.645*std(Case(:, 3))/sqrt(10);
            ConfVoip(n/25 + 3*1 + 7*1, 2) = 1.645*std(Case(:,
4))/sqrt(10);
         end
```

```
         %MDA

         if f== 10
            Result(n/25 + 3*0 + 7*1, 3) = mean(Case(:,5));
            ResultVoip(n/25 + 3*0 + 7*1 , 3) = mean(Case(:,6));
            Conf(n/25 + 3*0 + 7*1 , 3) = 1.645*std(Case(:,
5))/sqrt(10);
            ConfVoip(n/25 + 3*0 + 7*1 , 3) = 1.645*std(Case(:,
6))/sqrt(10);
         else
            Result(n/25 + 3*1 + 7*1, 3) = mean(Case(:,5));
            ResultVoip(n/25 + 3*1 + 7*1, 3) = mean(Case(:,6));
            Conf(n/25 + 3*1 + 7*1, 3) = 1.645*std(Case(:,
5))/sqrt(10);
            ConfVoip(n/25 + 3*1 + 7*1, 3) = 1.645*std(Case(:,
6))/sqrt(10);
         end

         %TT

         if f== 10
            Result(n/25 + 3*0 + 7*1, 4) = mean(Case(:,7));
            Conf(n/25 + 3*0 + 7*1 , 4) = 1.645*std(Case(:,
7))/sqrt(10);
         else
            Result(n/25 + 3*1 + 7*1, 4) = mean(Case(:,7));
            Conf(n/25 + 3*1 + 7*1, 4) = 1.645*std(Case(:,
7))/sqrt(10);
         end

        disp(Result);
        disp(Conf);
        disp(ResultVoip);
        disp(ConfVoip);
     end
end
```

Case	λ data (pps)	C (Mbps)	f (Bytes)	n	Packet Loss Data / VoIP (%)	Avg. Delay Data / VoIP (msec)	Max. Delay data / VoIP (msec)	Transm. Through. (Mbps)
A	200	2	100000	5	0 +/- 0 / 0 +/- 0	11.9807 +/- 0.0884 / 2.3117 +/- 0.0039	102.0998 +/- 8.7558 / 7.2451 +/- 0.0686	1.5752 +/- 0.0018
B	200	2	100000	10	0 +/- 0 / 0 +/- 0	26.5094 +/- 0.7484 / 2.6445 +/- 0.0072	204.3075 +/- 15.3271 / 7.7765 +/- 0.1140	1.8168 +/- 0.0035
C	200	2	100000	15	2.5863 +/- 0.1284 / 0.2038 +/- 0.0108	530.7263 +/- 5.7293 / 2.9902 +/- 0.0042	633.4766 +/- 0.3971 / 8.4033 +/- 0.1181	1.9996 +/- 0.0001
D	200	2	10000	5	0.2478 +/- 0.0169 / 0.0180 +/- 0.0027	11.2542 +/- 0.0698 / 2.2976 +/- 0.0047	50.2874 +/- 0.1040 / 7.1855 +/- 0.0455	1.5683 +/- 0.0021
E	200	2	10000	10	1.2498 +/- 0.0331 / 0.4137 +/- 0.0060	18.6390 +/- 0.1278 / 2.5972 +/- 0.0043	57.9324 +/- 0.2978 / 7.7920 +/- 0.1572	1.7898 +/- 0.0027
F	200	2	10000	15	5.2669 +/- 0.0790 / 0.4137 +/- 0.0131	34.1557 +/- 0.2014 / 2.8675 +/- 0.0042	68.8149 +/- 0.3311 / 8.3913 +/- 0.1414	1.9473 +/- 0.0013

Case	λ data (pps)	C (Mbps)	f (Bytes)	n	Packet Loss Data / VoIP (%)	Avg. Delay Data / VoIP (msec)	Max. Delay data / VoIP (msec)	Transm. Through. (Mbps)
G	1000	10	100000	25	0 +/- 0 / 0 +/- 0	2.4081 +/- 0.0234 / 0.4661 +/- 0.0009	19.8780 +/- 1.0880 / 1.4528 +/- 0.0144	7.8807 +/- 0.0099
H	1000	10	100000	50	0 +/- 0 / 0 +/- 0	5.2832 +/- 0.1148 / 0.5324 +/- 0.0016	41.2896 +/- 2.8120 / 1.6390 +/- 0.0305	9.0689 +/- 0.0211
I	1000	10	100000	75	0 +/- 0 / 0 +/- 0	106.3893 +/- 0.9621 / 0.6014 +/- 0.0006	127.0949 +/- 0.1026 / 1.7207 +/- 0.0188	9.9986 +/- 0.0005
J	1000	10	10000	25	0.2513 +/- 0.0192 / 0.0215 +/- 0.0035	2.2743 +/- 0.0188 / 0.4647 +/- 0.0007	10.2721 +/- 0.0544 / 1.4803 +/- 0.0204	7.8517 +/- 0.0128
K	1000	10	10000	50	1.2713 +/- 0.0285 / 0.0956 +/- 0.0060	3.7355 +/- 0.0139 / 0.5227 +/- 0.0008	12.0251 +/- 0.0927 / 1.5986 +/- 0.0228	8.9449 +/- 0.0053
L	1000	10	10000	75	5.2914 +/- 0.0706 / 0.4290 +/- 0.0125	6.8014 +/- 0.0248 / 0.5765 +/- 0.0009	14.5673 +/- 0.0932 / 1.7237 +/- 0.0358	9.7274 +/- 0.0046

m) Analysis:

When analyzing the data obtained, we noticed that the packet loss data behavior was the same as that explained in **k)**: the packet rate and connection capacity do not seem to affect the values when incremented at the same rate. Decreasing the queue size though seems to cause a higher packet loss to happen. Packet loss also increases with VoIP flow increases. Once again, this is the case for both data flows. The VoIP flows displayed, once again, much smaller values than the data flow, as well as smaller confidence intervals.

When analyzing the data obtained, we noticed that the average packet delay data behavior was the same as that explained in **k)**: when both the packet rate and the connection capacity increased, the average packet delay decreased. For the same packet rate and connection capacity values, as the number of packet flows increases the average packet delay increases as well. This increase is less impactful when the queue size is decreased. In this simulator though, both flows did not obtain similar results, having the VoIP flow displayed much smaller results.

When analyzing the data obtained, we noticed that the average packet delay data behavior was the same as that explained in **k)**: similarly to the average packet delay, the maximum packet delay seems to decrease as the values of both the packet rate and the connection capacity increase. The conclusions taken regarding queue size and VoIP flows also stand. Just as it was the case for the average packet delay values in this simulator, the maximum packet delay values too were vastly different, contrasting the witnessed in **k)**. The VoIP flow saw, once again, much smaller values.

When analyzing the data obtained, we noticed that the average packet delay data behavior was the same as that explained in **k)**: when increasing both the packet rate and the connection capacity, both the transmitted throughput and its confidence interval seem to increase. When the queue size decreases though, transmitted throughput also decreases, maintaining similar incrementations for the same packet rate and connection capacity values as VoIP flows increase. Meanwhile, while increasing VoIP flows, the transmitted throughput also seems to increase, bolstering higher increments for larger packet rate and connection capacity values. The transmitted throughput values though, were very much just similar.

Simulator 4:

```
function [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim4(lambda,C,f,P,n,r)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
% OUTPUT PARAMETERS:
%  PL   - packet loss (%)
%  APD  - average packet delay (milliseconds)
%  MPD  - maximum packet delay (milliseconds)
%  TT   - transmitted throughput (Mbps)

%Events:
ARRIVAL= 0;       % Arrival of a packet
DEPARTURE= 1;     % Departure of a packet

DATA= 1;
VOIP= 0;

%State variables:
State = 0;          % 0 - connection free; 1 - connection bysy
QueueOccupation= 0; % Occupation of the queue (in Bytes)
Queue= [];          % Size and arriving time instant of each packet in
the queue

%Statistical Counters:
TotalPackets= 0;       % No. of packets arrived to the system
LostPackets= 0;        % No. of packets dropped due to buffer overflow
TransmittedPackets= 0; % No. of transmitted packets
TransmittedBytes= 0;   % Sum of the Bytes of transmitted packets
Delays= 0;             % Sum of the delays of transmitted packets
MaxDelay= 0;           % Maximum delay among all transmitted packets
TotalPacketsVoip= 0;
LostPacketsVoip= 0;
TransmittedPacketsVoip= 0;
DelaysVoip= 0;
MaxDelayVoip= 0;

%Auxiliary variables:
% Initializing the simulation clock:
Clock= 0;

% Initializing the List of Events with the first ARRIVAL:
EventList = [ARRIVAL , Clock + exprnd(1/lambda) , GeneratePacketSize() ,
0, DATA];

for i=1:n
    EventList= [EventList; ARRIVAL , Clock +  0.02*rand(),
GeneratePacketSizeVoip() , 0, VOIP];
end
```

```
%Similation loop:
while TransmittedPackets<P                % Stopping criterium
    EventList= sortrows(EventList,2);     % Order EentList by time
    Event= EventList(1,1);                % Get first event and
    Clock= EventList(1,2);                %   and
    PacketSize= EventList(1,3);           %   associated
    ArrivalInstant= EventList(1,4);       %   parameters.
    Type = EventList(1,5);
    EventList(1,:)= [];                   % Eliminate first event
    switch Event
        case ARRIVAL                      % If first event is an ARRIVAL
            if Type == DATA
                TotalPackets= TotalPackets+1;
                   EventList = [EventList; ARRIVAL , Clock +
exprnd(1/lambda) , GeneratePacketSize() , 0, DATA];
            else
                TotalPacketsVoip= TotalPacketsVoip +1;
                   EventList= [EventList; ARRIVAL , Clock +
0.001*randi([16,24]), GeneratePacketSizeVoip() , 0, VOIP];
            end

            if State==0
                State= 1;
                EventList = [EventList; DEPARTURE , Clock +
8*PacketSize/(C*10^6) , PacketSize , Clock, Type];
            else

                if (QueueOccupation <= f*r) || (QueueOccupation +
PacketSize <= f && Type==VOIP)
                    Queue= [Queue;PacketSize , Clock, Type];
                    Queue= sortrows(Queue,3);

                    QueueOccupation= QueueOccupation + PacketSize;
                else
                    if Type == DATA
                        LostPackets= LostPackets + 1;
                    else
                        LostPacketsVoip= LostPacketsVoip +1;
                    end
                end
            end
        case DEPARTURE                    % If first event is a
DEPARTURE
            TransmittedBytes= TransmittedBytes + PacketSize;

            if Type == DATA
                Delays= Delays + (Clock - ArrivalInstant);
            else
                DelaysVoip= DelaysVoip + (Clock - ArrivalInstant);
            end
```

```
            if Clock - ArrivalInstant > MaxDelay
                if Type == DATA
                    MaxDelay= Clock - ArrivalInstant;
                end
            end
            if Clock - ArrivalInstant > MaxDelayVoip
                if Type == VOIP
                    MaxDelayVoip= Clock - ArrivalInstant;
                end
            end

            if Type == DATA
                TransmittedPackets= TransmittedPackets + 1;
            else
                TransmittedPacketsVoip= TransmittedPacketsVoip + 1;
            end

            if QueueOccupation > 0
                EventList = [EventList; DEPARTURE , Clock +
8*Queue(1,1)/(C*10^6) , Queue(1,1) , Queue(1,2), Queue(1,3)];
                QueueOccupation= QueueOccupation - Queue(1,1);
                Queue(1,:)= [];
            else
                State= 0;
            end
    end
end

%Performance parameters determination:
PL= 100*LostPackets/TotalPackets      % in %
PLvoip= 100*LostPacketsVoip/TotalPacketsVoip
APD= 1000*Delays/TransmittedPackets   % in milliseconds
APDvoip= 1000*DelaysVoip/TransmittedPacketsVoip
MPD= 1000*MaxDelay                    % in milliseconds
MPDvoip= 1000*MaxDelayVoip
TT= 10^(-6)*TransmittedBytes*8/Clock  % in Mbps

end

function out= GeneratePacketSize()
    aux= rand();
    if aux <= 0.16
        out= 64;
    elseif aux >= 0.78
        out= 1518;
    else
        out = randi([65 1517]);
    end
end

function out= GeneratePacketSizeVoip()
    out = randi([110 130]);
end
```

p) Data gathering:

```
Result = zeros(100,4);
Conf = zeros(100,4);

ResultVoip = zeros(100,4);
ConfVoip = zeros(100,4);

Case = zeros(10,7);

lambda = 2;

for r = [0.25 0.5 0.75]
        for n = [5 10 15]
             for k = 1:10
                [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim4(lambda*100,2,10000,100000,n,r);
                Case(k,1) = PL;
                Case(k,2) = PLvoip;
                Case(k,3) = APD;
                Case(k,4) = APDvoip;
                Case(k,5) = MPD;
                Case(k,6) = MPDvoip;
                Case(k,7) = TT;
             end

             %PL
                Result(n/5 + 3*1 + 7*0 +12*(r*4-1), 1) = mean(Case(:,1));
                ResultVoip(n/5 + 3*1 + 7*0 +12*(r*4-1), 1) =
mean(Case(:,2));
                Conf(n/5 + 3*1 + 7*0 +12*(r*4-1), 1) = 1.645*std(Case(:,
1))/sqrt(10);
                ConfVoip(n/5 + 3*1 + 7*0 +12*(r*4-1), 1) =
1.645*std(Case(:, 2))/sqrt(10);

            %ADD
                Result(n/5 + 3*1 + 7*0 +12*(r*4-1), 2) = mean(Case(:,3));
                ResultVoip(n/5 + 3*1 + 7*0 +12*(r*4-1), 2) =
mean(Case(:,4));
                Conf(n/5 + 3*1 + 7*0 +12*(r*4-1), 2) = 1.645*std(Case(:,
3))/sqrt(10);
                ConfVoip(n/5 + 3*1 + 7*0 +12*(r*4-1), 2) =
1.645*std(Case(:, 4))/sqrt(10);

             %MDA
                Result(n/5 + 3*1 + 7*0 +12*(r*4-1), 3) = mean(Case(:,5));
                ResultVoip(n/5 + 3*1 + 7*0 +12*(r*4-1), 3) =
mean(Case(:,6));
                Conf(n/5 + 3*1 + 7*0 +12*(r*4-1), 3) = 1.645*std(Case(:,
5))/sqrt(10);
                ConfVoip(n/5 + 3*1 + 7*0 +12*(r*4-1), 3) =
1.645*std(Case(:, 6))/sqrt(10);
```

```
            %TT
               Result(n/5 + 3*1 + 7*0 +12*(r*4-1), 4) =
mean(Case(:,7));
               Conf(n/5 + 3*1 + 7*0 +12*(r*4-1), 4) =
1.645*std(Case(:, 7))/sqrt(10);

            disp(Result);
            disp(Conf);
            disp(ResultVoip);
            disp(ConfVoip);
         end
end

lambda = 10 ;

for r = [0.25 0.5 0.75]
        for n = [25 50 75]
            for k = 1:10
               [PL, PLvoip, APD, APDvoip, MPD ,MPDvoip ,TT] =
sim4(lambda*100,10,10000,100000,n,r);
               Case(k,1) = PL;
               Case(k,2) = PLvoip;
               Case(k,3) = APD;
               Case(k,4) = APDvoip;
               Case(k,5) = MPD;
               Case(k,6) = MPDvoip;
               Case(k,7) = TT;
            end

            %PL
               Result(n/25 + 3*1 + 7*1+12*(r*4-1), 1) =
mean(Case(:,1));
               ResultVoip(n/25 + 3*1 + 7*1+12*(r*4-1), 1) =
mean(Case(:,2));
               Conf(n/25 + 3*1 + 7*1+12*(r*4-1), 1) =
1.645*std(Case(:, 1))/sqrt(10);
               ConfVoip(n/25 + 3*1 + 7*1+12*(r*4-1), 1) =
1.645*std(Case(:, 2))/sqrt(10);

            %ADD
               Result(n/25 + 3*1 + 7*1+12*(r*4-1), 2) =
mean(Case(:,3));
               ResultVoip(n/25 + 3*1 + 7*1+12*(r*4-1), 2) =
mean(Case(:,4));
               Conf(n/25 + 3*1 + 7*1+12*(r*4-1), 2) =
1.645*std(Case(:, 3))/sqrt(10);
               ConfVoip(n/25 + 3*1 + 7*1+12*(r*4-1), 2) =
1.645*std(Case(:, 4))/sqrt(10);
```

```
            %MDA
               Result(n/25 + 3*1 + 7*1+12*(r*4-1), 3) =
mean(Case(:,5));
               ResultVoip(n/25 + 3*1 + 7*1+12*(r*4-1), 3) =
mean(Case(:,6));
               Conf(n/25 + 3*1 + 7*1+12*(r*4-1), 3) =
1.645*std(Case(:, 5))/sqrt(10);
               ConfVoip(n/25 + 3*1 + 7*1+12*(r*4-1), 3) =
1.645*std(Case(:, 6))/sqrt(10);

%TT

               Result(n/25 + 3*1 + 7*1+12*(r*4-1), 4) =
mean(Case(:,7));
               Conf(n/25 + 3*1 + 7*1+12*(r*4-1), 4) =
1.645*std(Case(:, 7))/sqrt(10);

            disp(Result);
            disp(Conf);
            disp(ResultVoip);
            disp(ConfVoip);
         end
end
```

For r = 25%:

Case	λ data (pps)	C (Mbps)	f (Bytes)	n	Packet Loss Data / VoIP (%)	Avg. Delay Data / VoIP (msec)	Max. Delay data / VoIP (msec)	Transm. Through. (Mbps)
D	200	2	10000	5	7.0219 +/- 0.0679 / 0 +/- 0	7.3214 +/- 0.0251 / 2.1791 +/- 0.0038	23.8535 +/- 0.1837 / 7.2176 +/- 0.0546	1.4833 +/- 0.0021
E	200	2	10000	10	11.1283 +/- 0.0728 / 0 +/- 0	8.8379 +/- 0.0268 / 2.3995 +/- 0.1308	27.2438 +/- 0.2846 / 7.8115 +/- 0.0847	1.6661 +/- 0.0019
F	200	2	10000	15	17.5923 +/- 0.0772 / 0 +/- 0	11.2154 +/- 0.0253 / 2.6346 +/- 0.0035	32.2962 +/- 0.5858 / 8.2635 +/- 0.1217	1.8192 +/- 0.0013
J	1000	10	10000	25	7.0560 +/- 0.0883 / 0 +/- 0	1.4647 +/- 0.0028 / 0.4399 +/- 0.0007	5.0707 +/- 0.0482 / 1.4838 +/- 0.0147	7.3958 +/- 0.0071
K	1000	10	10000	50	11.3312 +/- 0.0706 / 0 +/- 0	1.7769 +/- 0.0019 / 0.4843 +/- 0.0008	5.9236 +/- 0.0783 / 1.6027 +/- 0.0147	8.3192 +/- 0.0060
L	1000	10	10000	75	18.0319 +/- 0.0828 / 0 +/- 0	2.2567 +/- 0.0046 / 0.5325 +/- 0.0009	7.3717 +/- 0.1198 / 1.7616 +/- 0.0484	9.0790 +/- 0.0039

For r = 50%:

Case	λ Data (pps)	C (Mbps)	f (Bytes)	n	Packet Loss Data / VoIP (%)	Avg. Delay Data / VoIP (msec)	Max. Delay data / VoIP (msec)	Transm. Through. (Mbps)
D	200	2	10000	5	2.1022 +/- 0.0581 / 0 +/- 0	9.7181 +/- 0.1110 / 2.2721 +/- 0.0052	35.3984 +/- 0.1842 / 7.2562 +/- 0.0568	1.5480 +/- 0.0028
E	200	2	10000	10	4.8786 +/- 0.0741 / 0 +/- 0	13.3483 +/- 0.0542 / 2.5319 +/- 0.0055	40.4085 +/- 0.2124 / 7.7442 +/- 0.1082	1.7493 +/- 0.0016
F	200	2	10000	15	11.5198 +/- 0.1110 / 0 +/- 0	19.7567 +/- 0.0471 / 2.8050 +/- 0.0029	47.2594 +/- 0.4456 / 8.3486 +/- 0.0979	1.9010 +/- 0.0006
J	1000	10	10000	25	2.0842 +/- 0.0413 / 0 +/- 0	1.9418 +/- 0.0095 / 0.4578 +/- 0.0012	7.2871 +/- 0.1050 / 1.4759 +/- 0.0204	7.7293 +/- 0.0154
K	1000	10	10000	50	4.9824 +/- 0.0621 / 0 +/- 0	2.6771 +/- 0.0103 / 0.5103 +/- 0.0012	8.5650 +/- 0.1296 / 1.5629 +/- 0.0176	8.7356 +/- 0.0103
L	1000	10	10000	75	11.6548 +/- 0.0706 / 0 +/- 0	3.9480 +/- 0.0152 / 0.5641 +/- 0.0007	10.3396 +/- 0.1300 / 1.7478 +/- 0.0222	9.4974 +/- 0.0055

For r = 75%:

Case	λ data (pps)	C (Mbps)	f (Bytes)	n	Packet Loss Data / VoIP (%)	Avg. Delay Data / VoIP (msec)	Max. Delay data / VoIP (msec)	Transm. Through. (Mbps)
D	200	2	10000	5	0.6943 +/- 0.0332 / 0 +/- 0	10.9362 +/- 0.0706 / 2.2990 +/- 0.0071	45.8314 +/- 0.2458 / 7.2014 +/- 0.0053	1.5660 +/- 0.0042
E	200	2	10000	10	2.6023 +/- 0.0861 / 0 +/- 0	16.9647 +/- 0.1357 / 2.5896 +/- 0.0053	53.4319 +/- 0.3749 / 7.6684 +/- 0.0457	1.7815 +/- 0.0021
F	200	2	10000	15	8.9516 +/- 0.1157 / 0 +/- 0.0001	28.7653 +/- 0.0984 / 2.8728 +/- 0.0022	62.4143 +/- 0.3804 / 8.3991 +/- 0.1352	1.9366 +/- 0.0008
J	1000	10	10000	25	0.6615 +/- 0.0310 / 0 +/- 0	2.1795 +/- 0.0156 / 0.4625 +/- 0.0015	9.3534 +/- 0.0517 / 1.4630 +/- 0.0119	7.8165 +/- 0.0167
K	1000	10	10000	50	2.6015 +/- 0.0715 / 0 +/- 0	3.3859 +/- 0.0234 / 0.5208 +/- 0.0011	11.0993 +/- 0.0866 / 1.5931 +/- 0.0169	8.9033 +/- 0.0115
L	1000	10	10000	75	8.9694 +/- 0.1318 / 0.0001 +/- 0.0002	5.7317 +/- 0.0294 / 0.5791 +/- 0.0013	13.4328 +/- 0.1146 / 1.7649 +/- 0.0466	9.6776 +/- 0.0058

q) Analysis:

- Packet loss:

The smaller the queue size that can be used to store packages of data is, the larger will be the number of data packages lost.

The smaller the queue size that can be used to store packages of data is, the lower will be the number of VOIP packages lost.

The two previous sentences can be explained by two examples/with two examples:

- If 75% of the queue can be used for both type of packages, 25% exist that can solely be used by VoIP packages. (less space for data, more losses; more space for VOIP, less losses).
- If 25% of the queue can be used for both type of packages, 75% exist that can solely be used by VoIP packages. (less space for data, more losses; more space for VOIP, less losses).

The number of VOIP packages lost in simulator 4 is almost always zero. Except the case where there are 75 flows of VOIP packages and just 25% of the queue that just can be used by VOIP packages. It is though, in all cases, lower than simulator 3. The exception happens because the number of flows of VOIP, creates enough packages to occupy the space that cannot be occupied by data packages and competes with data packages to occupy the other side of the queue.

Unless the number of VOIP flows is enough (VOIP packages are less than data packages) , in the case of some queue space not being able to have data packages, the Queue will not be full, as we can see by the existence of 0 VOIP package losses and lower Transmitted Throughput.

- Average and Maximum Packet Delays relation with Packet Rate:

Simulator 4's delays are lower than simulator 3's when regarding the data flows due to more data packet losses (Less data packets queued, less wait time, less delay).

VoIP flow packet delays did not vary much from simulator 3 to simulator 4.

- Transmitted Throughput relation with Packet Rate

Decreases when the queue is not fully occupied. In the simulator 4, this behavior can be seen when there are no VOIP packets losses. This happen because VOIP packages are small and few, depending on the number of flows in this last, what associated with part of the queue can just be used by this type of packages leads to unused space.

Increases when more space of the queue can be used for data packets, due to these packets being bigger and a lot more.