Information Networks
TSIN01 Assignment

Tinhinene Ait Hamouda
920612-T403

October 2016

1 Introduction

The objective of this assignment is to implement slotted ALOHA protocol in a programming language in order to simulate its behaviour. This report will cover documentation about an implementation done on MATLAB as well as results discussions.

2 Slotted ALOHA

2.1 Protocol Flowchart

In each time slot, if a new packets arrives at an unbacklogged node, then the packet will be transmitted with a probability of q_a. But, if this packets arrives at a backlogged node, then it would be discarded. If a node is already backlogged, then it tries to retransmit with probability q_r. Figure 1 above shows this process.

2.2 Implementation on MATLAB

Slotted ALOHA Protocol was implemented on MATLAB. Basically, the strategy followed simulates the whole system at each time slot. Probabilities Qa and Qr are calculated, and then different cases are tested. For example, if we're at a situation where one backlogged node is retransmitting and no unbacklogged is transmitting, then this is a successful transmission. Afterwards, depending on whether it is a collision, feedback or idle situation, the state of backlog is updated and the packets entering/leaving the system as well.
As for question 4, the attempt rate is calculated at each time slot using the current value the backlog. And the probability of success, it is calculated for each possible value of the backlog n, using the formula $P_s = Q_a(1,n) * Q_r(0,n) + Q_a(0,n) * Q_r(1,n)$, and then it is multiplied by the probability (frequency) that this state n appears in the system. Finally, the frequency of success is obtained from simulation results by simply counting the number of successful transmission and dividing it by $t = 1000$.

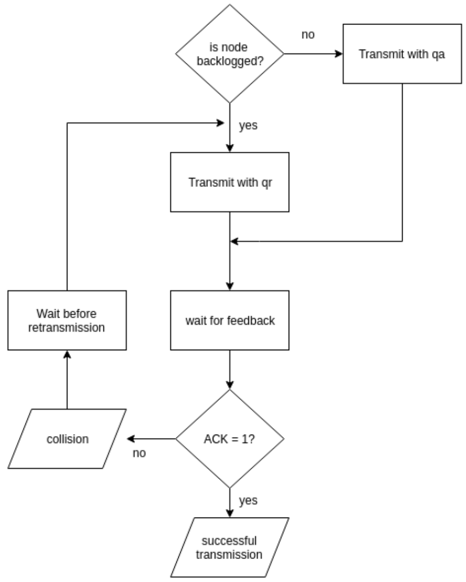

Figure 1: Slotted ALOHA flowchart

2.3 Simulation with $q_r = 0.01$

Simulation results for this case shows the following:

- The backlog is a significant fraction of m, but it increases in a reasonable way.
- The gap between the packets entering and leaving can be improved in the stabilized slotted ALOHA.
- The probability of success can be improved (It is around 0.32) as we know that the maximum is 1/e.
- The attempt rate increases when n increases and decreases with n as well. This can be verified by the formula $G(n) = (m - n) * q_a + n * q_r$

Below are shown figures obtained for question 3, regarding the backlog of the system, and the packets entering/leaving the system.

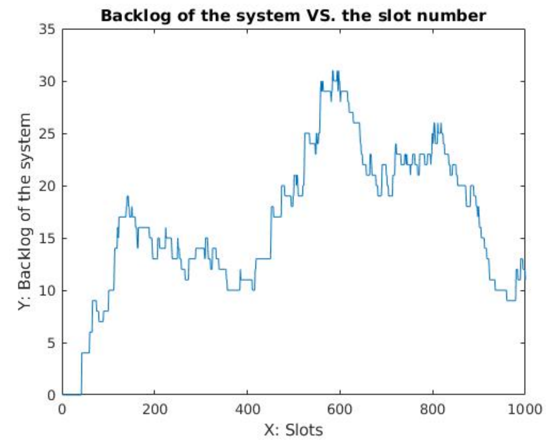

Figure 2: System Backlog

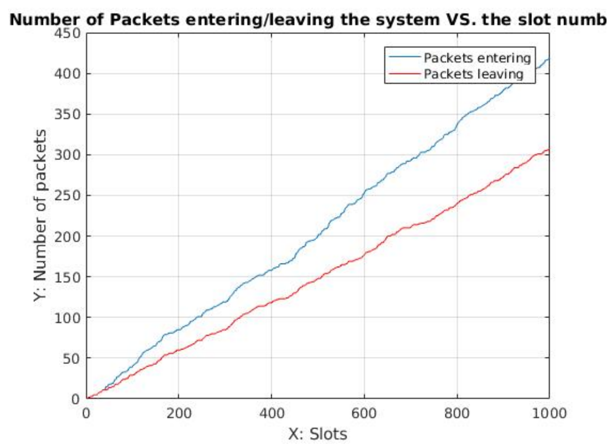

Figure 3: Packets entering vs. Packets leaving

Below are shown figures obtained for question 4, regarding the histogram of the backlog and the attempt rate.

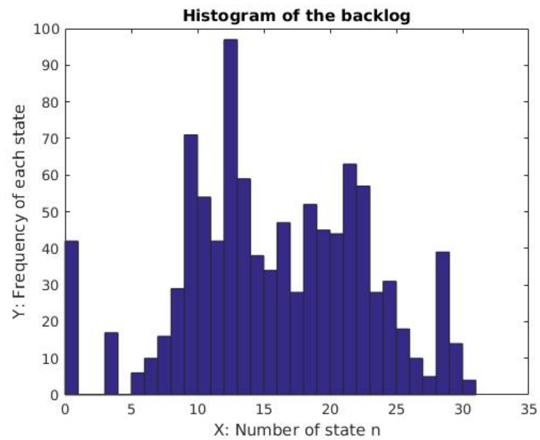

Figure 4: Histogram of the Backlog

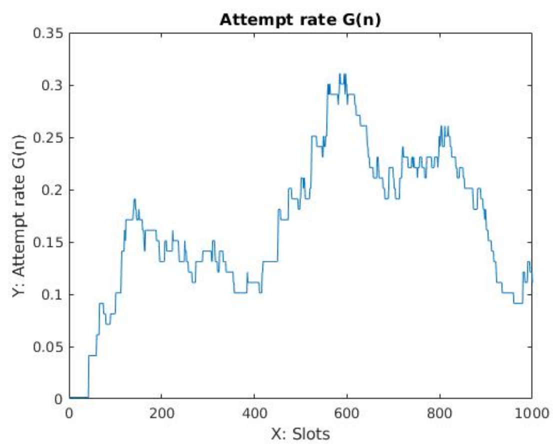

Figure 5: The attempt rate

2.4 Simulation with $\lambda = 0.5$

Simulation results for this case shows the following:

- The gap between the packets entering and leaving the system is larger. In fact, increasing lambda will affect the attempt rate, and as G increases the number of collisions increases, in other words not so many packets are leaving the system.
- Since the number of collisions is more important in this case, the backlog of the system is higher.

- Increasing the value of λ (The system was tested with 0.5, 0.6, 0.8) will not increase the probability of success more than $1/e$.

Figures obtained for this simulation ($\lambda = 0.5$, $q_r = 0.01$), are shown down below.

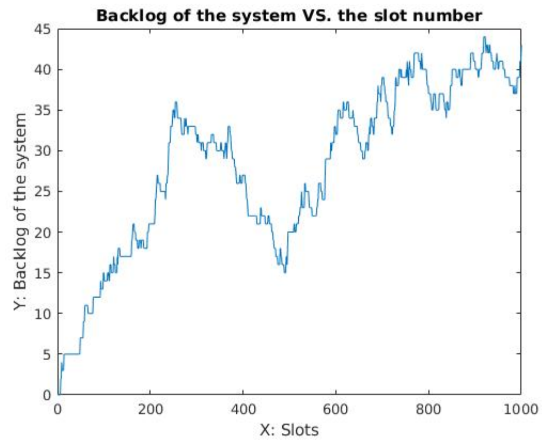

Figure 6: System Backlog

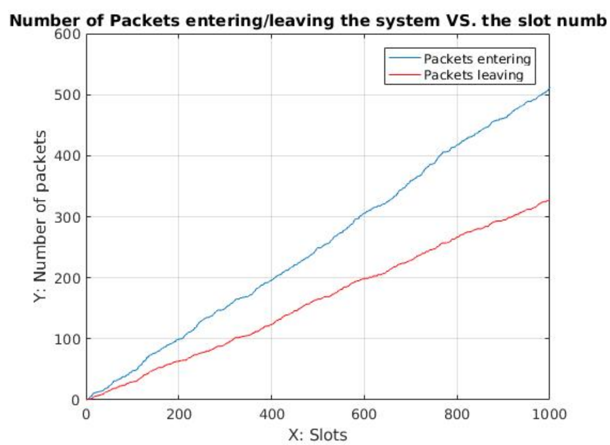

Figure 7: Packets entering/leaving the system

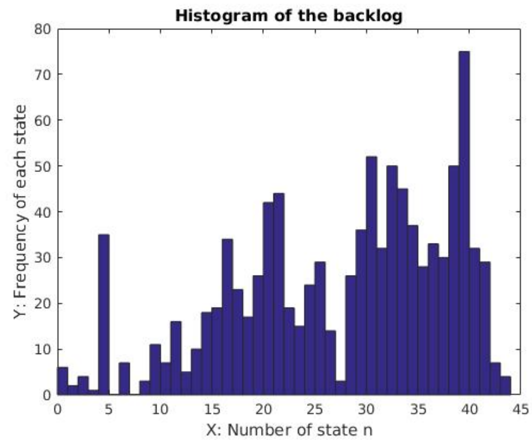

Figure 8: Histogram of the Backlog

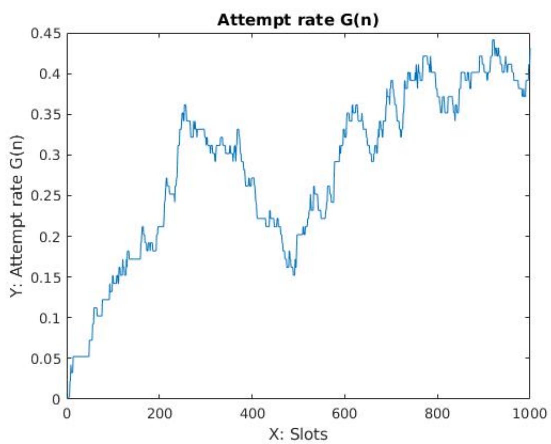

Figure 9: The attempt rate

2.5 Simulation with $q_r = 0.1$

Simulation results for this case shows the following:

- The attempt rate $G(n)$ increases with n faster when the value of q_r is higher.
- The system is heavily backlogged and this happens when $G(n) > 1$.
- A large number of nodes becomes backlogged very quickly and thus an important number of new packets arrival are discarded.

- When q_r is very large, the system saturates. This statement can be verified with larger values than 0.1 as well.
- Another observation is that for some simulations, the backlog can become zero at some given time slots and this is due to the fact that q_r is not a very small value compared to the λ.

Figures obtained for the simulation with $q_r = 0.1$ and $\lambda = 0.37$), are shown down below.

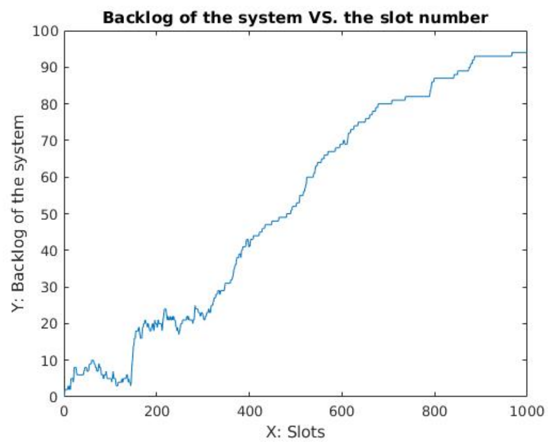

Figure 10: System Backlog

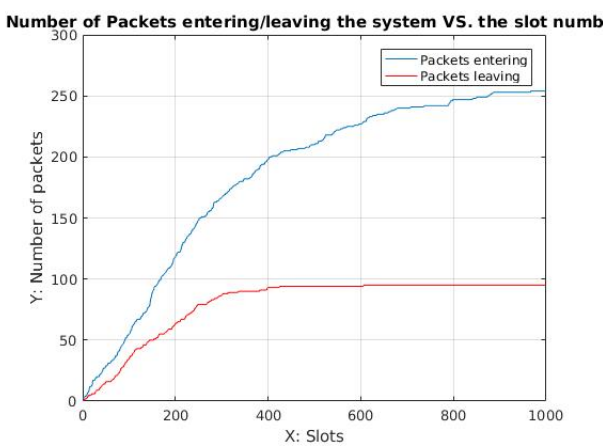

Figure 11: Packets entering/leaving the system

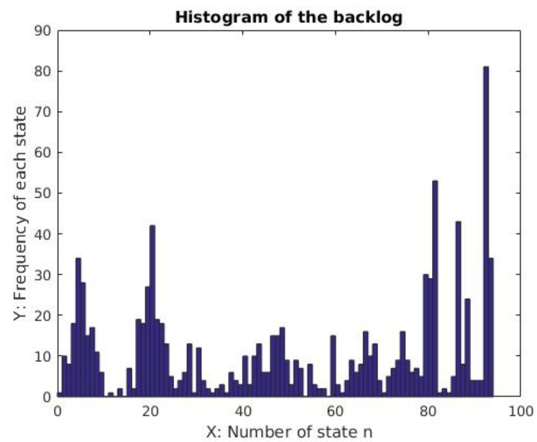

Figure 12: Histogram of the Backlog

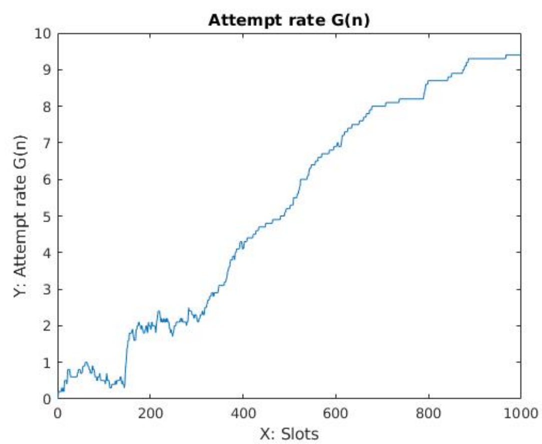

Figure 13: The attempt rate

3 Pseudo Bayesian Slotted ALOHA

The following subsections cover a Pseudo Bayesian stabilized slotted ALOHA.

3.1 Flowchart

The Pseudo Bayesian stabilization states that when a new packets arrives at an unbacklogged node, there is a probability q_r that this packets will be transmitted. If the node is already backlogged, then new arrivals are discarded.

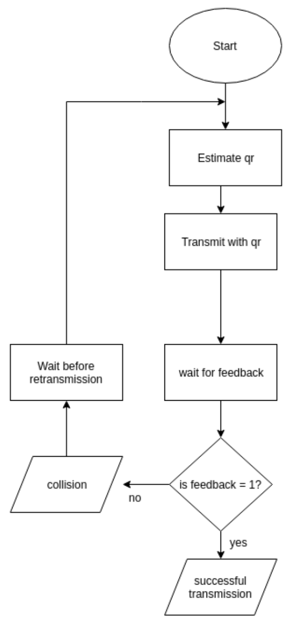

Figure 14: Pseudo Bayesian Slotted ALOHA flowchart

The flowchart shows that regardless of the node status, it will transmit with probability q_r. Figure 2 shows this process.

3.2 Implementation

At each time slot, Poisson packets arrival are simulated, then the number of packets entering the system is calculated. This value depend on the number of nodes that are unbacklogged, because an already backlogged node would discard any new arrival. Next, q_r is calculated, and the number of transmissions occuring in this time slot is obtained. Depending on this number, 0 (idle), 1 (success) or $>=$ 2 (collision), the number of packets leaving is calculated and a new value for the backlog is estimated. This value is important for the calculation of q_r at each time slot.

3.3 Discussion of findings

Clearly, the gap between packets arrival and packets leaving is closed. The reason is that the system is stabilized with the Pseudo Bayesian, which estimates the backlog at each time t_t slot depending on the feedback at t_{t-1} and then adapts q_r. If too many idles, increase q_r, if too many collisions then decrease q_r. Moreover, the advantage of a such stabilization is that the system will return satisfying results for different values of λ.

Figures obtained for a stablized slotted aloha are shown down below.

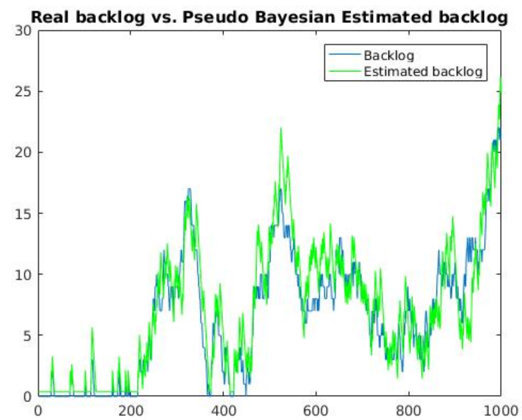

Figure 15: System Backlog

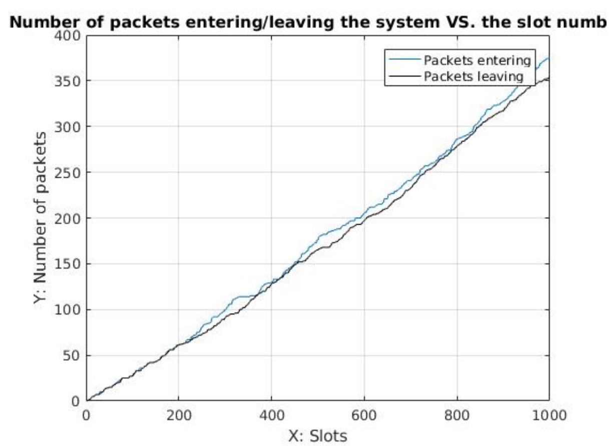

Figure 16: Packets entering/leaving the system

3.4 System Delay

The delay of the system using simulation results is calculated by tracking the time a packet enters the system and the time it leaves (successfly transmitted). The Pseudo Bayesian delay is calculated using the theoretical formula.
It can be seen that when λ is small numerical and theoretical values are very close, but as λ gets greater the gap between numerical and theoretical values becomes important. Another observation is that that ALOHA achieves lower delays when the arrival rate λ is small. Note that the source code used for this part calculations is attached in the Annex C.

Figure down below shows the system (theoretical/real) delay.

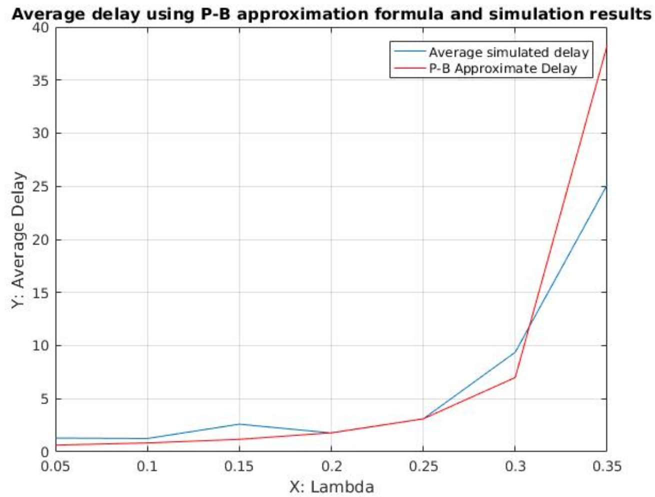

Figure 17: System delay (real/theoretical)

4 Conclusion

All simulations were done with the no-buffering assumption ($m = 100$). The stabilized slotted ALOHA has a defined steady state behaviour for different arrival rates, as it has been tested on the last part. But the system discards a large number of arriving packages and thus has an important finite delay. Its maximum throughput is around $1/e$, this can be improved by more sophisticated algorithms (Splitting Algorithms) seen in the second part of the course.

A Slotted ALOHA

```
%Name: Tinhinene AIT HAMOUDA
%Personal Number: 920612-T403
%Implementation of slotted ALOHA
function out=slotted_aloha(lambda,q_r,m)
%*****************************************************************
% lambda: total arrival rate
% q_a: transm. prob. of an unbacklogged node
% q_r: retransmission prob. of backlogged nodes
% m: total number of nodes
% n: number of backlogged nodes
% m-n: number of unbacklogged nodes
% t: time
%******************************************************************
t=1000;%Time in slots
slots=1:t;%Slots array
n=0;%Backlogged nodes at the begining = 0
backlog=zeros(size(1:t));%Backlog array of the system
packet_arr = 1:t;%Array of packets arriving
packet_leav = 1:t;%Array of packets leaving
succ=0;%Counter of the successful transmissions
state_probs=zeros(size(1:m));
Att_Rate = 1:t; %Attemp rate array

%Check lambda and m value
if lambda<=0 || lambda >1
  fprintf ('Syntax: s_aloha(lambda,q_r,m)');
  error ('Bad parameter: lambda shoud be between 0 and 0.36 ');
elseif m~=ceil(m)||m==0
  fprintf ('Syntax: s_aloha(lambda,q_r,m)');
  error ('Bad parameter: m should be a positive integer ');
end

%Calculate q_a: The prob. of an unbacklogged node to send a packet
q_a=1-exp(1)^(-lambda/m);
fprintf('q_a is %f \n',q_a);

```

```
%Check q_r value
if q_r<q_a || q_r>=1
  fprintf ('Syntax: s_aloha(lambda,q_r,m)');
  error ('Bad parameter: q_r should be between q_a and 1
      ');
end

%Print values
fprintf('lambda is %f \n',lambda);
fprintf('q_r is %f \n',q_r);
fprintf('m is %d \n',m);
pwd

%Overall loop from slot 1 to 1000
for i=1:t

   %Calculate Qa and Qr probabilities of the system
   Qa = zeros(size(1:101));
   Qr = zeros(size(1:101));
   for j=0:100
        Qa(j+1)=binopdf(j,m-n,q_a);
   end
   for j=0:100
       Qr(j+1)=binopdf(j,n,q_r);
   end

   %Generation of two random probabilities for the sake
       of comparison
   pQa = rand(1);
   pQb = rand(1);

   %We have two overall cases:
   %1. No backlogged node (n==0)
   %2. There is at least one backlogged node (n!=0)

   %Case 01: No backlogged node
   if n == 0
      if 0 <= pQa && pQa <= sum(Qa(1:1)) %Idle slot (Note
           that Qa(1) means j=0, no unbacklogged node is
          transmitting)
         packet_arr(i)=0;
         packet_leav(i)=0;
      elseif sum(Qa(1:1))<pQa && pQa<=sum(Qa(1:2)) %
          Succssful slot (1 unbacklogged node is
          transmitting)
          packet_arr(i)=1;
```

```
            packet_leav(i)=1;
            succ=succ+1;%Update succssful
        elseif sum(Qa(1:2))< pQa && pQa <=1 %Collision slot (More than 1 unbacklogged node is transmitting)
            x=1;
            while x<101
              if sum(Qa(1:x))>=pQa
                  k=x-1; %k represents the number of unbacklogged node which are transmitting at the same time
                  break;
              end
              x = x+1;
            end
            n = n+k; %Update the backlog of the system
            packet_arr(i)=k;
            packet_leav(i)=0; %No packet will leave because it is a collision case
        end
    %Case 02: There is at least one backlogged node
    else
        if 0 <= pQa && pQa <= sum(Qr(1:1)) %No backlogged node is transmitting
              if 0 <= pQb && pQb <= sum(Qa(1:1)) %Idle slot
                  packet_arr(i)=0;
                  packet_leav(i)=0;
              elseif sum(Qa(1:1)) < pQb && pQb <= sum(Qa(1:2)) %Succssful slot
                  packet_arr(i)=1;
                  packet_leav(i)=1;
                  succ=succ+1;
              elseif sum(Qa(1:2))<pQb && pQb<=1 %Collision slot
                  x=1;
                  while x<101
                      if sum(Qa(1:x))>=pQb
                          k=x-1;
                          break;
                      end
                      x = x+1;
                  end
                  n=n+k;
                  packet_arr(i)=k;
                  packet_leav(i)=0;
              end
```

```
        elseif sum(Qr(1:1)) < pQa && pQa <= sum(Qr(1:2)) %One backlogged node is retransmitting
            if 0 <= pQb && pQb <= sum(Qa(1:1)) %Succssful slot (No unbacklogged node is transmitting)
                n=n-1; %Update backlog of the system (goes from state n to n-1)
                packet_arr(i)=1;%No new arrival
                packet_leav(i)=1;
                succ=succ+1;
            elseif sum(Qa(1:1))< pQb && pQb <=1 %Collision slot (1 or more unbacklogged node is transmitting)
                %Calculate the number of unbacklogged node that just
                %received a new packet
                x=1;
                while x<101
                    if sum(Qa(1:x))>=pQb
                        k=x-1;
                        break;
                    end
                    x = x+1;
                end
                n = n+k;
                packet_arr(i)=k;%k corresponds to the number of new arrivals
                packet_leav(i)=0;
            end

        elseif sum(Qr(1:2)) < pQa && pQa <= 1 %Two or more backlogged node are retransmitting

            if  0 <= pQb && pQb <= sum(Qa(1:1)) %No unbacklogged node is transmitting
                packet_arr(i)=0;
                packet_leav(i)=0;
            elseif sum(Qa(1:1))<pQb && pQb <=1 %One or more unbacklogged is transmitting
                x=1;
                while x<101
                    if sum(Qa(1:x))>=pQb
                        k = x-1;
                        break;
                    end
                    x = x+1;
```

```
                end

                n =n+k;
                packet_arr(i)=k;
                packet_leav(i)=0;
            end
        end
    backlog(i) = n; %Fill the array of the system's
       backlog
   end
end

%Figure 01: Setting up the plotting environment for the
    backlog of the system
figure(1)
plot(slots,backlog);
xlabel('X: Slots')
ylabel('Y: Backlog of the system')
title('Backlog of the system VS. the slot number')

%Figure 02: Setting up the plotting environment for
    packets entering/leaving the system
packets_arrived=1:t;
packets_left=1:t;
%At t(n+1) sum the number of packets arrived/left with
    the packets that
%arrived/left the system at t(n)
for x=1:t
    packets_arrived(x)=sum(packet_arr(1:x));
    packets_left(x)=sum(packet_leav(1:x));
end
figure(2)
plot(slots,packets_arrived)
hold on
plot(slots,packets_left,'r')
grid on
xlabel('X: Slots')
ylabel('Y: Number of packets')
title('Number of Packets entering/leaving the system VS.
    the slot number')
legend('Packets entering','Packets leaving')


%Figure 03: Setting up the plotting environment for the
    histogram of the system
figure(3)
```

```
M = max(backlog);
hist(backlog,M) %Array with the counts of the times in each state
nelements = hist(backlog,100); %Count how many times element x was seen
for x = 1:m
    state_probs(x) = nelements(x)/1000;
end
xlabel('X: Number of state n')
ylabel('Y: Frequency of each state')
title('Histogram of the backlog')

%Figure 04: Setting up the plotting environment for the attempt rate
%Calculation of attempt rate plot (theoretical value):
for z = 1:1000
    Att_Rate(z) = q_a*(m-backlog(z))*q_a + q_r*(backlog(z));
end
figure(4)
plot(slots,Att_Rate)
xlabel('X: Slots')
ylabel('Y: Attempt rate G(n)')
title('Attempt rate G(n)')


%%Compare frequency of success to the theoretical probability of sucess
%1. Calculation of average probability of success derived from simulation
Ps_sim = succ/t;
%2. Calculation of average theoretical probability of success
Ps_theor = 0;
for i = 1:m
    %tmp: Qr(1,n)*Qa(0,n)+Qa(1,0)*Qr(0,n)
    tmp = (binopdf(1,m-(i-1),q_a)*binopdf(0,(i-1),q_r))+(binopdf(0,m-(i-1),q_a)*binopdf(1,i-1,q_r));
    %tmp2 : tmp*the probability that this state n happens
    tmp2 = tmp * state_probs(i);
    Ps_theor = Ps_theor + tmp2;%Update Ps_theor
end

%Display values of simulation/theoretical probability of success
```

```
fprintf('Simulated Probability of Success is %f \n', Ps_sim);
fprintf('Theoretical Probability of Success is %f \n', Ps_theor);
```

B Stabilized Slotted ALOHA

```
%Implementation of Pseudo Bayesian stabilization for Slotted ALOHA

m=100;%Total number of nodes
n=0;%Real backlog
backlog=zeros(size(1:1000));%Backlog array
n_estimated=0;%Estimated backlog
backlog_estimate=zeros(size(1:1000));%Estimated backlog array
node_status=zeros(size(1:100));%Status of nodes (1 stands for backlogged/0 for unbacklogged)
Pr = zeros(size(1:20));%Probability of k packets arriving in a node at a given time slot
packets_arrival=zeros(size(1:1000));%Number of packets arriving at a time slot
packets_leaving=zeros(size(1:1000));%Number of packets leaving at a time slot
lambda=1/exp(1);%Arrival rate

%Simulation of up to 20 packets arrival
for j = 0:21
  %Poisson arrival of packets at a node
  Pr(j+1) = poisspdf(j, lambda/m);
end

for t = 1:1000

    transmit = 0;%Temporary variable to see how many packets are there in the system

    %Part 01: Figure our which nodes are backlogged
    count=0;
    for j = 1:100 %Loop over all nodes
        a=rand(1);%Random realization of Pr (probability that k packets arrived at node j)
        if node_status(j) == 0 %Unbacklogged node
            if 0 <= a && a <= Pr(1)% No packet arrival: Pr(x <= 0)
                %Nothing changes
```

```
            elseif a > sum(Pr(1:1))%More than one arrival
                node_status(j)=1; %1. Node becomes backlogged
                n=n+1;%2. Update the backlog of the system
                count=count+1;
            end
        else %Backlogged nodes
            %Nothing changes
        end
    end
    %Update number of packets that entered the system at this time slot
    packets_arrival(t) = count;

    %At this stage we know which nodes are backlogged

    %Calculate qr
    if n_estimated >= 0 && n_estimated < 1
        q_r = 1;
    else
        q_r = 1/n_estimated;
    end

    for j = 1:100 %Loop over all nodes and test the backlogged nodes
        if node_status(j) == 1 %Backlogged node (has a packet)
            b=rand(1);%Random outcome for each backlogged node
            if b <= q_r
                transmit = transmit +1;
            end
        end
    end

    if transmit == 0 %Idle slot
        n_estimated=max(lambda, n_estimated + lambda - 1);%Based on feedback 0
        packets_leaving(t) = 0;
    elseif transmit == 1 %Successful slot
        n = n-1;%Backlog decreases
        n_estimated = max(lambda, n_estimated + lambda - 1);%Based on 1 feedback
        packets_leaving(t) = 1;
        %Update node_status : one backlogged node has to
```

```
            become unbacklogged
        for x =1:100
            if node_status(x) == 1
                node_status(x)=0;%Free one backlogged to become unbacklogged
                break;
            end
        end
    else %Collision slot
        n_estimated = n_estimated + lambda + (exp(1)-2)^-1;%Based on e feedback
        packets_leaving(t) = 0;
    end

    %Save backlog results at time t
    backlog(t) = n;
    backlog_estimate(t) = n_estimated;
end

slots=1:1000;
figure(1) %Setting up the plotting environment for the backlog of the system
xlabel('Slot number, n')
ylabel('Backlogged packets')
plot(slots, backlog)
hold on
plot(slots,backlog_estimate,'g')
title('Real backlog vs. Pseudo Bayesian Estimated backlog')
legend('Backlog', 'Estimated backlog')



%Figure 02: Setting up the plotting environment for the packets entering/leaving the system
packets_arrived=1:1000;%Count total number of packets entering from the begining
packets_left=1:1000;%Count total number of packets leaving from the begining

for x=1:1000
    packets_arrived(x)= sum(packets_arrival(1:x));
    packets_left(x)= sum(packets_leaving(1:x));
end
figure(2)
plot(slots,packets_arrived)
```

```
hold on
plot(slots,packets_left,'k')
grid on
xlabel('X: Slots')
ylabel('Y: Number of packets')
title('Number of packets entering/leaving the system VS.
   the slot number')
legend('Packets entering', 'Packets leaving')
```

C Stabilized Slotted ALOHA, with different values of λ

```
W = zeros(size(1:7));%Pseudo Bayesian approximation delay
    (Theoretical)
D = zeros(size(1:7));%Average simulated delay (Real)
index = 1;%This variable will be used to iterate through
   W and D arrays

for lambda=0.05:0.05:0.35

    m=100;%Total number of nodes
    n=0;%Real backlog
    backlog=zeros(size(1:1000));%Real backlog array
    n_estimated=0;%Estimated backlog
    backlog_estimate=zeros(size(1:1000));%Estimated
        backlog array
    node_status=zeros(size(1:100));%Status of nodes ( 1
        stands for backlogged/ 0 for unbacklogged )
    Pr = zeros(size(1:100));%Probability of k packets
        arriving in a node at a given time slot
    packets_arrival=zeros(size(1:1000));%Number of
        packets arriving at a time slot
    packets_leaving=zeros(size(1:1000));%Number of
        packets leaving at a time slot
    track_time=zeros(size(1:100));%Track packet delay
        array
    real_delay=[];%Accumulated real delay

    for j = 0:101
        %Poisson arrival of packets at a node
        Pr(j+1) = poisspdf(j, lambda/m);
    end

    for t = 1:1000

```

```
        transmit=zeros(size(1:100));%Temporary array to see how many nodes are trying to transmit in the system

        %Part 01: Figure our which nodes are backlogged
        count=0;
        for j = 1:100 %Loop over all nodes
            a=rand(1);%Random realization of Pr (probability that k packets arrived at node j)
            if node_status(j) == 0 %Unbacklogged node
                if 0 <= a && a <= Pr(1)% No packet arrival: Pr(x <= 0)
                    %Nothing changes
                elseif a > sum(Pr(1:1))%More than one arrival
                    node_status(j)=1; %1. Node becomes backlogged
                    track_time(j)=1;%Note arrival time
                    n=n+1;%2. Update the backlog of the system
                    count=count+1;
                end
            else %Backlogged nodes
                %Nothing changes
            end
        end
        packets_arrival(t) = count;%Update number of arrivals at this time slot

        %Calculate qr
        if n_estimated >= 0 && n_estimated < 1
           q_r = 1;
        else
           q_r = 1/n_estimated;
        end

        for j = 1:100 %Loop over all nodes and test the backlogged nodes
           if node_status(j) == 1 %Backlogged node (has a packet)
               b=rand(1);%Random outcome for each backlogged node
               if b <= q_r
```

```
                        transmit(j) = 1;%This node is transmitting
                    else%The node is not transmitting
                        track_time(j)=track_time(j)+1;%Increment the delay time of this node (packet)
                    end
                end
            end

            if sum(transmit(1:100)) == 0 %Idle slot
                n_estimated=max(lambda, n_estimated + lambda - 1);%Based on feedback 0
                packets_leaving(t) = 0;
            elseif sum(transmit) == 1 %Successful slot
                n = n-1;%n decreases
                n_estimated = max(lambda, n_estimated + lambda - 1);%Based on feedback 1
                packets_leaving(t) = 1;
                %Update node_status : one backlogged node has to become unbacklogged
                for x =1:100
                    if transmit(x) == 1
                        node_status(x)=0;%Free one node to become unbacklogged
                        real_delay= [real_delay, track_time(x)];
                        track_time(x)=0;
                        break;
                    end
                end
            else %Collision slot
                n_estimated = n_estimated + lambda + (exp(1)-2)^-1;%Based on feedback e
                packets_leaving(t) = 0;
                track_time = track_time + node_status;
            end

            %Save backlog results at time t
            backlog(t) = n;
            backlog_estimate(t) = n_estimated;

    end

    packets_arrived=1:1000;%Total packets arrived from the begining
```

```
    packets_left=1:1000;%Total packets leaving from the
        begining
    for x=1:1000
            packets_arrived(x)= sum(packets_arrival(1:x))
                ;
            packets_left(x)= sum(packets_leaving(1:x));
    end

    D(index) = mean(real_delay);
    W(index) = ((exp(1)-0.5)/(1-lambda*exp(1)))-(((exp(1)
        -1)*((exp(1)^lambda)-1))/(lambda*(1-((exp(1)-1)*((
        exp(1)^lambda)-1)))));
    index = index + 1;
end


lambdaArray = [0.05 0.1 0.15 0.2 0.25 0.3 0.35];
figure(3)
plot(lambdaArray,D)
hold on
plot(lambdaArray,W,'r')
grid on
xlabel('X: Lambda')
ylabel('Y: Average Delay')
title('Average delay using P-B approximation formula and
    simulation results')
legend('Average simulated delay','P-B Approximate Delay')
```