Project 5: Switch Performance including Buffering

EE 511 – Section: Tuesday 5 pm

Name: Darshan Patil

Student ID: 9575227834

1.

I. Problem Statement

As discussed in class, you are study the operation of an $N\ x\ N$ switch with less than saturated traffic flows. This will build on project 4 and include monitoring the buffers or queues on the input side. The traffic pattern in terms of desired output ports will remain the same as in the last project, but now you will also simulate the arrival of packets to the input ports. A new packet will arrive to an input port in a slot with probability $P_{arrival}$. The packet arrival probability is the same for each input port (to keep the model manageable in terms of complexity). Each input port has a HOL slot and an additional number of buffers to store arriving traffic. Buffers are not shared between ports. If a packet arrives to find all the buffers for that input queue occupied, the packet is dropped – the number of dropped packets should be recorded and reported. To avoid excessive buffer overflows (drops) the rate of packet arrival should not exceed the maximum throughput capability of the switch (which you determined in Project 4). I suggest using 10 buffers per input port, but this should be a parameter in your simulation. You may want to try at least one run with a larger number of buffers (perhaps 50 per input) under a fairly heavily load scenario (perhaps 90% or 95% of the maximum through that the switch can support). It is not necessary to simulate the packets in the buffers, rather it is sufficient to just keep a count of the number of packets in the queue; the destination for a packet can be determined when the packet moves into the HOL slot.

Focus primarily on an 8x8 switch and explore the impact of traffic patterns as in the last project.

Simulate for: 1) Balanced Traffic ($\alpha_j = \frac{1}{N} \quad \forall j$

2) Hot-spot traffic ($\alpha_1 = \frac{1}{k},\ \alpha_j = \left(\frac{1}{N-1}\right)\left(\frac{k-1}{k}\right)\ for\ j \neq 1$

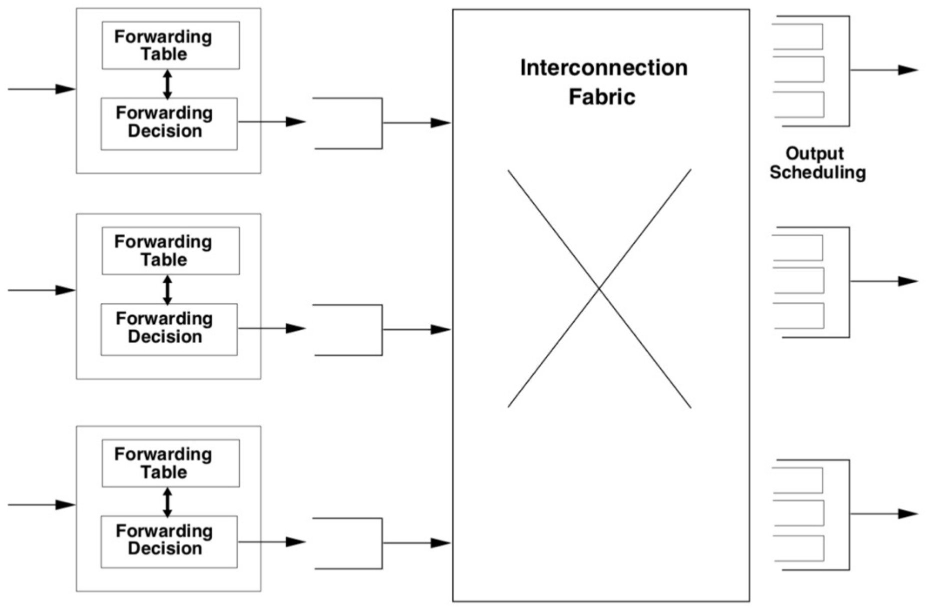
Look at the cases k = 2, 3, and 8 (the case of k=N=8, reverts to being balanced traffic.)

More interesting in this simulation is to determine the queueing statistics of the traffic flows. You should monitor the buffer occupancy for each input on a slot by slot basis so that you can determine the "steady-state" queue size distribution and thus the mean queue length. You can then use Little's result $N = \lambda T$; where N is the average number in a system, λ is the arrival rate in packets per unit time (micro-seconds) , and T is the time an average packet spends in the system in micro-seconds. It is fairly straightforward to calculate mean times for the input queues, estimating delays based on output port requires a little more thought.

You should also monitor packet drops and the number of packet delays while in the HOL slot due to HOL output port blocking for each output port. By splitting the delays in the input queue (until reaching the HOL position) and estimating the delay due to HOL blocking, you can estimate the overall average delay for packets destined to each output port. As a check compare input and output queue statistics.

II. Theoretical Exploration

N x N Switching



- Packets arrive at the input. The decision to route is made. Switching is done. Output scheduling follows. Packets are sent out.
- Head-of-line blocking limits switch performance to 62%
- Consider an N × N input-queued switch
  - time is slotted, so that at most one packet can arrive (depart) per time slot
  - packets arrive at each input with probability $P_{arrival}$, independently across inputs/time
  - the destination of a packet is equally likely to be one of the outputs and independent across all packet
  - the "load matrix" $\{\lambda_{ij}\}$ equals {p} for every i and j
- The scheduling policy: At each time an output chooses one HOL packet.

Little's Theorem

Little's Theorem states that — for a given arrival rate, the time in the system is proportional to packet occupancy.

$N = \lambda T$

where: N = average number of packets in the system,

λ = average packet arrival rate (packets per unit time),

T = average delay (time spent) per packet.

Let Q = Average time spent waiting in queue, T = Average packet delay (transmission plus queuing)

Note that: $T = 1/\mu + Q$

Also, by Little's law: $N = \lambda T$ and $N_q = \lambda Q$

Where N_q = Average number waiting in queue

Queuing Theory

Queuing theory is the mathematical study of the congestion and delays of waiting in line. Queuing theory examines every component of waiting in line to be served, including the arrival process, service process, number of servers, number of system places and the number of "customers" (which might be people, data packets, cars, etc.). As a branch of operations research, queuing theory can help users make informed business decisions on how to build efficient and cost-effective workflow systems. Real-life applications of queuing theory cover a wide range of applications, such as how to provide faster customer service, improve traffic flow, efficiently ship orders from a warehouse and the design of telecommunications systems, from data networks to call centers.

Queues happen when resources are limited. In fact, queues make economic sense; no queues would equate to costly overcapacity. Queuing theory helps in the design of balanced systems that serve customers quickly and efficiently but do not cost too much to be sustainable. All queuing systems are broken down into the entities queuing for an activity. By applying queuing theory, a business can develop more efficient queuing systems, processes, pricing mechanisms, staffing solutions and arrival management strategies to reduce customer wait times and increase the number of customers that can be served.

III. Results

Theoretical Result for N=8

$$throughput = 4.947$$

$$Overall\ switch\ performance = \frac{4.94}{8} * 100 = 61.74\%$$

Experimental Results for N=8, slots=1000,
Balanced Traffic

P_arrival	Throughput	rho (ρ)	avg. Lambda Input	avg. Lambda Output	avg. Arrival Rate	avg. Drop Rate
0.0	0.0000	0	0.0000	0	0.0000	0.0000
0.1	0.7800	0.0925	0.0989	0.0945	0.0097	0.0040
0.2	1.5010	0.1702	0.1993	0.1821	0.2007	0.0174
0.3	2.0700	0.2618	0.2959	0.2643	0.3010	0.0343
0.4	2.7500	0.3294	0.4032	0.3433	0.4022	0.0610
0.5	3.2100	0.4002	0.5045	0.4141	0.5015	0.0968
0.6	3.7000	0.4523	0.5968	0.4725	0.6052	0.1261
0.7	4.1540	0.5103	0.7027	0.5363	0.7033	0.1659
0.8	4.5670	0.5495	0.7995	0.5853	0.8041	0.2159
0.9	4.9050	0.6043	0.8981	0.6332	0.9007	0.2636
1.0	4.9970	0.63175	1.0000	0.6826	1.0020	0.3170

Average Number of Packets in each Buffer,

	Avg. Number of Packets in each Buffer	
Port Number	if Buffer Limit = 10	if Buffer Limit = 50
1	4.6750	24.5090
2	4.3160	22.2760
3	3.7800	22.1480
4	3.6070	20.0530
5	4.9550	17.2030
6	3.7790	19.6550
7	3.7570	15.6080
8	4.2140	13.2970

Balanced Traffic,

Port Number	Input Queueing Delay in µs	Input HOL Delay in µs	Output Queueing Delay in µs	Output HOL Delay in µs
1	5.3060	2.2510	1.9613	2.5320
2	4.9630	2.2250	1.7349	2.3210
3	4.5840	2.5410	1.8214	2.6520
4	3.9350	2.7850	1.5329	2.4540
5	3.6590	2.3740	1.5189	2.6550
6	2.9310	2.9140	1.3048	2.3120
7	2.4160	2.4650	1.1877	2.7890
8	1.4620	2.5560	1.1338	2.4440

Hotspot Traffic,

$$\alpha_1 = \frac{1}{k}, \alpha_j = \left(\frac{1}{N-1}\right)\left(\frac{k-1}{k}\right) \; for \; j \neq 1$$

results for K=2,3, and 8 is calculated for P_arrival = 0.9, and Buffer_Length_Limit=10

K=2,

Port Number	Input Queueing Delay in µs	Input HOL Delay in µs	Output Queueing Delay in µs	Output HOL Delay in µs
1	2.3573	0.9890	1.0806	0.9740
2	2.9841	0.6540	1.2038	0.9340
3	3.2011	0.8450	1.2170	0.9480
4	3.7464	0.4660	1.3713	0.9650
5	3.8734	0.8790	1.4321	0.8890
6	4.1161	0.7450	1.3827	0.7540
7	4.1057	0.7450	1.4246	0.9460
8	4.1150	0.9420	1.3411	0.8550

K=3,

Port Number	Input Queueing Delay in µs	Input HOL Delay in µs	Output Queueing Delay in µs	Output HOL Delay in µs
1	4.0695	1.9890	1.6806	1.6640
2	3.9475	1.6540	1.5694	1.8650
3	4.1054	1.5230	1.6317	1.6780
4	3.7525	1.6650	1.5806	1.8790
5	3.5228	1.6870	1.5050	1.6660
6	3.0802	1.8530	1.3175	1.9540
7	2.7817	1.4560	1.4151	1.4520
8	2.3278	1.3250	1.2094	1.6320

K=8,

Port Number	Input Queueing Delay in µs	Input HOL Delay in µs	Output Queueing Delay in µs	Output HOL Delay in µs
1	5.0542	2.4510	1.9790	2.2720
2	4.3667	2.2450	1.7135	2.3110
3	4.3018	2.3410	1.6517	2.6530
4	3.6000	2.9850	1.4780	2.4520
5	3.3703	2.3740	1.3568	2.6750
6	2.9410	3.2140	1.3694	2.3720
7	2.8895	2.9650	1.1976	2.7290
8	2.3556	2.4560	1.0574	2.4240

Observation

- We can observe the experimental obtained throughput value is approximately same as theoretical value for P_arrival=1
- Throughput decreases as the P_arrival decreases.
- Link Utilization is maximum (i.e.63% for 8x8 switch) for P_arrival=1. Link Utilization decreases as the Arrival rate decreases.
- Drop rate is maximum when the Arrival rate is maximum i.e. because the packet Traffic in switch is maximum when the arrival rate is maxium thus there is high probability that the buffer may overflow and packets to be dropped.
- Average Number of packets in queue is calculated for buffer limit=10 and buffer limit=50.
- Input/Output queueing and HOL delay is calculated for Balanced Traffic.
- For Hotspot Traffic, Input/Output Queueing and HOL delay is calculated for P_arrival=0.9 and Buffer limit=10 for k=2,3, and 8. Similar computations can be made for different P_arrival and buffer limit.
- In Hotspot Traffic, when k=8 we can observe that it is same as Balanced Traffic. Hence, we can observe similar values.
- If we increase the buffer limit, we can observe the increase in queuing delay.

IV. Reference

- https://web.stanford.edu/class/ee384x/EE384X//handouts/handout9.pdf
- http://www.columbia.edu/~ks20/stochastic-I/stochastic-I-MCII.pdf
- Lecture Notes by professor Silvester

V. Source Code

```
%Start of main program
clc;clear all;    % clear window and workspace
N=8;              % N*N Switch
n=1000;           % n-->n_slots
Trials=1;         % Trials
loop=100;         %
S=zeros(N,n);     % Generating Random number and store which port wants to go to
which output
buf=zeros(N,n);      % Buffer
pps = zeros(1,n);                       % packets per slot matrix
alpha=zeros(1,N);                       % alpha values matrix
Mean_Buffer_Length=zeros(N,Trials);     % Mean Buffer Queue Length for each port
Mean_bf=zeros(1,Trials);
Throughput=zeros(1,Trials);             % Throughput Matrix
p_arr=0.2;                                % P_arrival probability
buffer_limit=10;                        % Buffer Limit
AR=zeros(N,1);                          % Arrival Rate Matrix
DR=zeros(N,1);  % Drop Rate Matrix
asum=0;
kh=8;  % Enter K=2,3 and 8
choice=1;       % choice=1 for balanced switch | choice=2 Hotspot Traffic
rho=0;          % calculate rho value

%if choice =1 for Balanced Switch, choice=2 Hotspot Traffic
if(choice==1)
    for i=1:N
        alpha(i)=asum+1/N;
        asum=alpha(i);
    end
else
    alpha(1)=1/kh;
    asum=alpha(1);
    for i=2:N
        alpha(i)=asum+(1/(N-1))*((kh-1)/kh);
        asum=alpha(i);
    end
end

for l=1:Trials
    buf=zeros(N,n);
%     pps=zeros(1,n);
    S=zeros(N,n);
    pps = zeros(1,n);
    P1=rand(N,1);
    P2=zeros(N,1);
    S(:,i)=zeros(N,1);

    for j=1:n
        P1=rand(N,1);
        for i=1:N
            if(P1(i)<p_arr)
```

```
                    S(i,j)=randi([1,N],1,1);
                end
            end
        end

        for qw=1:loop
        for j=1:n
            X=zeros(N,1);
            Y=zeros(size(X));
            % Ya=Y;
            X=S(:,j);
            Z=(unique(X));           %   calculate the unique values in X
            %    W=zeros(length(Z),2);

            for i=1:length(Z)
                Y(i,1)=sum(X==Z(i,1));      % calculate the repeated Random numbers
                if(Z(i,1)==0)
                    Y(i,1)=0;
                end
            end

            Ya=Y(1:length(Z),1);
            W=[Z,Ya];
            pps(j)=sum(Y(:)~=0);
            flag=0;

            for i=1:length(Z)
                [buf]=buffer(i,j,W,X,alpha,buf,N);
            end
        end
        end

    drop=zeros(N,n);
        for j=1:n
             for i=1:N
                 if(buf(i,j)>10)
                    drop(i,j)=buf(i,j)-buffer_limit;
                    buf(i,j)=buffer_limit;
                 end
             end
        end
        bf=buf.';
        for k=1:N
            Mean_Buffer_Length(k,1) = mean(bf(:,k));
        end

        for i=1:N
             aasum=0;
             for j=1:n
                 if(S(i,j)~=0)
                     aasum=aasum+1;
                 end
             end
             AR(i)=aasum/n;
        end

        dest=zeros(N,1);

        for j=1:n
             for i=1:N
                 if(S(i,j)==1)
                     dest(1)=dest(1)+1;
                 elseif(S(i,j)==2)
```

```
                    dest(2)=dest(2)+1;
                elseif(S(i,j)==3)
                    dest(3)=dest(3)+1;
                elseif(S(i,j)==4)
                    dest(4)=dest(4)+1;
                elseif(S(i,j)==5)
                    dest(5)=dest(5)+1;
                elseif(S(i,j)==6)
                    dest(6)=dest(6)+1;
                elseif(S(i,j)==7)
                    dest(7)=dest(7)+1;
                elseif(S(i,j)==8)
                    dest(8)=dest(8)+1;
                end
            end
        end

        for i=1:N
            aasum=0;
            for j=1:n
                if(drop(i,j)~=0)
                    aasum=aasum+1;

                end
            end
            DR(i)=aasum/n;
        end

        Lamda_input=AR;
        Lamda_output=AR-DR;
        Lamda_input_mean=mean(Lamda_input);
        Lamda_Ouput_mean=mean(Lamda_output);
        DRR=mean(DR);
        ARR=mean(AR);
        output_Q_delay=(dest/n)./Lamda_output;
        input_Q_delay=Mean_Buffer_Length./AR;
        input_HOL_delay=inHOL/n;
        output_HOL_delay=outHOL/(n-DP);
        Throughput(l)=sum(pps)/n;
    end

        for m=1:Trials
            Mean_bf(m) = mean(Mean_Buffer_Length(:,m));
        end
    Throughpt=mean(Throughput);
    rho=Throughput/N;
    %End of Main

    %Start of Functions: Buffer, Push , Pop, Cal
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    function [buf]=buffer(i,j,W,X,alpha,buf,N)

            if(W(i,2)==1)
                for k=1:N
                    if(X(k)==W(i,1))
                       [buf(k,j)] = pop(buf(k,j));
                    end
                end
            elseif(W(i,2)==2)
                N1=cal(W(i,1),X,2,N);
                P1 = rand(1);
                b1=N1(1,1);
                b2=N1(2,1);
                if(P1 <= alpha(1))
                    buf(b1,j) = pop(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                else
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = pop(buf(b2,j));
                end
            elseif(W(i,2)==3)
                N1=cal(W(i,1),X,3,N);
                P2 = rand(1)*alpha(3);
                b1=N1(1,1);
                b2=N1(2,1);
                b3=N1(3,1);
                if(P2 <= alpha(1))
                    buf(b1,j) = pop(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j)=push(buf(b3,j));
                elseif(alpha(1)<P2 <= alpha(2))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = pop(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                elseif(alpha(2)<P2 <= alpha(3))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = pop(buf(b3,j));
                end
            elseif(W(i,2)==4)
                N1=cal(W(i,1),X,4,N);
                P3 = rand(1)*alpha(4);
                b1=N1(1,1);
                b2=N1(2,1);
                b3=N1(3,1);
                b4=N1(4,1);
                if(P3 <= alpha(1))
                    buf(b1,j) = pop(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j)=push(buf(b3,j));
                    buf(b4,j)=push(buf(b4,j));
                elseif(alpha(1)<P3 <= alpha(2))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = pop(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                    buf(b4,j) = push(buf(b4,j));
                elseif(alpha(2)<P3 <= alpha(3))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = pop(buf(b3,j));
                    buf(b4,j) = push(buf(b4,j));
                elseif(alpha(3)<P3 <= alpha(4))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                    buf(b4,j) = pop(buf(b4,j));
                end
            elseif(W(i,2)==5)
                N1=cal(W(i,1),X,5,N);
                P3 = rand(1)*alpha(5);
                b1=N1(1,1);
                b2=N1(2,1);
                b3=N1(3,1);
                b4=N1(4,1);
                b5=N1(5,1);
                if(P3 <= alpha(1))
```

```
                buf(b1,j) = pop(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j)=push(buf(b3,j));
                buf(b4,j)=push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
            elseif(alpha(1)<P3 <= alpha(2))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = pop(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
            elseif(alpha(2)<P3 <= alpha(3))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = pop(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
            elseif(alpha(3)<P3 <= alpha(4))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = pop(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
            elseif(alpha(4)<P3 <= alpha(5))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=pop(buf(b5,j));
            end
        elseif(W(i,2)==6)
            N1=cal(W(i,1),X,6,N);
            P3 = rand(1)*alpha(6);
            b1=N1(1,1);
            b2=N1(2,1);
            b3=N1(3,1);
            b4=N1(4,1);
            b5=N1(5,1);
            b6=N1(6,1);
            if(P3 <= alpha(1))
                buf(b1,j) = pop(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j)=push(buf(b3,j));
                buf(b4,j)=push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
            elseif(alpha(1)<P3 <= alpha(2))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = pop(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
            elseif(alpha(2)<P3 <= alpha(3))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = pop(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
            elseif(alpha(3)<P3 <= alpha(4))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = pop(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
            elseif(alpha(4)<P3 <= alpha(5))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j) = pop(buf(b5,j));
                buf(b6,j) = push(buf(b6,j));
            elseif(alpha(5)<P3 <= alpha(6))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j) = push(buf(b5,j));
                buf(b6,j) = pop(buf(b6,j));
            end
        elseif(W(i,2)==7)
            N1=cal(W(i,1),X,7,N);
            P3 = rand(1)*alpha(7);
            b1=N1(1,1);
            b2=N1(2,1);
            b3=N1(3,1);
            b4=N1(4,1);
            b5=N1(5,1);
            b6=N1(6,1);
            b7=N1(7,1);
            if(P3 <= alpha(1))
                buf(b1,j) = pop(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j)=push(buf(b3,j));
                buf(b4,j)=push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
            elseif(alpha(1)<P3 <= alpha(2))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = pop(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
            elseif(alpha(2)<P3 <= alpha(3))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = pop(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
            elseif(alpha(3)<P3 <= alpha(4))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = pop(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
            elseif(alpha(4)<P3 <= alpha(5))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j) = pop(buf(b5,j));
                buf(b6,j) = push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
            elseif(alpha(5)<P3 <= alpha(6))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j) = push(buf(b5,j));
                buf(b6,j) = pop(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
            elseif(alpha(6)<P3 <= alpha(7))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j) = push(buf(b5,j));
                buf(b6,j) = push(buf(b6,j));
                buf(b7,j) = pop(buf(b7,j));
            end
        elseif(W(i,2)==8)
            N1=cal(W(i,1),X,8,N);
            P3 = rand(1)*alpha(8);
            b1=N1(1,1);
            b2=N1(2,1);
            b3=N1(3,1);
            b4=N1(4,1);
            b5=N1(5,1);
            b6=N1(6,1);
            b7=N1(7,1);
            b8=N1(8,1);
            if(P3 <= alpha(1))
                buf(b1,j) = pop(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j)=push(buf(b3,j));
                buf(b4,j)=push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
                buf(b8,j)=push(buf(b8,j));
            elseif(alpha(1)<P3 <= alpha(2))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = pop(buf(b2,j));
                buf(b3,j) = push(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
                buf(b8,j)=push(buf(b8,j));
            elseif(alpha(2)<P3 <= alpha(3))
                buf(b1,j) = push(buf(b1,j));
                buf(b2,j) = push(buf(b2,j));
                buf(b3,j) = pop(buf(b3,j));
                buf(b4,j) = push(buf(b4,j));
                buf(b5,j)=push(buf(b5,j));
                buf(b6,j)=push(buf(b6,j));
                buf(b7,j)=push(buf(b7,j));
                buf(b8,j)=push(buf(b8,j));
            elseif(alpha(3)<P3 <= alpha(4))
                buf(b1,j) = push(buf(b1,j));
```

```
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                    buf(b4,j) = pop(buf(b4,j));
                    buf(b5,j)=push(buf(b5,j));
                    buf(b6,j)=push(buf(b6,j));
                    buf(b7,j)=push(buf(b7,j));
                    buf(b8,j)=push(buf(b8,j));
                elseif(alpha(4)<P3 <= alpha(5))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                    buf(b4,j) = push(buf(b4,j));
                    buf(b5,j) = pop(buf(b5,j));
                    buf(b6,j) = push(buf(b6,j));
                    buf(b7,j)=push(buf(b7,j));
                    buf(b8,j)=push(buf(b8,j));
                elseif(alpha(5)<P3 <= alpha(6))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                    buf(b4,j) = push(buf(b4,j));
                    buf(b5,j) = push(buf(b5,j));
                    buf(b6,j) = pop(buf(b6,j));
                    buf(b7,j)=push(buf(b7,j));
                    buf(b8,j)=push(buf(b8,j));
                elseif(alpha(6)<P3 <= alpha(7))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                    buf(b4,j) = push(buf(b4,j));
                    buf(b5,j) = push(buf(b5,j));
                    buf(b6,j) = push(buf(b6,j));
                    buf(b7,j) = pop(buf(b7,j));
                    buf(b8,j)=push(buf(b8,j));
                elseif(alpha(7)<P3 <= alpha(8))
                    buf(b1,j) = push(buf(b1,j));
                    buf(b2,j) = push(buf(b2,j));
                    buf(b3,j) = push(buf(b3,j));
                    buf(b4,j) = push(buf(b4,j));
                    buf(b5,j) = push(buf(b5,j));
                    buf(b6,j) = push(buf(b6,j));
                    buf(b7,j) = push(buf(b7,j));
                    buf(b8,j) = pop(buf(b8,j));
                end
            else
            end
    end

    %Function to increment queue buffer for specific Port
    function [N1]=cal(v,X,size,N)
        N1=zeros(size,1);
        j=1;
        for k=1:N
          if(X(k)==v)
                N1(j,1)=k;
                j=j+1;
          end
        end
    end
```

```
%Function to decrement buffer queue when packet is sent
%if the number of packets in the buffer not 0 then pop one packet out
function [bf] = pop(bf)
    if(bf ~= 0)
        bf = bf - 1;
    else
        bf = 0;
    end
end

%Function to Increment buffer when packet fails to go to specific output
% increment queue count
function [bf] = push(bf)
    if(bf>=0)
        bf = bf+1;
    end
end
```