Project 5: Switch Performance including Buffering

EE 511 – Section Thursday 9 am

Name: Junquan Yu

Student ID #: 3372029142

1. Problem Statement

As discussed in class, you are study the operation of an N *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 parrival . 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 (a_j=1/N $\forall$j) and

2) Hot-spot traffic a_1=1/k, $a_j=(\frac{1}{N-1})(\frac{k-1}{k})$ for j≠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=λT; 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.

2. Theoretical Exploration or Analysis

Consider the system in state X(t) at time t, where X(t) is a random variable in this problem, we consider discrete time with unit of a single time slot in other word a cycle, and time index is an integer. In order to describe the future of the system state X(t+1), we only need to know the state X(t), which is an important property of Markov chain.

Let X_n be a discrete-time integer-valued Markov chain that starts at n=0 with pmf:

$p_j(0) = P[X_0 = j]$ j=0, 1, 2, …

We will assume X_n that takes on values from a countable set of integers, usually {0, 1, 2, …}. We say that the Markov chain is finite state if takes on values from a finite set.

The joint pmf for the first n+1 values of the process is:

$$P[X_n = i_n, \ldots, X_0 = i_0] = P[X_n = i_n \mid X_{n-1} = i_{n-1}] \ldots P[X_1 = i_1 \mid X_0 = i_0] P[X_0 = i_0]$$

Thus the joint pmf for a particular sequence is simply the product of the probability for the initial state and the probabilities for the subsequent one-step state transitions.

We will assume that the one-step state transition probabilities are fixed and do not change with time, that is,

$P[X_{n+1} = j \mid X_n = i] = p_{ij}$ for all n

X_n is said to have homogeneous transition probabilities. The joint pmf for $X_n, \ldots, X_0$ is then given by:

$$P[X_n = i_n, \ldots, X_0 = i_0] = p_{i_{n-1},i_n} \ldots p_{i_0,i_1} p_{i_0}(0)$$

Thus X_n is completely specified by the initial pmf $p_i(0)$ and the matrix of one-step transition probabilities P:

$$P=\begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdots \\ p_{10} & p_{11} & p_{12} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ p_{i0} & p_{i1} & \cdots & \cdots \end{bmatrix}$$

We will call P the transition probability matrix**,** and that each row of P must add to one since:

$$1 = \sum_j P[X_{n+1} = j \mid X_n = i] = \sum_j p_{ij}$$

If the Markov chain is finite state, then the matrix P will be an n $\times$ n nonnegative square with rows that add up to 1.

A simple N*N blocking switch can only transfer one packet to a specific output in one slot, and other inputs with that output packet will be blocked. If the output selection is not uniform, this phenomenon leads to a significant decrease in switching performance. Due to HOL blocking, it is necessary to set a buffer to store the packets and avoid this situation.

3. Simulation Methodology

I created a M $\times$ N matrix to simulate the buffer and HOL slots. The matrix is intended for storing and forwarding the packets (the character M denotes the number of ports and the character N corresponds to the size of HOL slot).

Then the whole simulation consists of two main parts, which are packets arrival and packets switching. By applying the condition rand<$p_{arrival}$, we can decide whether there is a new packet arriving at each port. If a new one indeed reaches, then we decide the where it is forwarded. At the same time, it is worth noting that the arrived packets would be dropped automatically if the buffer is already full, so I need to count the number of drop at each port. Here comes the next part, the packets begin to be forwarded at HOL slots. After that, I checked the destination of each port to see if there is a conflict or not and forwarded the packets based on the condition. For those ports sending packets, the row of that port in the matrix will be moved left for a step and the final number should be 0 while for those being blocked, the row of it in the matrix remains the same. I used this regulation to keep my matrix updated and simulated for a single time slot. Then I just repeated this process for n times for the final results.

4. Experiments and Results

As I don't expect that the buffer to be always full, it is necessary for me to estimate and calculate the maximum value of $\mathrm{P_{arrival}}$, which is $\mathrm{P_{arrmax}}$. Assuming that $\mathrm{T_{port\_min}}$ denotes the minimum value of the throughput among the 8 output ports and 1 second is divided into 10^6 time slots. From the statistics gathered for the 8 x 8 switch from the Project 4, I can easily know $\mathrm{P_{arrmax}}$ for the different values taken on by k, using the equation that using the equation that $\mathrm{P_{arrmax}}$ =($\mathrm{T_{port\_min}}$)/10^6. For k=8, the $\mathrm{P_{arrmax}}$=6.1716 x $10^5/10^6 \approx$0.61. When $\mathrm{P_{arrival}} \geqslant 0.9\mathrm{P_{arrmax}} \approx 0.55$, the traffic load is rather severe and heavy.

[1] the impact of arrival possibility on the switch performance given certain k and buffer size and the impact of buffer size on the switch performance given certain k and arrival possibility:

According to this, I intend to simulate for three times with three sets of input ($\mathrm{P_{arrival}}$=0.05, k=8, buffer size=15), ($\mathrm{P_{arrival}}$=0.30, k=8, buffer size=15), ($\mathrm{P_{arrival}}$=0.55, k=8, buffer size=15), which corresponds to scenarios with less input traffic, medium input traffic and heavy input traffic. The results are as follows:

a) $P_{arrival}$=0.15, k=8, buffer size=10:

	Throughput(pps)	$Delay_{queue}$(μs)	$Delay_{HOL}$(μs)	Number of dropped packets
Port1	149967	2.107×10^{-3}	0.076	0
Port2	149832	1.44×10^{-3}	0.077	0
Port3	149454	1.8×10^{-3}	0.076	0
Port4	150536	1.9×10^{-3}	0.077	0
Port5	151267	1.56×10^{-3}	0.076	0
Port6	149977	1.687×10^{-3}	0.076	0
Port7	150338	1.88×10^{-3}	0.077	0
Port8	150112	1.96×10^{-3}	0.075	0

Then I can derive following statistics from the table listed above:

The total output throughput is 1201483 pps.

The mean of the queue delay is 1.792×10^{-3} μs

The mean of the HOL delay is 0.076 μs

The mean of the total delay is 0.078 μs

The total number of dropped packets is 0

b) $P_{arrival}$=0.35, k=8, buffer size=10

	Throughput(pps)	$Delay_{queue}$(μs)	$Delay_{HOL}$(μs)	Number of dropped packets
Port1	350294	0.062	0.236	0
Port2	350264	0.067	0.240	0
Port3	349406	0.065	0.239	0
Port4	350507	0.063	0.238	0
Port5	350571	0.064	0.238	0
Port6	350437	0.067	0.239	0
Port7	349993	0.066	0.240	0
Port8	349520	0.066	0.239	0

Then I can derive following statistics from the table listed above:

The total output throughput is 2800992 pps.

The mean of the queue delay is 0.065 μs

The mean of the HOL delay is 0.239 μs

The mean of the total delay is 0.304 μs

The total number of dropped packets is 0

c) $P_{arrival}$=0.55, k=8, buffer size=10:

	Throughput(pps)	$Delay_{queue}$(μs)	$Delay_{HOL}$(μs)	Number of dropped packets
Port1	548788	1.636	0.601	330
Port2	549748	1.685	0.604	396
Port3	549806	1.671	0.602	384
Port4	549358	1.653	0.601	379
Port5	547596	1.661	0.602	382
Port6	548303	1.689	0.604	465
Port7	547981	1.669	0.603	423
Port8	548053	1.650	0.600	405

Then I can derive following statistics from the table listed above:

The total output throughput is 4389633 pps.

The mean of the queue delay is 1.664 μs

The mean of the HOL delay is 0.602 μs

The mean of the total delay is 2.265 μs

The total number of dropped packets is 3164

It can be seen from statistics listed above that when I chose $P_{arrival} < P_{arrmax}$ and k value and buffer size remain unchanged, the throughput would boost with the $P_{arrival}$ rising. I think it is because the rate of traffic usage increases. What's more, the total delay would tend to be longer since the queue tends to be longer for the relatively higher packets arrival possibility, which results in longer queue in the buffer of the switch. It is obvious that there is almost no packets loss in the buffer with the less or medium traffic loads ($P_{arrival} < 0.9P_{arrmax}$) while the packets loss is much severe with the heavy traffic loads ($P_{arrival} \geq 0.9P_{arrmax}$).

For the purpose of comparison, I just expanded the buff size from 10 to 60 manually and simulated again to see if there are some changes to the results. The results are as follows:

d) $P_{arrival}$=0.55, k=8, buffer size=60:

	Throughput(pps)	$Delay_{queue}$(μs)	$Delay_{HOL}$(μs)	Number of dropped packets
Port1	549499	1.862	0.609	0
Port2	548816	1.869	0.609	0
Port3	549627	1.846	0.606	0
Port4	549983	1.836	0.608	0
Port5	550200	1.887	0.606	0
Port6	550312	1.858	0.608	0
Port7	550145	1.843	0.607	0
Port8	550257	1.806	0.607	0

Then I can derive following statistics from the table listed above:

The total output throughput is 4398839 pps.

The mean of the queue delay is 1.851 μs

The mean of the HOL delay is 0.608 μs

The mean of the total delay is 2.458 μs

The total number of dropped packets is 0

It can be seen from the above table that the total number of dropped packets decreases back from 3164 to 0 while the queuing delay becomes larger after I adjusted the buffer size from 10 to 60. With the expansion of the buffer size, the packets are more likely to queue in the buffer, instead of being dropped under the heavy traffic condition. Therefore, the delay derived from the scenario with buffer size=60 is the queuing delay for the ideal situation, which is with no packets loss.

[2] The impact of k on the switch performance given certain arrival possibility and buffer size:

I chose $P_{arrival}$=0.3 and buffer size=10, and ran the program several times for different values of k. The results are as follows:

a) The throughput (pps):

Throughput\k	2	3	4	5	6	7	8
The total throughput	1599252	1600930	1600540	1601046	1600014	1599413	1599432
The throughput for port 1	798801	534635	400346	320098	267119	228324	200479
The throughput for port 2	114187	151877	171815	183814	190030	195550	199272
The throughput for port 3	114673	152028	171380	183665	190085	196630	199927
The throughput for port 4	114357	152273	170588	182707	190278	197053	199874
The throughput for port 5	114388	152450	171119	182965	190262	195308	200148
The throughput for port 6	114430	152622	171439	182502	190488	194815	200538
The throughput for port 7	114444	151899	172041	183128	190910	195572	199416
The throughput for port 8	113972	153146	171812	182167	190842	196161	199778

b) The queuing delay (μs):

Delay\k	2	3	4	5	6	7	8
The mean of delay	0.408	0.032	0.011	0.007	0.006	0.005	0.005
The delay for port 1	0.412	0.038	0.012	0.008	0.006	0.006	0.006
The delay for port 2	0.413	0.031	0.012	0.007	0.005	0.006	0.005

The delay for port 3	0.397	0.033	0.011	0.007	0.006	0.005	0.006
The delay for port 4	0.402	0.032	0.011	0.007	0.006	0.005	0.005
The delay for port 5	0.401	0.032	0.011	0.007	0.006	0.006	0.005
The delay for port 6	0.403	0.031	0.011	0.006	0.006	0.005	0.005
The delay for port 7	0.421	0.033	0.011	0.007	0.006	0.005	0.006
The delay for port 8	0.416	0.0332	0.011	0.008	0.006	0.005	0.006

c) The HOL delay (μs):

Delay\k	2	3	4	5	6	7	8
The mean of delay	0.639	0.202	0.136	0.118	0.111	0.109	0.108
The delay for port 1	0.642	0.203	0.137	0.118	0.112	0.108	0.109
The delay for port 2	0.642	0.202	0.136	0.117	0.112	0.109	0.109
The delay for port 3	0.638	0.203	0.137	0.118	0.111	0.109	0.109
The delay for port 4	0.636	0.203	0.136	0.118	0.111	0.109	0.108
The delay for port 5	0.636	0.203	0.137	0.119	0.110	0.110	0.108
The delay for port 6	0.638	0.202	0.135	0.117	0.112	0.108	0.108
The delay for port 7	0.643	0.203	0.138	0.119	0.111	0.109	0.108
The delay for port 8	0.637	0.200	0.137	0.117	0.111	0.107	0.109

d) The total delay (μs):

Delay\k	2	3	4	5	6	7	8
The delay	1.047	0.235	0.148	0.125	0.117	0.114	0.114

e) The number of the dropped packets:

Number\k	2	3	4	5	6	7	8
The total num	20	0	0	0	0	0	0
The num for port 1	0	0	0	0	0	0	0
The num for port 2	0	0	0	0	0	0	0
The num for port 3	5	0	0	0	0	0	0
The num for port 4	5	0	0	0	0	0	0
The num for port 5	2	0	0	0	0	0	0
The num for port 6	6	0	0	0	0	0	0
The num for port 7	2	0	0	0	0	0	0
The num for port 8	0	0	0	0	0	0	0

It can be seen from the above five tables that for the certain k, and ports 2-8 have the almost same throughput while the throughput for port 1 is the largest one. With the value of k rising, the throughput for port 1 decreases and the throughput of ports 2-8 increase until the throughput of all ports is the same when the scenario for balanced traffic occurs. However, what different is that here the overall throughput remains the same in different circumstances. This is because the dominant factor of overall throughput here is the usage of the traffic. Here we have $P_{arrival}$ =0.2, which means that the load is not heavy, so the usage of traffic is not high. As a result, it limits the overall throughput, unlike in Project 4 it is the conflicts that limits the overall throughput. In terms of delay, with the increase of k, both queuing delay and HOL delay are reduced, especially from k=2~5, resulting in the total delay's decrease. I think this is because more balanced the traffic is, less conflict will occur, resulting in shorter queue and less HOL block. For number of dropped packets, when

$P_{arrival}$ =0.2, the traffic load is not so heavy, so packets loss is least likely to happen.

5. References

1. *Alberto Leon-Garcia. (2008). Probability, Statistics, and Random Processes for Electrical Engineering. Upper Saddle River, NJ 07458. Pearson Education, Inc.*

2. *Zhou Sheng, Shiqian Xie, Chengyi Pan. (2008). Probability Theory and Mathematical Statistics. No.4, Dewai Street, Xicheng District, Beijing. Higher Education Press.*

6. Source Code

```
p_arrival=input('Enter the probability that a new packet arrival p=');    %enter the arrival probability
k=input('Enter k=');
buffer_size=input('Enter buffer size=');
N=8;
n=10^6;
buffer(N,buffer_size+1)=zeros;

a=zeros(1,N);
a_cum=zeros(1  ,N);
num_drop=zeros(N,1);
outputport=zeros(N,1);
length_buffer=zeros(N,1);
HOL_block=zeros(N,1);
load=zeros(N,1);

a(1)=1/k;    %the probability of a new packet reaching each output port
for j=2:N
   a(j)=(1/(N-1))*((k-1)/k);
end
a_cum(1)=a(1);     %create probability array
for j=2:N
   a_cum(j)=a(j)+ a_cum(j-1);
end
a_cum;
```

```
for k=1:n

for i=1:N
    FIND=find(buffer(i,:)==0);
    if rand<p_arrival
        load(i,1)=load(i,1)+1;
        if ~isempty(FIND)
            xr=rand;   %decide the destination of new arrived packet
            m=1;
            while xr>a_cum(m)
                m=m+1;
            end
        buffer(i,FIND(1))=m;

        else     %compute the packet drop
            num_drop(i,1)=num_drop(i,1)+1;
        end
    end
end


U=unique(buffer(:,1));   %forward packets in input port
for i=1:length(U)
    if U(i)==0
        continue;
    else
        FIND=find(buffer(:,1)==U(i));
        if length(FIND)==1;
            outputport(U(i),1)=outputport(U(i),1)+1;
            for j=1:buffer_size
                buffer(FIND(1,1),j)=buffer(FIND(1,1),j+1);
                buffer(FIND(1,1),buffer_size+1)=0;
            end
        else
            outputport(U(i),1)=outputport(U(i),1)+1;
            inputport_rank=randperm(length(FIND));  %decide which port to send when there is a
conflict
            num_chos=inputport_rank(1,1);
            FIND(num_chos,1);

            for j=1:buffer_size   %the packets are switched and forwarded
```

```
            buffer(FIND(num_chos,1),j)=buffer(FIND(num_chos,1),j+1);
            buffer(FIND(num_chos,1),buffer_size+1)=0;
         end

         for j=1:length(FIND)-1      %compute HOL blocks
            num_block=FIND(inputport_rank(1,j+1),1);
            HOL_block(num_block,1)=HOL_block(num_block,1)+1;
         end
end
   end
end

for i=1:N        %compute length of buffer occupied in each time slot
   for j=2:buffer_size+1
      if buffer(i,j)~=0
         length_buffer(i,1)=length_buffer(i,1)+1;
      end
   end
end

end

delay_portqueue=vpa((length_buffer/n)/p_arrival)   %compute throughput and delay
HOL_block=vpa((HOL_block/n)/p_arrival)
delay_queue=vpa(mean(delay_portqueue))
delay_HOL=vpa(mean(HOL_block))
delay_total=vpa(delay_queue+delay_HOL)
num_drop
outputport
num_drop=sum(num_drop)
throughput=sum(outputport)
```