

Throughput of Internally Buffered Crossbar Switch

Mingjie Lin (EE384Y Final Report)

Thursday, May 15 2003

Abstract

Internally Buffered Crossbar Switch(IBCS) is studied analytically to obtain insight of throughput performance of this particular type of switching fabric under i.i.d. uniformly distributed Bernoulli traffic. Both analytical and simulation studies are performed. Simulation results are utilized to validate the analytical models employed and demonstrate a correct match with the analytical solution. The major finding is that when switch size N goes to infinity, the throughput of IBCS converges to 100% in both blocking and non-blocking cases, which sharply contrasts with the well known throughput results of Crossbar Switch without internal buffer [1](58% and 63% for blocking and non-blocking cases, respectively, when $N \rightarrow \infty$). Another interesting observation is that for crossbar switch without crosspoint buffer, the saturation throughput decreases with increasing switch size N , whereas for crossbar switch with internal buffer, the opposite is true. We believe, the IBCS architecture will potentially be valuable especially when memory access time continuously seems to be the performance bottleneck of the high speed packet switches, whereas in the meanwhile, the high capacity memory chip is becoming cheaper and more accessible.

1 Introduction

Crossbar switching fabrics are very popular for switching implementation because of their non-blocking capability, simplicity, and their marketing availability. In an Output-Queued(OQ) switch, the possible fan-in of N input ports to a single output port requires a memory bandwidth equal to N times link speed if packet loss is to be prevented. In an Input-Queued(IQ) switch with no speedup, the memory bandwidth only needs to be sufficient to support the data rate of a single link, the switch scalability is limited only by switch scheduling delay.

Head of Line(HOL) blocking is a well-known problem for a crossbar with FIFOs at the input [1]. This problem is overcome by using Virtual Output Queues(VOQ) [2] [3] [4]. For a crossbar switch with VOQs, maximum matching algorithms have been proposed to achieve 100% throughput. Maximum

matching algorithms are sufficient but with such a high complexity [5] that implementation is infeasible for high-speed systems. There are also many variation systems, such as Longest Port Queueing(LPQ), Oldest Cell First(OCF), and Longest Port First(LPF) [6]. Many maximal matching schemes have also been considered as an alternative to maximum matching algorithms, among them, *i*SLIP [3], Dual Round-Robin Matching(DRRM)[7], and Longest Output Occupancy First Algorithm(LOOFA)[8].

For a long time, Internally Buffered Crossbar Switch(IBCS) have been considered as a solution to improve switching throughput instead of traditional crossbar switches without internal buffer. It is true that the number of buffer memory will grow in the same order as the number of crosspoints ($O(N^2)$, where N is the number of input/output ports), but with the current billion transistor technology, the memory capacity demand seems to pose no significant problem. As a matter of fact, it seems that the memory bandwidth demand is continuously to be the performance bottleneck of high-performance switches. In this sense, IBCS can potentially become a valuable alternative switching fabric technology.

There are several studies about performance aspects of IBCS, most of them are based on simulation results. Nabeshima [9] uses Oldest Cell First (OCF) scheduling for each input and output scheduler. This system is shown to have very high throughput by simulation, if the input traffic follows i.i.d. uniformly distributed Bernoulli arrival. Tara, etc. [10] extended Nabeshima's work, he simplifies the scheduling process by using RR schedulers at the outputs and then uses fluid model techniques to prove that 100% throughput is achieved for input traffic that satisfies the strong law of large numbers and that produces a load less than $1/N$ for any input/output pair of $N \times N$ switching fabric, which is a subset of admissible traffic load. Furthermore, his simulation results suggest that his scheduling algorithm for buffered crossbar with one cell may achieve 100% throughput for much larger class of admissible loads. More recently, a Combined Input-One-Cell-Crosspoint Buffer crossbar (CIXB-1) with VOQs at the inputs has been proposed [11]. It is shown that the proposed architecture can provide 100% throughput under uniform traffic.

The objective of the present study has three parts (the first part is finished, and part 2 and part 3 are ongoing): (1)To analyze the saturation throughput of the Internally Buffered Crossbar Switch(IBCS) in both blocking and non-blocking situations using analytical model for i.i.d. uniformly distributed Bernoulli traffic. (2) For IBCS fabric with VOQ(Virtual Output Queue), we want to investigate the minimum speedup required to achieve 100% throughput for any admissible traffic. Obviously, if complicated scheduling algorithm, such as MWM etc., is allowed to be employed, 100% throughput can be easily obtained even with no speedup. What we are interested in is to find out the minimum speedup when using simple randomized scheduling algorithm. Sundar, Da, and McKeown [?] have proved speedup 2 is sufficient for simple randomized

algorithm to achieve 100% . We suspect that a lower bound can be found. (3) Also, for IBCS fabric with VOQ(Virtual Output Queue), we want to find out the potential benefit of having more than 1 buffer cell at each crosspoint, especially in the aspect of emulating OQ(output queuing). It has been proved that speedup 2 is sufficient for VOQ crossbar switch to fully emulate an OQ crossbar switch. Our observation is that internal buffer cells are functionally similar to the output queueing buffer, although it is not as flexible and efficient as pure output queueing buffer. Our conjecture is that if we can have certain constant size of internal buffer cells (such as 4) at each crosspoint, by employing certain simple scheduling algorithm, we may achieve full emulation with speedup 1 to an OQ crossbar switch.

2 Buffered Crossbar Switch

Consider a $N \times N$ buffered crossbar fabric with a one-cell buffer at each crosspoint. Assume time is slotted and that packets arrive according to i.i.d uniform Bernoulli random process and exactly at the beginning of a time slot. For concreteness the time slot n corresponds to the time interval $[n - 1, n)$, $n = 1, 2, 3, \dots$. Each input port i has an input queue(IQ), which has infinite capacity. Each packet arriving at input port i will have a destination port number j randomly selected from all N output ports according to a uniform distribution.

In each time slot, the switching operation actually consists of two scheduling phases conceptually, named as "Buffer In" and "Buffer out" phases. During the "Buffer In" phase, for a particular input port i , if there is a HOL packet in Q_i with destined output port j , the scheduler will check if crosspoint buffer cell $B_{i,j}$ is empty, if it is, packet i will be immediately removed and put into $B_{i,j}$, otherwise, nothing will be done. The operation will be done in parallel for all input ports during one time slot. During the "Buffer Out" phase, for each output port, the scheduler will look at all crosspoint buffer cells, $B_{i,j}$, where $i = 1, 2, 3, \dots, N$, and then randomly choose one cell uniformly from all occupied crosspoint buffer cells and remove the occupying packet and output it. This operation also will be done in parallel during one time slot. For the sake of the analysis, we assume the packet size is constant and packet transmission from crosspoint buffer cell happens only at the end of the time slot. Figure 1 is a diagram of the switch fabric described above.

Throughout our analysis, we use the following notation to facilitate the presentation.

- $B_{i,j}$ – Internal buffer corresponding to input $i \in [1, N]$, output $j \in [1, N]$, 1 cell in size;

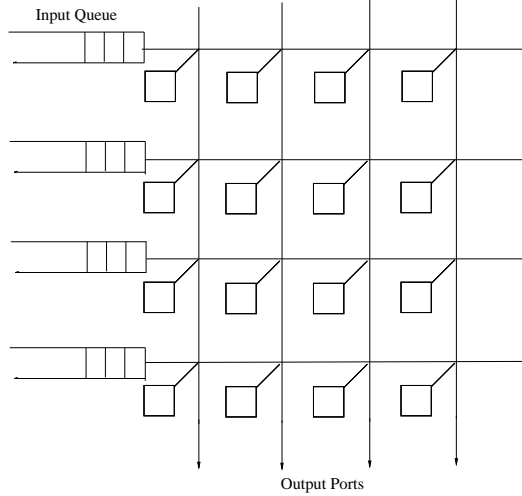


Figure 1: Diagram of a 4×4 Buffered Crossbar Switch

- Q_i – Input queue at input port i ;
- $p_k[n]$ – For any column of buffer cells $B_{.,j}$, the probability of having k packets in total at the end of time n ;
- $p_{i,j}$ – State transition probability of Markov chain model. After 1 time unit, the probability of changing from state i into state j , i and j represents the number of total packets in $B_{.,j}$;
- \bar{p}_k – $\lim_{n \rightarrow \infty} p_k[n]$
- \bar{P}_B^k – For a IBCS switch of size N , when $Q_{.,j}$ has totally k packets, the probability that none of the incoming packets will encounter blocking.
- $P_B(N)$ – For a IBCS switch of size N , when in steady state, the probability that none of the incoming packets will encounter blocking.

3 Study Approach and Results

3.1 Key Observations and Analytical Model

Because of symmetry (traffic and switching fabric structure), it is obvious that overall saturation throughput of the system equals to the line utilization of any output. The saturation throughput of the whole crossbar switch system is:

$$S_N = (1 - \bar{p}_o) + \bar{p}_o \cdot [1 - (1 - 1/N)^N]$$

At the beginning of any time slot t , as long as there is at least one packet in $B_{:,j}$, the output port will output a packet, the probability of that is $1 - \bar{p}_o$. Moreover, even if $B_{:,j}$ is empty at the beginning of time slot t , there still will be probability $1 - (1 - 1/N)^N$ that at least 1 of $B_{:,j}$ cells will receive a incoming packet, therefore the output port j will output a packet at the end of time slot t . It should be noted that when N goes to infinity, the above equation will converge to:

$$S = \lim_{N \rightarrow \infty} S_N = (1 - \bar{p}_o) + \bar{p}_o \cdot [1 - e^{-1}]$$

It can be easily seen that Discrete Markov Chain model can be used to model the above switching system. For one particular output line, the number of occupied crosspoint buffer cells in $B_{:,j}$ is chosen as the state of the Markov chain. The following is a transition diagram of the employed Markov chain:

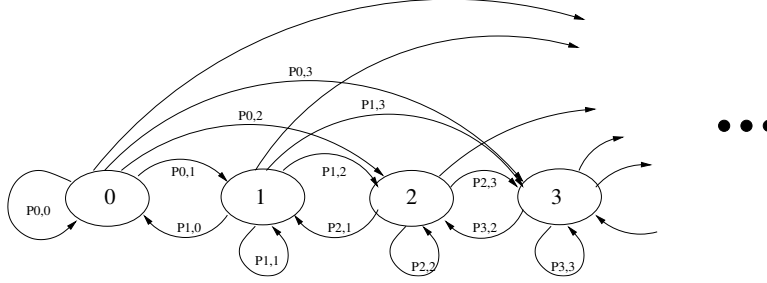


Figure 2: Diagram of a Markov chain to model state transition of the switch

3.2 Derivation Procedure

First, let us look at state transition matrix, where $p_{i,j}$ denotes the probability for the chain state change from state i to state j :

$$T = \begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} & \cdots & \cdots & p_{0,N-1} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} & \cdots & \cdots & p_{1,N-1} \\ 0 & p_{2,1} & p_{2,2} & p_{2,3} & \cdots & \cdots & p_{2,N-1} \\ 0 & 0 & p_{3,2} & p_{3,3} & \cdots & \cdots & p_{3,N-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots & p_{N-1,N-1} \end{pmatrix}$$

Key equations:

$$p_k[n+1] = p_0[n] \cdot p_{0,k} + p_1[n] \cdot p_{1,k} + \cdots + p_k[n] \cdot p_{k,k} + \cdots + p_{k+1}[n] \cdot p_{k+1,k}$$

Hence,

$$p_k[n+1] = \sum_{i=0}^k p_i[n] \cdot p_{i,k} + p_{k+1}[n] \cdot p_{k+1,k}$$

However, because,

$$\lim_{n \rightarrow \infty} p_k[n+1] = \lim_{n \rightarrow \infty} p_k[n]$$

Hence,

$$\bar{p}_k[1 - p_{k,k}] = \sum_{i=0}^{k-1} \bar{p}_i \cdot p_{i,k} + \bar{p}_{k+1} \cdot p_{k+1,k}$$

Therefore,

$$\bar{p}_k = \frac{1}{1 - p_{k,k}} \cdot \left(\sum_{i=0}^{k-1} \bar{p}_i \cdot p_{i,k} + \bar{p}_{k+1} \cdot p_{k+1,k} \right), \quad k \in [0, N-1]$$

Actually, we are only interested in the result of \bar{p}_0 , we need to solve N linear equations, once we know $p_{i,j}$ and N .

Now, we try to find out $p_{i,j}$:

$$p_{i,j} = \begin{cases} C_{N-i}^{j+1-i} \cdot \left(\frac{1}{N}\right)^{j+1-i} \left(1 - \frac{1}{N}\right)^{N-(j+1-i)-i} + \delta(i,j) \cdot \left(1 - \frac{1}{N}\right)^N, & \text{if } j+1 \geq i, \text{ or } j-i \geq -1 \text{ and } i, j \in [0, N-1]; \\ 0 & \text{o.w.} \end{cases}$$

At the beginning of "Buffer in" phase, there are i packets residing in $Q_{\cdot,j}$ buffer cells, after the end of "Buffer out" phase during the same time slot, there will be j packets in $Q_{\cdot,j}$, which obviously means that during the "Buffer in" phase, there are $j+1-i$ new packets coming in some empty buffer cells of $Q_{\cdot,j}$, the probability of that happening is $C_{N-i}^{j+1-i} \cdot \left(\frac{1}{N}\right)^{j+1-i} \left(1 - \frac{1}{N}\right)^{N-(j+1-i)-i}$. The reason of addition of term $\delta(i,j) \cdot \left(1 - \frac{1}{N}\right)^N$ is to account for the special case when $i, j = 0$.

After simplification,

$$p_{i,j} = \begin{cases} C_{N-i}^{j+1-i} \cdot \left(\frac{1}{N}\right)^{j+1-i} \left(1 - \frac{1}{N}\right)^{N-j-1} + \delta(i,j) \cdot \left(1 - \frac{1}{N}\right)^N, & \text{if } j+1 \geq i, \text{ or } j-i \geq -1; \\ 0 & \text{o.w.} \end{cases}$$

In summary, in order to solve for the saturation throughput $S_N (= (1 - \bar{p}_0) + \bar{p}_0 \cdot [1 - (1 - 1/N)^N])$, we need to first find out \bar{p}_0 , which can be easily solved by the following linear equation system, which consists of $N+1$ linear equations:

$$\begin{cases} \bar{p}_k = \frac{1}{1 - p_{k,k}} \cdot \left(\sum_{i=0}^{k-1} \bar{p}_i \cdot p_{i,k} + \bar{p}_{k+1} \cdot p_{k+1,k} \right), & k \in [0, N-1] \\ \bar{p}_N = 0; \\ \sum_{k=0}^N \bar{p}_k = 1 \end{cases}$$

In the above linear equation systems, the only unknown is $p_{i,j}$, which can be obtained according to the following:

$$p_{i,j} = \begin{cases} C_{N-i}^{j+1-i} \cdot \left(\frac{1}{N}\right)^{j+1-i} \left(1 - \frac{1}{N}\right)^{N-j-1} + \delta(i,j) \cdot \left(1 - \frac{1}{N}\right)^N, & \text{if } j+1 \geq i, \text{ or } j-i \geq -1; \\ 0 & \text{o.w.} \end{cases}$$

It should be noted, although the close form solution of those linear equations is hard to obtain, for any finite size N , solving those equations numerically is straightforward. We solved for switch size $N = 2, 4, 16, 32, 64, 128, \dots$, the analytical results matches exactly with the simulation results.

3.3 Internally Buffered Crossbar Switch without Blocking

In this subsection, we use the previous results to prove that when switch size N goes to infinity, the saturation throughput of IBCS fabric without blocking converges to 100%. Here, by "without blocking", we mean that for each input queue Q_i , at the beginning of time slot, if the head-of-line packet finds its corresponding crosspoint buffer cell $B_{i,j}$ is not available (occupied), it will be immediately dropped without being blocked. Apparently, the HOL(head-of-line) blocking in this case is removed. The assumption of non-blocking significantly reduces the complexity of analysis, making the Markov chain modelling manageable and the statespace reasonably small. In next subsection, we will remove this assumption. In short, we want to prove: $\lim_{N \rightarrow \infty} S_N = 1$ for non-blocking case. Obviously, because $S_N = (1 - \bar{p}_0) + \bar{p}_0 \cdot [1 - (1 - 1/N)^N]$, in order to prove $\lim_{N \rightarrow \infty} S_N = 1$, we only need to prove: $\bar{p}_0 = 0$.

We know from previous derivation:

$$p_{i,j} = C_{N-i}^{j+1-i} \cdot \left(\frac{1}{N}\right)^{j+1-i} \left(1 - \frac{1}{N}\right)^{N-j-1} + \delta(i,j) \cdot \left(1 - \frac{1}{N}\right)^N$$

When $N \rightarrow \infty$,

$$\begin{aligned} p_{i,j} &= \frac{(N-i)!}{(j+1-i)!(N-j-1)!} \cdot \frac{1}{N^{j+1-i}} \cdot \left(\frac{N-1}{N}\right)^{-j-1} \cdot e^{-1} + \delta(i,j) \cdot e^{-1} \\ &= \frac{e^{-1}}{(j+1-i)!} + \delta(i,j) \cdot e^{-1} \end{aligned}$$

The linear equation system will be following in $N \rightarrow \infty$ case,

$$\begin{pmatrix} p_{0,0} & p_{1,0} & 0 & 0 & \dots & \dots & \dots \\ p_{0,1} & p_{1,1} & p_{2,1} & 0 & \dots & \dots & \dots \\ p_{0,2} & p_{1,2} & p_{2,2} & p_{3,2} & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \cdot \\ \cdot \\ p_{N-1} \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \cdot \\ \cdot \\ p_{N-1} \end{pmatrix}$$

Therefore,

$$\begin{aligned}
p_0 &= p_0 \cdot 2e^{-1} + p_1 \cdot \frac{e^{-1}}{0!} \\
p_1 &= p_0 \cdot \frac{e^{-1}}{2!} + p_1 \cdot \frac{e^{-1}}{1!} + p_2 \cdot \frac{e^{-1}}{0!} \\
p_2 &= p_0 \cdot \frac{e^{-1}}{3!} + p_1 \cdot \frac{e^{-1}}{2!} + p_2 \cdot \frac{e^{-1}}{1!} + p_3 \cdot \frac{e^{-1}}{0!} \\
&\dots \quad \dots \quad \dots \\
&\dots \quad \dots \quad \dots \\
&\dots \quad \dots \quad \dots
\end{aligned}$$

Now, add all the above equations together, and merge all terms:

$$1 = p_0 \cdot e^{-1} + p_0 \cdot e^{-1} \cdot (1 + 1/1! + 1/2! + 1/3! + \dots) + p_1 \cdot e^{-1} \cdot (1 + 1/1! + 1/2! + 1/3! + \dots) + \dots$$

$$1 = p_0 \cdot e^{-1} + (p_0 + p_1 + p_2 + \dots)$$

Hence,

$$p_0 = 0.$$

Which leads to: $S_N = (1 - 0) + 0 \cdot [1 - (1 - 1/N)^N] = 1$.

3.4 Buffered Crossbar Switch with Blocking

In last section, we prove that for the internally buffered crossbar switch, if we allow packet dropping to happen when incoming packet finds the corresponding crosspoint buffer cell is occupied, the saturation throughput of the switch will approach 100% if switch size N goes to infinite. In this section, we remove that assumption, when the incoming packet finds no available crosspoint buffer cell, it will be blocked and wait for that cell to become available again (this phenomenon is very similar to HOL blocking in IQ switch without internal buffer).

Because of the complexity of this crossbar switch architecture, if we still use Markov chain to model the behavior, the state space size will exponentially increase with the switch size N , finding throughput result becomes extremely difficult if not impossible. Although we can only analytically compute the throughput of this type of system of small size N , our simulation results strongly suggest that for this type of system, when switch size N goes to infinite, the saturation throughput will also approach to 100%. In the following, we will analytically prove this conclusion.

The key to analyze this type of system is based on the following observation,

the major difference between blocking and non-blocking system lies in the fact that packets in blocking system may encounter HOL blocking, whereas in non-blocking system, such blocking will be immediately removed by dropping the blocked packet. Therefore, if the blocking probability for packets is 0, then there will be no difference in performance between blocking and non-blocking switches.

In the following, we are going to prove that when the switch size N goes to infinity, the blocking probability for arriving packets in internally buffered switch with blocking will approach 0. This result will lead to the conclusion that even for internally buffered crossbar switch with blocking, the saturation throughput will still be 100% when switch size N goes to infinity, i.e., $\lim_{N \rightarrow \infty} S_N = 1$.

Now, consider a particular column of crosspoint buffer cells, $B_{.,j}$. At the beginning of time slot t , there will be a packet at each input port, with a destined output randomly chosen from N output ports uniformly. Based on the i.i.d. bernoulli traffic assumption, those incoming packets are independent to each other. The probability for at least one packet is blocked is denoted as P_B . Obviously, the probability that none of incoming packets is blocked is $P_{\bar{B}} = 1 - P_B$. Clearly, $P_{\bar{B}}$ is only a function of the total number of packets k in $B_{.,j}$, which equals to $\left(\frac{N-1}{N}\right)^k$ for a particular k . In order to find the total non-blocking probability, we apply the law of total probability conditioning on the total number of packets in $B_{.,j}$.

$$P_{\bar{B}} = P\{\text{non-blocking}\} = \sum_{k=0}^N P(\bar{B}|k) \cdot P(k)$$

where: $P(\bar{B}|k)$ denotes the conditional probability of no blocking for all incoming packets, given there are k packets in $B_{.,j}$. $P(k)$ denotes the probability that there are k packets in $B_{.,j}$.

In section 4.3, we know that:

$$\bar{p}_k \cdot [1 - p_{k,k}] = \sum_{i=0}^{k-1} \bar{p}_i \cdot p_{i,k} + \bar{p}_{k+1} \cdot p_{k+1,k}, \quad k \in [0, N-1]$$

i.e.,

$$\bar{p}_k = \sum_{i=0}^{k+1} \bar{p}_i \cdot p_{i,k}$$

$$\begin{aligned} P_{\bar{B}}(N) &= \sum_{k=0}^N \bar{P}_B^k \cdot \bar{p}_k \\ &= \sum_{k=0}^N \sum_{i=0}^{k+1} \bar{P}_B^k \cdot \bar{p}_i \cdot p_{i,k} \end{aligned}$$

$$\begin{aligned}
&= \sum_{i=0}^{N+1} \sum_{k=i}^N \overline{P}_B^k \cdot \bar{p}_i \cdot p_{i,k} \\
&= \sum_{i=0}^{N+1} \bar{p}_i \left(\sum_{k=i}^N \overline{P}_B^k \cdot p_{i,k} \right).
\end{aligned}$$

Where,

$$\overline{P}_B^k \cdot p_{i,k} = \left(\frac{N-1}{N} \right)^k \cdot \left(C_{N-i}^{k+1-i} \cdot \left(\frac{1}{N} \right)^{k+1-i} \left(1 - \frac{1}{N} \right)^{N-k-1} + \delta(i, k) \cdot \left(1 - \frac{1}{N} \right)^N \right)$$

When $N \rightarrow \infty$

$$\lim_{N \rightarrow \infty} \left(\overline{P}_B^k \cdot p_{i,k} \right) = \frac{e^{-1}}{(k+1-i)!}$$

Therefore,

$$\begin{aligned}
P_{\overline{B}}(\infty) &= \sum_{i=0}^{\infty} \bar{p}_i \cdot \sum_{k=i}^{\infty} \frac{e^{-1}}{(k+1-i)!} \\
&= 1.
\end{aligned}$$

$$\begin{aligned}
P_B(\infty) &= 1 - P_{\overline{B}}(\infty) \\
&= 0
\end{aligned}$$

The above result demonstrates that when $N \rightarrow \infty$, the blocking probability of IBCS is 0, which implies that when switch size N goes to infinity, the saturation throughput performance is the same for IBCS with and without crosspoint buffer. Since we already proved that for IBCS without blocking, the saturation through converges to 100%, this implies that for IBCS with blocking, throughput is also 100%.

3.5 Results Summary

The throughput results including both analytical study and simulation are summarized in Table 1 ¹.

¹IBCS denotes Internally Buffered Crossbar Switch, and B and NB denotes with Blocking and without blocking.

N	Saturation Throughput [1]	IBCS-NB	IBCS-B
1	1.0000	1.0000	1.0000
2	0.7500	0.8331	0.8076
3	0.6825	0.8292	0.7751
4	0.6553	0.8356	0.7683
5	0.6399	0.8429	0.7672
6	0.6302	0.8497	0.7684
7	0.6234	0.8558	0.7701
8	0.6184	0.8611	0.7729
16	-	0.8905	0.7927
32	-	0.9166	0.8187
64	-	0.9378	0.8442
128	-	0.9542	0.8677
251	-	0.9668	0.8912
512	-	0.9758	0.9113
∞	0.5858	1.0000	1.0000

Table 1 shows that for all finite switch size, the saturation throughput of IBCS is larger than the corresponding crossbar switch without internal buffer in both blocking and non-blocking cases. Not surprisingly, For IBCS, the case without packet blocking achieves higher throughput than the one with packet blocking. On interesting thing to notice is that for both blocking and non-blocking cases, the throughput is not monotonously increasing until N larger than 5 and 4 respectively. Overall, as N increases, the throughput of IBCS increases, which is in sharp contrast with conventional crossbar switch without internal buffer. Figure 3 demonstrates the above observations.

Figure 4 illustrates the throughput improvement if internal buffer size W is increased (number of buffer cells at each crosspoint). It is obvious that more internal buffer cells at each crosspoint are beneficial because it can reduce more HOL blocking. At one extreme, when W goes to ∞ , even for finite switch size N , no HOL blocking will occur, and 100% throughput is achievable for any admissible traffic load.

4 Conclusion

The throughput performance of an IQ Buffered Crossbar Switch with 1 buffer cell at each N^2 crosspoint is studied using Markov chain model.

1. For case without packet blocking, the analytical solution of saturation throughput is obtained for finite switch size N , which is further verified by simulation results. We also prove that throughput converges to 100%

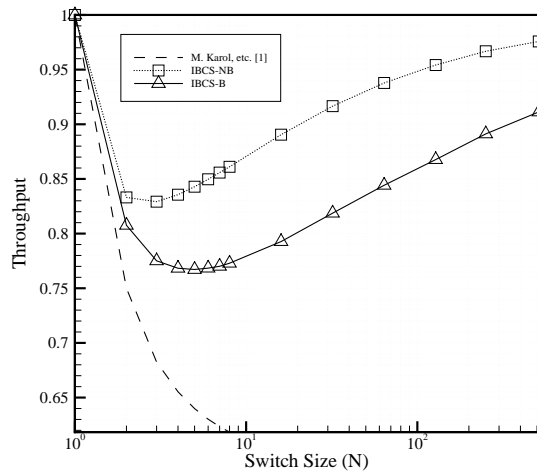


Figure 3: Comparison of throughput for all 3 cases(Karol, etc., IBCS-B, and IBCS-NB)

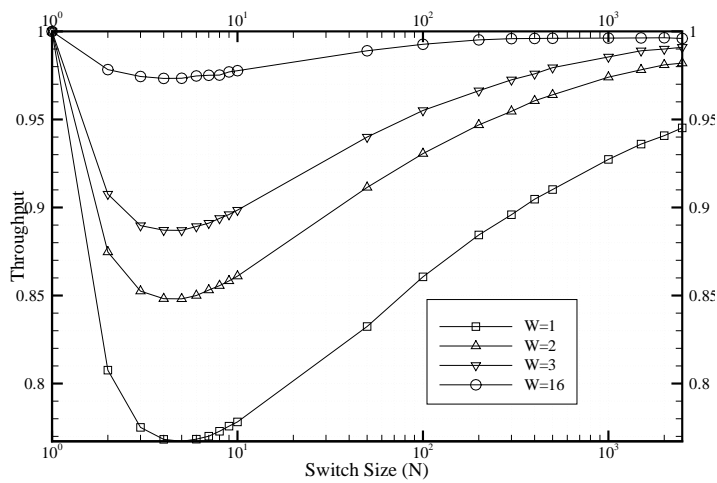


Figure 4: Throughput of IBCS for different internal buffer size($W=1, 2,$ and 3)

when switch size goes to infinity.

2. For case with packet blocking, the analytical solution of saturation throughput is obtained only for infinite switch size N . Extensive simulation studies are done for different switch size N . The simulation results strongly indicates 100% throughput for infinite N , which is proved analytically by proving blocking probability approaches 0 when $N \rightarrow \infty$.

References

- [1] K. M. J. Hluchyj, M. G., "Queueing in high-performance packet switching," *IEEE Journal of Selected Areas in Communications*, vol. 6, pp. 1587–1597, December 1988.
- [2] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-speed switch scheduling for localarea networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319–352, 1993.
- [3] N. McKeown, "The islip scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 2, pp. 188–201, 1999.
- [4] Y. Tamir and G. Frazier, "High performance multi-queue buffers for vlsi communication switches," *In Proc. of 15th Ann. Symp. on Comp. Arch.*, pp. 343–354, June 1998.
- [5] V. A. Nick McKeown, Adisak Mekittikul and J. Walrand, "Achieving 100throughput in an input-queued switch (extended version)," *IEEE Transactions on Communications*, vol. 47, pp. 1260–1267, August 1999.
- [6] A. Mekittikul and N. McKeown, "A practical scheduling algorithm to achieve 100pp. 792–799, April 1998.
- [7] H. J. Chao and J.-S. Park, "Centralized contention resolution schemes for a large-capacity optical atm switch," *in Proc. IEEE ATM Workshop, Fairfax, VA*, pp. 11–16, May 1998.
- [8] A. C. E Krishna, N. S. Patel and R. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 1052–1066, June 1999.
- [9] M. NABESHIMA, "Performance evaluation of a combined input- and crosspoint-queued switch," *IEICE Trans, Commun.*, vol. E83-B, March 2000.
- [10] R. M. T. Javadi and T. Hrabik, "A high-throughput scheduling algorithm for a buffered crossbar switch fabric," *Proceedings of IEEE 2001 International Conference on Communications*, pp. 1581–1591, June 2001.

- [11] Z. J. Rojas-Cessa, E. Oki and H. J. Chao, "Cixb-1: Combined input-once-cell-crosspoint buffered switch," *IEEE Workshop on High Performance Switching and Routing*, vol. Dallas, July 2001.