Design of ATM Switch Using Hypercube with Distributed Shared Input Buffers and Dedicated Output Buffers

Derek C. W. Pao and W. N. Chau
Department of Electronic Engineering
City University of Hong Kong
Tat Chee Avenue, Kowloon
Hong Kong
Email: eedpao@cityu.edu.hk

Abstract

We investigate a dynamic packet routing approach to ATM switch design using hypercube. An $(n+1)$-dimensional hypercube is used to implement an $N \times N$ switch, where $N = 2^n$. Cells arriving at input ports are routed towards their destinations in store-and-forward (SAF) manner. In addition to the SAF buffer, each input/output port has a dedicated buffer. A distributed deflection routing algorithm where the routing priority is based on the age of the cells is developed. An interesting feature of the routing algorithm is that the store-and-forward buffers and the input buffers behave as distributed shared-buffer which effectively smooth out uneven traffic. In addition, our routing algorithm does not suffer from the HOL blocking problem as in the conventional input-output buffered switch architecture. The processing power of each node in the hypercube scales up by a factor of $O(\log N)$ as the network size N is increased. Hence, our approach is suitable for implementing large scale ATM switches. Performance of our design is studied via simulation and found to be better than the conventional input-output buffered nonblocking switch architecture.

1. Introduction

Asynchronous Transfer Mode (ATM) is the switching and multiplexing technique chosen by CCITT for broadband access to the Integrated Services Digital Network (ISDN) and is widely accepted by common carriers as the mode of operation for future communication systems. ATM is based on packet switching using small, fixed-size packets of 53 bytes called *cells*, and line speeds equal to 150 Mbps, 600 Mbps and above.

In general, an ATM switch has N input and N output lines. The arrival times of cells at the input lines are time-synchronized. The time required to transmit a cell is usually referred to as a *cell time*. However, there is no coordination among arriving cells as far as their destination requests are concerned. It is possible to have more than one cells arriving in the same cell time that destined to the same output port. Such an event is referred to as *output contention*. Buffering within the switch is required to resolve output contention. Due to resource limitations (finite buffer memory), cells may be lost. Hence, one of the major issues in the design of ATM switches is concerned with the buffering strategy that aimed at minimizing cell loss probability.

There have been a large number of ATM switch designs published in the literature. Comprehensive surveys on the subject can be found in [1, 9]. ATM switch architectures can be broadly classified into three categories: the *shared-memory* type, the *shared-medium* type, and the *space-division* type. The major drawback of the shared-memory and shared-medium architectures is that the shared component (the memory or switching hardware) of the switch need to operate at a speed equal to N times the speed of the input line. As a result, these two types of architecture are only suitable for implementing small ATM switches. Recently, Wei and Kumar attempted to alleviate the bandwidth requirement of the shared-memory by using multiple shared-memory modules [11, 12]. In their approach, the time required to schedule the arriving cells to be stored in one of the memory modules is equal to $2N/R$, where R is the ratio of the bandwidth of each memory module over the input line speed. Since R is likely to be a constant factor, the scalability achievable is still limited.

Much of the recent studies on ATM switch have been focused on the performances of buffering schemes for various types of space-division switches. The most studied is the input-output buffered nonblocking networks [2, 4, 6, 7]. It is well known that simple input-queueing methods suffer from the *head-of-line* (HOL) blocking problem and the maximum throughput achievable is about 58.6% under uniform random traffic [13]. Some common approaches to improve the system throughput

0-8186-7216-1/95 $04.00

includes look ahead scheduling of the cells in the input queues to reduce output contention, input-port/output-port expansion (or lines grouping), switch speed-up, and output-queueing. Different combinations of the above methods may be incorporated in an ATM switch design.

In output-queueing, the output buffer should be able to accept multiple cells that are destined to it in a cell time. Hence, the output buffer should operate at a speed higher than the input line speed. The actual speedup required depends on the design of the switching fabric. Cells arriving at an output port whose output buffer is full will be lost. To reduce cell loss due to output buffer overflow, some kind of coordination between the output ports and the input ports is required. In [2], cells at the HOL of each input queues are dispatched by a centralized scheduler who is assumed to know the up-to-date status of each output port. The major drawback of this approach, other than the memory bandwidth requirement, is that the scheduler need to operate at a speed proportional to the size of the switch.

In this paper, we investigate an alternative approach to the design of ATM switch using dynamic routing network, such as the hypercube. In an n-dimensional hypercube, there are 2^n nodes, and each node has n internal communication links (or links in short) connecting to its n neighboring nodes. In our design, we employ an $(n+1)$-dimensional hypercube to implement an $N \times N$ switch. Half of the nodes in the hypercube are input nodes, and the other half are output nodes. An input (output) node will have an additional external input (output) link. We assume that all communication links operate at the same speed. In our approach, once a cell is injected into the hypercube, it will find its own way to the destination port in store-and-forward (SAF) manner. Beside the SAF buffer, there is also an input (output) buffer of size B_i (B_o) associated with each input (output) port. When a cell arrives at its destination, it will be stored in the output buffer. If the output buffer is full, the cell may be allowed to circulate in the network and try again at some later time. A cell will be dropped if it fails to enter the destination output buffer after a predefined amount of time. Since the cells may arrive at the destination out of sequence, the output node is required to restore the original sequence when sending cells to the external link. When a cell arrives at an input port, it will be scheduled to be sent to a neighboring node or stored in the input buffer if empty space is available, otherwise it is dropped. In each cell time, the input node selects the $n+1$ "oldest" cells among the cells in the input buffer and the set of cells it received from its incoming links. The selected cells will be sent to the neighboring nodes, while up to B_i of the remaining cells will be stored in the input buffer and wait for the next cell time. It is possible for a cell which enters the network at node u to be temporarily stored in the input buffer of another node v. Thus, the input buffers and the SAF buffers behave as a distributed "shared-memory".

It is obvious that our design does not suffer from the HOL blocking problem. The processing power of each node and the memory bandwidth requriement only scale up by a factor of $O(\log_2 N)$. According to our simulation study, a 32×32 switch with input buffer size of 2 and output buffer size of 32 is able to achieve a maximum (saturated) throughput of about 98.5% for uniform random traffic, and 81.1% for bursty traffic with mean burst length equal to 10 where the bursts are geometrically distributed. If the output buffer size is increased to 128, the maximum throughput is improved to about 99.6% and 92.2% for uniform random traffic and bursty traffic, respectively. The simulation study also reveals that output buffering is much more effective than input buffering even under bursty traffic at moderate load. This disagrees with the findings of [6] for input-output buffered nonblocking switches. The discrepancy may be due to the differences in behaviour of the input buffer and the presence of the SAF buffer in our design.

Packet routing in hypercube has been studied extensively in the field of parallel processing [5]. In the context of parallel processing, one is primiarily concerned with the permutation routing problem where packets are not continuously injected into the network.

Deflection packet routing in hypercube in the context of data communication has been studied in [3, 8], and a modulo routing approach has been studied in [10]. In these studies, each node is assumed to have an infinite output buffer and the requirement of FIFO delivery of packets is relaxed. The input traffic is assumed to be random and uniformly distributed. In [3, 10], each node may inject up to n (which is the dimension of the hypercube) packets into the network in each time slot (cell time). The maximum throughput of each node is found to be less than or equal to 2 packets per time slot for both deflection routing and the modulo routing approach. Because of the differences in assumptions and packet arrival model, it is rather difficult to relate these results to our work. The study in [8] is closer to our work in which each node may inject no more than one packet in each time slot. A maximum (saturated) throughtput for a 10-dimensional hypercube under uniform random traffic is found (by simulation) to be 99.93%. This can be regarded as the upper bound of the throughput achievable. We can see that with moderate amount of output buffer (say 32 to 128 cells), our design achieve a throughput very close to that upper bound.

The organization of this paper is as follow. Details of the switch architecture, routing algorithm, and buffer management strategy will be presented in section 2. Simulation study on the performance of the design, and

comparisons with other input-output queueing methods are presented in section 3. Section 4 is the conluding remarks.

2. Switch Architecture

In an n-dimensional hypercube, there are 2^n nodes numbered from 0 to 2^n-1. Two nodes u and v are connected iff the binary representations of u and v differ at exactly one bit position. Hence, each node in the hypercube is connected to n neighboring nodes. If u and v differ at the ith bit position, the link connecting u and v is referred to as the ith link of node u (or v). In this paper, we assume that internal communication links of the hypercube are bidirectional.

Assume N is a power of 2, and $n=\log_2 N$. To implement an $N\times N$ switch, we employ an $(n+1)$-dimensional hypercube. The hypercube is divided into two halves, the output subcube consisting of the nodes numbered from 0 to $N-1$, and the input subcube consisting of the nodes numbered from N to $2N-1$. Each input (output) node has an external input (output) link. The input/output links are unidirectional. Each node contains a SAF buffer of size $n+1$ to implement the store-and-forward routing function. In addition to the SAF buffer, an input (output) node has an input (output) buffer of size B_i (B_o). Figure 1 shows the structure of a 8×8 switch.

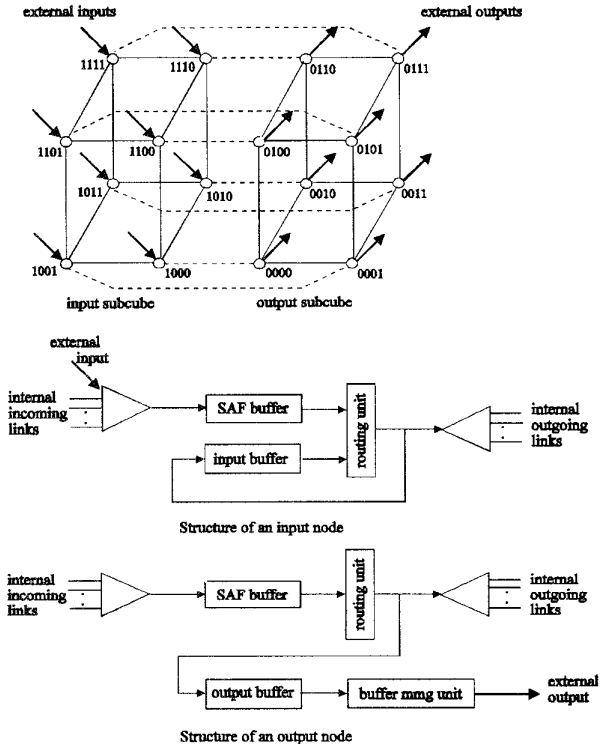

Figure 1. Organization of an 8×8 switch

2.1 Routing Function of Input Node

Cells arriving at input ports are routed towards the corresponding output ports in store-and-forward manner. The routing function of an input node outlined below. An age tag and a sequence number are attached to each cell upon arrival from the external link. The age of a cell, which is initialized to zero, is incremented by 1 in every cell time until it is admitted into the output buffer of the destination node. At each cell time, an input node may receive up to $n+1$ cells from its internal links and 1 cell from the external input link. The input node sorts the arriving cells according to their age (Since the age of a cell is likely to be bounded and approximate ordering of cells is tolerable, the sorting can be done by simple bucket sort with $O(n)$ buckets). Cells currently stored in the input buffer are also maintained in sorted order, a complete list of cells ordered by their ages is obtained by merging the list of arriving cells and the cells in the input buffer. The $n+1$ oldest cells are selected to be sent out in the next cell time. Up to B_i cells that have not been selected will be stored in the input buffer. In case if the input buffer is full and the node receives $n+1$ cells from its internal links, then the newly arrived cell from the external link will be dropped. Among the selected cells to be sent out, the oldest cell is given the highest priority in choosing the communication link to reduce its distance from the destination. It is possible that a younger cell is being routed farther away from its destination. However, as the cell gets older, it will be given higher priority and be able to progress towards the destination node. It is interesting to note that cells entering the hypercube at node u may be temporarily stored in the input buffer of another input node v. Hence, the SAF buffers and the input buffers behave as distributed shared-buffers.

2.2 Routing Function of Output Node

Two functions are performed at an output node, (i) routing of the arriving cells, and (ii) selection of a cell from the output buffer to be sent to the external link. Cells from a source node that destined for the same output node may reach the destination out of sequence. However, the routing function always gives higher priority to older cells, the degree of out of sequence arrivals should not be severe (It is found in the simulation study that the displacement of cells which are out of sequence is less than or equal to 4 in most of the cases). One of the functions of an output node is to restore the original sequence when sending cells to the external link. The process to select a cell for output is outlined below. The output node maintains N virtual queues, one for each (external) input line. Associated with each virtual queue is an expected sequence number of the next cell to be sent out. The output buffer is shared by the N virtual queues.

When a cell is admitted by an output node, it is inserted into the corresponding virtual queue. Cells in each virtual queue are ordered by their sequence numbers. A virtual queue is said to be active if it is not empty. Since cell loss due to input/output buffer overflow is possible, a timeout mechanism is introduced to avoid indefinite waiting of cells in the virtual queues. The HOL cell of a virtual queue is said to be enabled for output with *high* priority if its sequence number is equal to the corresponding expected sequence number, or it has been waiting in the virtual queue for a period greater than certain value, say *TimeoutH*. If the HOL cell's sequence number is greater than the expected value, and it has been waiting for a period less than or equal to *TimeoutH* but greater than *TimeoutL*, then the cell is enabled for output with *low* priority. Among the enabled cells of the same priority class, preference is given to cells that has longer waiting time in the queue (The selection of cells from the list of active virtual queues can be done in round-robin order. This may simplify the implementation of the output buffer management unit). The selection process is outlined below:

(i) If the *high* priority class is not empty, then the cell with the longest waiting time in the class is selected.

(ii) If (i) is not applicable and the output buffer is full, then the cell with the longest waiting time in the *low* priority class is selected.

The expected sequence number of the corresponding virtual queue is updated to 1 plus the sequence number of the cell selected.

The routing function of an output node is similar to that of an input node, except for the handling of cells reaching their destinations. The cells arriving at an output node are sorted in order according to their ages. Older cell is given priority over younger cells in choosing communication links or being admitted into the output buffer. A cell arriving at the destination node will be routed according to the following rules (applied in the given order):

(i) Dropped if its sequence number is smaller than the expected value.

(ii) Admitted into the output buffer if empty space is available.

(iii) Derouted to an adjacent node if the age of the cell is less than or equal to a threshold value, *AgeThreshold*; otherwise, it is dropped.

It has been shown in [6] that at high applied load if buffers are not shared in a fair manner, sharing buffers could actually make performance worse than not sharing buffers. Under heavy bursty traffic load, it is possible for the shared buffers to be taken up by cells destined for a few output ports, and adversely affects the traffic flow of the other communication paths. In our design, we introduce the *AgeThreshold* to prevent unfair sharing of buffers. When an output port is overloaded, i.e. its output buffer is full and cells destined to it cannot be admitted into the output buffer after circulating in the network for a certain period of time, then these cells should be eliminated so that the traffic flow of the other paths will not be affected. The optimum value of *AgeThreshold* depends on the overall system traffic conditions. In section 3.1, we will present a self-regulating scheme such that the system can adjust the value of *AgeThreshold* adaptively.

Intutitively, the values of *TimeoutH* and *TimeoutL* are closely related to *AgeThreshold*. If we allow a cell to circulate in the network for *AgeThreshold* number of cycles or more, then we cannot rule out the possibility of having successive cells from a source node to reach the destination node *AgeThreshold* number of cycles apart (and may be out of sequence). Hence, in our simulation study, we set *TimeoutH* to be equal to *AgeThreshold*, and *TimeoutL* to be $0.5 \times$ *TimeoutH*.

3. Simulation Study

In this section, we study the system performances in terms of cell loss probability and cell delays for different settings of the design parameters, which include the *AgeThreshold*, *TimeoutH* and *TimeoutL*; sizes of input buffer and output buffer; and network size. Let ρ denote the applied load, and $\alpha(\rho)$ denote the cell loss probability. Then the normalized throughput is given by $\Gamma(\rho) = 1 - \alpha(\rho)$.

Two types of traffic patterns, namely uniform random traffic (URT) and bursty traffic (BT), are used in the simulation. For bursty traffic, traffic correlation is modeled by ON-OFF source feeding each input port [2]. The burst length of an active period is geometrically distributed. Let p_A be the probability that an active period terminates in the next cell time. The probability that an active period will last for a duration of i cell times is $P_A(i) = p_A(1-p_A)^{i-1}$, $i \geq 1$. The mean active duration is $m(A) = \sum_{i=1}^{\infty} iP_A(i) = 1/p_A$. The geometrically distributed idle period is characterized by another parameter p_S. Similarly, the mean idle period is given by $m(S) = 1/p_S$. Cells belonging to the same burst are destined for the same output port, while bursts are addressed uniformly among the N output ports. Cells of the same burst are consecutively spaced. The average load applied to an output port is given by $\rho = m(A)/(m(A)+m(S))$.

For each simulation run, the network is made to reach a steady state by running the simulator for about 10,000 cell times before measurements are taken. To obtain cell loss probability with an accuracy in the order of 10^{-5}, measurements of data are done over a period in which

about 10 millions cells are injected into the network. Unless otherwise stated, a 32×32 switch with $B_i = 2$ is used in the simulation study.

3.1 Effects of *AgeThreshold* on System Throughput

From the discussion in section 2, we note that cell loss may be due to one of the following three reasons: (i) input buffer overflow; (ii) output buffer overflow; and (iii) timeout (due to out of sequence arrival). The three types of cell loss are closely related. The determining factor is the value of *AgeThreshold*. If cells are allowed to circulate longer in the network, then cell loss due to output buffer overflow can be reduced at the expenses of increasing cell loss of the other two categories, especially for input buffer overflow. Intuitively, when cell loss starts to occur at input nodes, it implies that traffic flow in part of the network is near saturation. The best way to improve the traffic flow of the overall system is to remove those cells that cause the congestion. This can be achieved by reducing *AgeThreshold*. Figure 2 shows the cell loss probability and the distribution of cell losses for different values of *AgeThreshold*. It can be easily observed that when cell loss starts to occur at inputs, the system throughput is near optimum. Hence, one simple way to adjust *AgeThreshold* is by monitoring the cell loss at the input nodes. If no cell loss has been observed at input nodes for a certain period of time, then the value of *AgeThreshold* can be slowly increased. Similarly, when cell loss occurring at input nodes exceeds a certain rate, the value of *AgeThreshold* should be slowly decreased. We recommend a minimum value of *AgeThreshold* to be about 4 plus the network diameter. This value will allow a cell to take at least two derouting steps in the network before reaching the destination. The recommended value is actually close to the optimum value when the network traffic is very heavy. The maximum value of *AgeThreshold* used in the simulation study is 40.

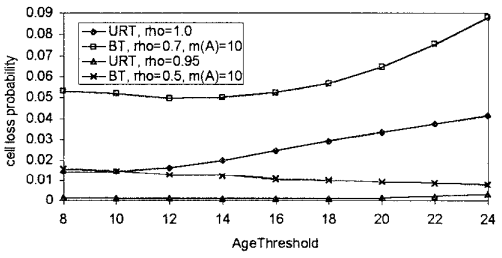

Figure 2a. Effects of agethreshold on cell loss probability

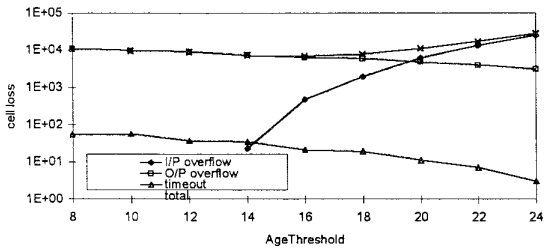

Figure 2b. Distribution of cell losses for URT with ρ=0.95

Another observation worth mentioning is that the amount of cell loss due to timeout accounts for only a very small percentage of the total cell loss. This fact implies that the degree of out of sequence arrival of cells is tolerable, and justifies the use of dynamic packet routing approach to implement ATM switches.

3.2 Effectiveness of Input Buffering versus Output Buffering

Figure 3 shows the throughput of the system for different combinations of B_i and B_o with $B_i + B_o = 48$. For heavy traffic load, the value of *AgeThreshold* is fixed at 10, while for moderate load, the value of *AgeThreshold* is adjusted from 10 to about 24 as the input buffer size is increased. It is interesting to note that output buffering consistently out perform input buffering under light to heavy applied load. One possible reason for that is because in output buffering, cells are removed from the network once they are stored in the output buffer; whereas in input buffering, cells are only delayed from progressing towards their destinations. In addition, we can observe that the cell loss probability shoots up sharply when the output buffer size drops below 16. It is because of the fact that the output node need a certain amount of output buffer to restore the original sequence of the cells. Based upon this observation, we only employ an input buffer of size 2 in the subsequent simulation study.

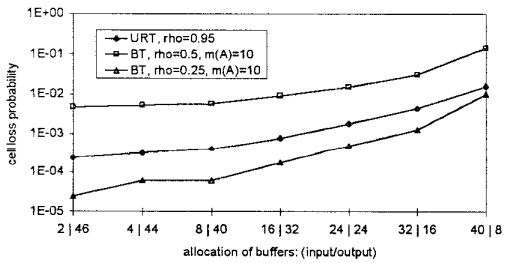

Figure 3. Effectiveness of input buffering vs. output buffering with total buffer = 48 cells.

3.3 Performance Comparison With Input-Output Buffered Nonblocking Switch

In this section, we compare the performance of our design with the input-output buffered nonblocking switch architecture of Badran and Mouftah [2]. In their design, each input (output) port has a dedicated buffer of size $b_i(b_o)$. An output port is able to accept up to $\min(b_o, N)$ cells in a cell time. The switching is controlled by a central control unit (CCU). The CCU obtains the current status of the output queues and scans through the N input queues in round-robin order in each cell time. Cells at the HOL position are dispatched to the corresponding output ports provided sufficient empty buffer spaces are available. If an output port j has b empty buffers and there are $k > b$ cells at the HOL position are destined for port j, then only b out of the k cells can be dispatched, and the other $k-b$ cells will be delayed. The selection criteria can be based on the queue length or the waiting time of the HOL-cell.

There are two major reasons why we make our comparison with this architecture. Firstly, this architecture assumes (i) internal switching speed and memory bandwidth to be proportional to the size of the switch, (ii) the availability of up-to-date status information of output ports, and (iii) centralized control; its performance should be close to the upper limit achievable for switches based on input-output buffering. Secondly, very detailed performance results have been reported, especially for bursty traffic.

Pattavina and Bruzzi presented both analytical models and simulation results for an input-output buffered nonblocking switch with internal speedup [7]. However, their study is limited to uniform random traffic.

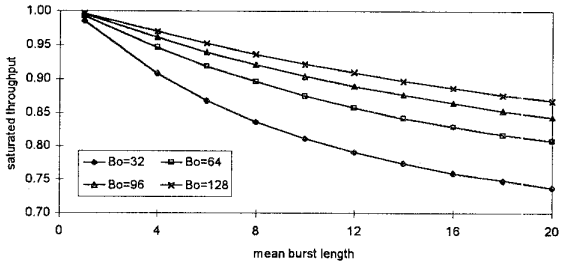

Figure 4. Saturated throughput vs. mean burst length

First, we look at the saturated throughput of our design. Figure 4 shows the achievable maximum throughput for different average burst lengths. The limiting case where the burst length equals 1 corresponds to uniform random traffic. It can be seen that the throughput decreases rapidly when the burst length is increased. If we have an input buffer size of 2 and output buffer size of 32, our design is able to achieve a saturated throughput of about 98.5% for uniform random traffic, and 81.1% for bursty traffic with $m(A)$=10. If the output buffer size is increased to 128, the saturated throughput is improved to about 99.6% and 92.2% for uniform random traffic and bursty traffic, respectively. These results compare favourably with that of [2,7]. In [7], it is shown that the asymptotic throughput under uniform random traffic for a switch with an internal speedup factor of 4 is 99.55%. Comparing with the result of [2] for bursty traffic with $m(A)$=10, they obtain a saturated throughput of about 65% when the output buffer size is 32, and about 83% when the output buffer size is 128.

Figure 5 shows the cell loss probability against the system loading for three traffic patterns, namely, uniform random traffic, and bursty traffic with $m(A)$ equal to 5 and 10. Let's define the *acceptable load*, ρ_a, be the amount of applied load to the switch that causes the cell loss probability to reach a reasonable limit of 10^{-5}. For $B_o = 32$, we have ρ_a equal to about 90%, 45%, and 20% for the above three traffic patterns, respectively. If B_o is increased to 128, then ρ_a is improved to about 97%, 75%, and 50%, respectively. These performance figures compare favourably with the input-output buffering architecture. In [2], when the total buffers per input/output line pair is 64, the acceptable load is about 45% and 20% for bursty traffic with $m(A)$ equal to 5 and 10, respectively.

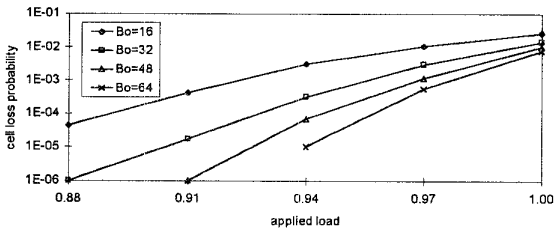

Figure 5a. Cell loss probability for URT

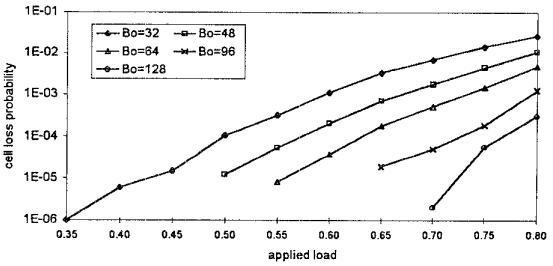

Figure 5b. Cell loss probability for BT with $m(A)$=5

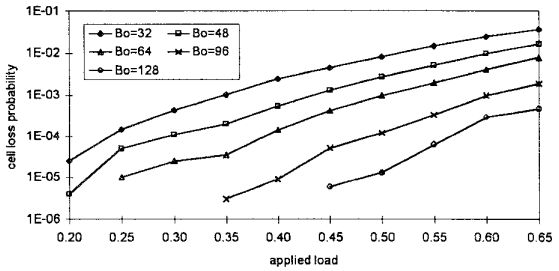

Figure 5c. Cell loss probability for BT with $m(A)$=10

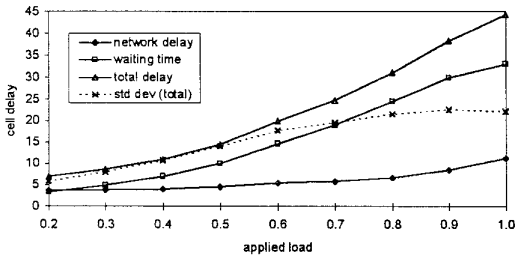

Figure 6b. Average cell delays for BT, $m(A)$=10 with N=32, Bo=64

3.4 Cell Delay and Effects of Network Size

There are two components that contribute to the total delay of a cell. The first one is the *network delay* which is the time taken by the cell to be admitted by the destination port, and the second is the waiting time in the output buffer. Figure 6 shows the average cell delay and standard deviation of the total delay under uniform and bursty traffic conditions. Only those cells that are successful in getting through the switch are included in the measurements of cell delay. It can be seen that the network delay is relatively stable from low to moderate applied load. Under heavy traffic, it is more likely that a cell will find the output buffer of the destination port full, and need to circulate longer in the network. This effect is even more prominent for bursty traffic condition. As expected, the average waiting time of a cell in the output buffer is significantly higher for bursty traffic condition. When the network is saturated (applied load = 1.0), the average waiting time is slightly above $0.5 \times B_o$ for both cases.

We have also studied the effects of network size on the saturated throughput and cell delay. We find that with increased network size, the throughput is improved marginally. This is because as the network size is increased, the total amount of SAF buffers is also increased. For cell delays, the network delay component increases linearly with the network dimension, whereas the waiting time at the output buffer is relatively insensitive to the network size.

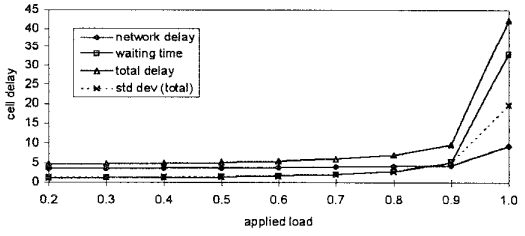

Figure 6a. Average cell delay for URT with N=32, Bo=64

4. Conclusion

We have demonstrated the feasibility of the dynamic packet routing approach to ATM switch design. The hypercube network is employed because of its regularity. However, other types of network with high bisection bandwidth may also be used. Each node in an n-dimensional hypercube has n internal communication links. Hence, multiple cells from the input buffer can be dispatched to adjacent nodes in each cell time. The selection of cells is based on the age of the cells rather than the destination ports. As a result, our design does not suffer from the HOL blocking problem as in the conventional input-output buffering architecture. Another important attribute of our design is that the memory bandwidth and procesing power of each node only need to be scaled up by a factor of $O(\log N)$ where N is the size of the switch. This makes our design much more scalable than the conventional ATM switch architectures.

The performance of our design is studied via simulation. It is shown that our design achieves better saturated throughput and lower cell loss probability as compared with an "idealized" input-output buffered nonblocking switch architecture [2]. The improvement in performance is due to the following factors:

(i) Our deisgn does not suffer from the HOL blocking problem.

(ii) The SAF buffers and the input buffers behave as distributed shared-buffer that effectively smooth out uneven traffic.

(iii) The self-regulating property of the routing algorithm prevents the unevenly distributed traffic at a few ports from congesting the network and blocking the other traffic paths.

Acknowledgement

This research is supported by City University of Hong Kong Research Grant No. 903349. The authors would like to thank Ms. Shirley Ng for her assistance in the preparation of the diagrams. We are also grateful to Dr. K. T. Ko for his valuable comments on the manuscript.

References

1. H. Ahmadi and W. E. Denzel, "A survey of modern high-performance switching techniques", IEEE J. on Selected Areas in Communications, Vol. 7, No. 7, 1989, pp. 1091-1103.
2. H. F. Badran and H. T. Mouftah, "ATM switch architecture with input-output-buffering: effect of input traffic correlation, contention resolution policies, buffer allocation strategies and delay in backpressure signal", Computer Networks and ISDN Systems, Vol. 26, pp. 1187-1213, 1994.
3. A. G. Greenberg and B. Hajek, "Deflection routing in hypercube networks", IEEE Trans. on Communications, Vol. 40, No. 6, pp. 1070-1081, 1992.
4. M. J. Karol, K. Y. Eng and H. Obara, "Improving the performance of input-queued ATM packet switches", INFOCOM'92, pp. 110-115, 1992.
5. T. F. Leighton, Introduction to Parallel Algorithms and Architectures Arrays-Trees-Hypercubes, Morgan Kaufmann Publishers, 1992.
6. S. C. Liew, "Performance of various input-buffered and output-buffered ATM switch design principles under bursty traffic: simulation study", IEEE Trans. on Communications, Vol. 42, No. 2/3/4, pp. 1371-1379, 1994.
7. A. Pattavina and G. Bruzzi, "Analysis of input and output queueing for nonblocking ATM switches", IEEE/ACM Trans. on Networking, Vol. 1, No. 3, pp. 314-327, 1993.
8. T. Szymanski, "An analysis of "hot-potato" routing in a fiber optic packet switched hypercube", INFOCOM'90, pp. 407-414, 1990.
9. F. A. Tobagi, "Fast packet switch architectures for broadband integrated service networks", Proceedings of the IEEE, Vol. 78, No. 1, pp. 133-167, 1990.
10. E. A. Varvarigos and D. P. Bertsekas, "Performance of hypercube routing schemes with or without buffering", IEEE/ACM Trans. on Networking, Vol. 2, No. 3, pp. 299-311, 1994.
11. S. X. Wei and V. P. Kumar, "On the multiple shared memory module approach to ATM switching", INFOCOM'92, pp. 116-123, 1992.
12. S. X. Wei and V. P. Kumar, "Decentralized control of a multiple shared memory module ATM switch", ICC'92, pp. 704-708, 1992.
13. M. Hluchyj and M. Karol, "Queueing in high-performance packet switching", IEEE J. Selected Areas in Communications, Vol. 6, No. 9, pp. 1587-1597, 1988.