Reducing Cell Loss in Banyan Based ATM Switching Fabrics

M. Al-Mouhamed, H. Youssef, and M. Kaleemuddin *

Abstract

In this paper, we propose a new technique for reducing cell loss in multi-banyan based ATM switching fabrics. We propose a switch architecture that uses incremental path reservation based on previously established connections. Path reservation is carried out sequentially within each banyan but multiple banyan planes can be concurrently reserved. We use a conflict resolution approach according to which banyans make concurrent reservation offers of conflict-free paths to head of the line cells waiting in input buffers. A reservation offer from a given banyan is allocated to the cell whose source to destination path uses the largest number of partially allocated switching elements which are shared with previously reserved paths. This approach leaves the largest number of free switching elements for subsequent reservations which has the effect of reducing potential of future conflicts and improves throughput. Paths are incrementally *clustered* within each banyan. We present a pipelined switch architecture based on the above concept of *path-clustering* which we call *Path Clustering Banyan Switching Fabric* (PCBSF). An efficient hardware to implement PCBSF is presented together with its theoretical basis. Performance and robustness of PCBSF are evaluated under simulated *uniform traffic* and *ATM traffic*. We also compare cell loss rate of PCBSF to that of other pipelined banyan switches by varying the switch size, input buffer size, and traffic pattern.

Keywords: ATM switch architecture, ATM traffic, multistage networks, performance evaluation, simulation, traffic modeling.

1 Introduction

Asynchronous Transfer Mode (ATM) is the transport mode of Broadband-ISDN (integrated services digital network) [1, 2, 3]. ATM is a cell switch technology. At the source end system, the traffic stream is split into small fragments of 48 bytes each. A fragment together with a five byte header make up an ATM protocol data unit called a *cell*. The 5 byte header contains ATM protocol information required to deliver the cell to the destination end system across an ATM network. ATM prides itself of being the base of future

*Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia (Email: mayez@ccse.kfupm.edu.sa)

ubiquitous computer networks. It is designed to operate over high speed and is limited only by the technological barriers of transmitting hardware and links.

The connection oriented nature of ATM, together with the use of statistical multiplexing and fixed size small cells allow ATM to adequately support real-time and non-real-time communication. ATM uses the concept of virtual connections between end-stations [4, 5]. Two types of connections are possible: permanent and switched. A permanent virtual circuit (PVC) is a connection that is manually established and manually released. End-stations do not have the ability to do that dynamically. A switched virtual circuit (SVC) is a connection that is dynamically established and released via signaling as required. Virtual connections are identified by their virtual paths and virtual channels. A virtual path is a logical construction of a group of virtual channels. The cell header combines both a virtual path identifier (VPI) and a virtual channel identifier (VCI). These identifiers will guide the cell through the ATM network

Besides its connection oriented nature, ATM has a number of unique and desirable features: (1) It provides the speed on a per-source basis, which means that each source can have its own high speed; (2) ATM is scaleable, since it can provide higher speeds to applications that require it; and (3) flexible, since we can mix speeds within a network according to the user requirements. The speed mixing is handled by the ATM equipment automatically.

ATM connections are established with negotiated quality of service (QoS) requirements, thus enabling real-time communication service such as videophony and video conferencing. Quality of Service is a unique feature of ATM. Depending on the application requirements, workstations can request specific QoS parameters for the connections they are going to setup. Five service categories are supported by ATM systems: (1) Constant Bit Rate (CBR), (2) Variable bit rate-real time (VBR-rt), (3) Variable Bit Rate-non real time (VBR-nrt), (4) Available bit rate (ABR), (5) and Unspecified bit rate (UBR). The source category is based on the declared QoS parameters, namely the Peak Cell Rate, the Sustainable Cell Rate, and Maximum Burst Size.

ATM technology is gaining ground, as more systems are getting deployed, and many of the complex issues getting resolved and standardized. One of the difficult problems addressed by the industrial and research communities is the engineering of ATM switches capable of accommodating a variety of traffic sources with conflicting quality of service requirements, and which can scale up in performance to rising bandwidth needs.

Banyan networks, a class of *multistage interconnection networks* (MINs), have several desirable features such as space division, self routing, low hardware complexity, and regular structure which make them suitable for VLSI implementation. Unfortunately, their throughput is far from being acceptable due to *internal* and *external* blocking. Example of commercial switches is the *Starlite switch* [6] and the *Sunshine switch* [7]. The Starlite switch uses recirculation to avoid output blocking in Batcher-banyan sorter networks [8]. The Sunshine switch consists of Batcher sorting network at the input which sorts the cells based on their destinations.

One common approach used to increase throughput of banyans is to partition the set of HOL cells into conflict-free subsets so that the cells of each subset can be routed simultaneously without conflict in one single banyan. In many proposed switches, all HOL cells

are issued to first banyan, successfully self-routed are retrieved, while all unsuccessful cells are re-issued to next banyan, and so on. This provides some sequential conflict resolution phases during which path reservations are processed in parallel within each banyan but sequentially iterated with respect to distinct banyans. Example of switches employing this approach is the *Tandem Banyan Switching Fabric* (TBSF) [9], *Piled Banyan Switching Fabric* (PBSF) [10], *Parallel-Tree Switching Fabric* (PTBSF) [11], and *Pipelined Switching Fabric* (PSF) [12].

In this paper we present a different conflict resolution approach in which paths are sequentially reserved within each banyan but path reservation is carried out in parallel with respect to distinct banyans. Input buffers may receive path reservation offers from parallel banyans and accept reservation from a given banyan if the corresponding path provides maximum sharing of switching elements (SEs) with respect to currently reserved paths. Thus, potential for future conflicts is reduced as the number of free SEs is wisely managed. This strategy promotes conflict-free path-clustering within each banyan. We present a method for finding valid paths for which we can efficiently determine the degree of sharing that is used as the basis for path selection. Using *path-clustering* as a method to obtain higher throughput, we propose a switch architecture called *Path Clustering Banyan Switching Fabric* (PCBSF). PCBSF is characterized by two main features: (a) to reduce blocking within each banyan, cell routes are setup so that they share the largest number of internal switches; and (b) to achieve very low cell loss a number of parallel banyans are employed. Both of these features are shown to result in a switch with very low cell loss ratio. We carry out performance evaluation of PCBSF under simulated *uniform traffic* and *ATM traffics*. An ON-OFF model is used for the generation of some realistic ATM traffic mixes.

The organization of this paper is as follows. Section 2 presents some background on recently proposed switches and features of the proposed approach. In Section 3 we present the path-clustering approach and the technique used for making reservation offers that maximize sharing of SEs. In Section 4 we present our proposed path-clustering banyan switching fabric. We also present the design of associated reservation, control, and data planes. Section 5 presents evaluation of the proposed switch architecture under uniform traffic and ATM traffic mixes and comment on throughput and robustness of the proposed switch. In Section 6 we conclude this work and outline some future research directions. Background information on multistage networks as well as the theory needed for path-clustering is confined in an appendix.

2 Switch anatomy and classification

The generic architecture of an ATM switch consists of: 1) input-output controllers, 2) switch fabric, and 3) management and control processor (see Figure 1). An input-output controller provides buffering, cell duplication for multicasting, cell processing, VCI translation, multiplexing traffic from several low-speed devices, and path connection requests and reservations through the switch fabric. An output controller provides buffering, VCI translation, demultiplexing and path selection. The switch fabric is essentially a device which routes cells from the switch input ports to its output ports. When a connection

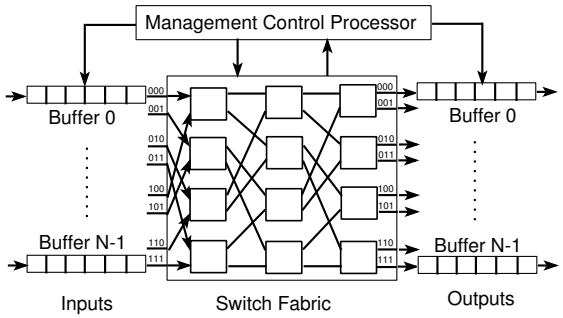

Figure 1: Model of an ATM switch

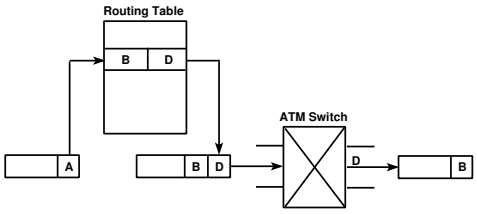

Figure 2: VCI translation

is set up between hosts, a virtual circuit is defined between source and destination. The connection establishment procedure initializes internal routing tables in the switches. The Management and Control Processor communicates with the I/O controllers and manages the switch operations. Upon entering an ATM switch, a cell's VPI and VCI fields are used to select an entry from the routing table of the switch that determines which output port the cell should be routed to. This is shown in Figure 2. A new VPI or VCI value may be placed in the cell header before forwarding the cell to the next switch.

ATM switch fabrics belong to two general categories: *time division* and *space division* as depicted in Figure 3. In *time division switches*, the switch fabric is time shared among several input output connections such as a shared memory or a shared medium. An example of shared memory switch is a dual ported memory that is shared by all inputs and outputs. Packets arriving on inputs are multiplexed into a single stream which is fed to the common memory for storage. Internally to the memory, cells are arranged into separate output queues, one for each output line. Cells are retrieved from the output queues sequentially, one per queue. The output stream is then multiplexed and cells are transmitted on the output lines.

In *space division switches* [13], the switch can support multiple, concurrent, non-conflicting connections such as the Crossbar Switch, Bus Matrix Switch, and switches employing multistage networks such as the Starlite and the Sunshine switches. Space division switch fabrics can also be easily implemented using unique path multistage networks such as *Omega*, *Delta*, or banyan networks, which is the approach used in this work. One common feature of space division switches is that the routing control need not be centralized but may be distributed throughout the switch.

Multistage interconnection networks (MINs) are among the most desirable and widely

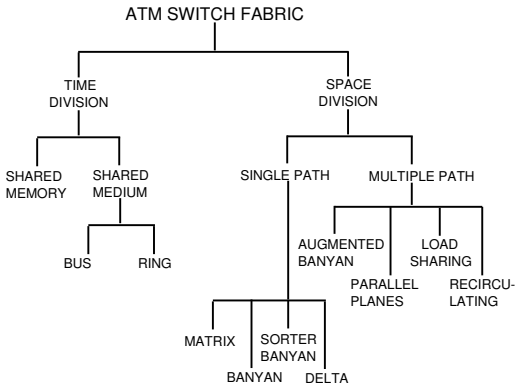

Figure 3: A taxonomy of the ATM switch fabric.

researched building blocks for space division fast cell switching. An ATM switch with $N = 2^n$ I/O controllers, a *switch fabric* (SF), and a *management control processor* (MCP) is shown in Figure 1 for the case of 8 inputs Ω network.

Banyan networks, a class of *multistage interconnection networks* (MINs), have several desirable features such as space division, self routing, low hardware complexity, and regular structure which make them suitable for VLSI implementation. Unfortunately, their throughput is far from being acceptable due to *internal* and *external* blocking. There has been a number of proposals to overcome this problem. One approach is to use internal buffering [14] to avoid loss of cells in the case of conflicts. Unfortunately, this increases the switch complexity as well as increases the *Head of Line* (HOL) queuing delays. The effect of HOL blocking can be decreased by allowing other queued cells to bypass the blocked HOL cell [15]. However, this leads to *out-of-sequencing* problem. Internal blocking can be avoided totally if we use a sorting network in front of the routing network [16]. The complexity of the sorting network is $O(N^3 logN)$.

The problem is to find an efficient approach to partition the set of HOL cells into conflict-free subsets so that each subset is assigned to a banyan where cells are switched without contentions. Unfortunately, there is no efficient method to solve this problem and several proposals have been made to provide partial solutions. The idea is to issue all HOL cells to one banyan, perform self-routing, retrieve cells which successfully reach their destination, and re-issue all unsuccessful cells to next banyan, and so on. The above assumes that some mechanism is available to distinguish successfully routed cells from others. We call this approach *Iterative Conflict Resolution* (ICR). The input cells of each phase are all unsuccessfully routed cells of previous phase (or initial) which means that the phases must operate sequentially over distinct banyans. Cell routing in each phase is done in parallel because all input cells are concurrently routed over the banyan.

Example of switches employing the ICR approach is the *Tandem Banyan Switching Fabric* (TBSF) [9] in which cells are applied to banyans arranged in series. Conflicting cells are misrouted on any free path and one header bit is set. By properly adjusting the number of banyans in series, the CLP can be made as low as needed but at the cost of relatively large propagation delays due to the sequential banyan structure and propagation of entire cells. In *Multi-Parallel Banyans* vertical interconnections among parallel banyans

are used to reduce the above mentioned sequential propagation delays. The *Piled Banyan Switching Fabric* (PBSF) [10] and *Parallel-Tree Switching Fabric* (PTBSF) [11] are two examples of this class. The vertical cell forwarding significantly reduces the propagation delay. However, complex hardware and large number of interconnections are needed to produce an acceptable CLP. A recent switch that optimizes the propagation delays is the *Pipelined Switching Fabric* (PSF) [12] in which only cell headers are used during path reservation phases. Due to the sequential nature of phases only one banyan called *control plane* is needed for reservation of paths. The path state information is assigned later to simple *data planes* for payload switching.

The principle of ICR consists of a number of sequential reservation phases (banyans) so that conflict resolution among cells is done in parallel during each phase. Path reservation is carried out in parallel as all HOL cells are applied at the input of the banyan, where conflicting cells are re-issued in next iteration to another banyan. Thus reservations are parallel within each banyan but reservation over distinct banyan planes must be done in a sequential manner. This might not be the best approach for permanent circuits and switched virtual circuits for which new switching requests must be submitted on arrival of new cells with no memory of previously established paths. Our objective is to investigate a new technique that uses incremental path reservation based on previously established connections. Path reservation is performed sequentially within each banyan but allows multiple banyan planes to be concurrently reserved. The destination ports of all HOL cells are used to find all conflict-free paths that can be initiated from all possible input buffers. Each such a path is characterized by the number of partially allocated SEs used from source to destination. Some HOL cells receive reservation offers from parallel banyans. The offer for the path that has the largest number of partially allocated SEs reaches the input buffer at the earliest time. Such path allocation strategy leaves the largest possible number of free SEs. Thus, the potential for future conflicts is reduced as the number of free internal switches is wisely managed. We call this approach *path-clustering* where paths are sequentially clustered for each banyan but all banyans compete for HOL cell reservation in parallel.

3 Path-clustering based routing

Switch architectures based on the use of several banyans arranged in series or in parallel have been proposed to increase the switch throughput. Parallel banyan networks, besides improving the throughput, they provide better delay characteristics than their tandem counterparts. Furthermore, parallel banyan networks are multipath networks, which means that the routing from source s to destination d may take any path out of a number of paths for each destination. Since the cells come time-multiplexed, these architectures exploit space parallelism as well as time parallelism (pipelining).

Banyan networks follow a distributed routing strategy whereby each SE will be examining locally a particular bit of the destination port address and forwarding the cell to its upper output if destination bit is '0', or to its lower output if the bit is '1'. The destination output port number of the switch fabric is fixed before the cell is supplied to the fabric. Typically, the destination port is selected randomly among the available

ports and regardless of the routes that are used by other cells. This is the major cause of internal blocking. To reduce blocking, one possible approach would be to proceed as follows: 1) partition the cells into groups of non-conflicting cells for a given banyan, and 2) routing the payload for those non-conflicting cells. The above process is pipelined in time.

Clearly cells may be selected for routing on the basis of 1) forming a non-conflicting group of cells of the largest possible size, 2) allocating and reserving paths, and 3) routing their corresponding payload on their reserved paths. Some HOL cells may occasionally fail in enrolling in one of the parallel banyans. Such cells will be lost if we do not keep them for a subsequent routing attempt. Therefore, some input buffering is required to smooth out the transient behavior of the switch. Obviously combining all the benefits of space parallelism, pipelining (time parallelism), and input buffering is expected to produce higher throughput than an architecture that is based on a subset of the above features.

To increase the throughput of parallel banyan architectures we propose an approach that uses additional hardware to achieve the following objectives:

- All the cells issued to a given banyan at a particular time slot should have neither internal conflicts nor output conflicts.
- A new cell that is issued to a given banyan should be routed so as to share the largest number of partially used SEs with the cells that are to be transmitted during the same time slot. The selected cell to be removed from some input buffer is a cell whose single path from source to destination, for a given banyan, passes through the largest number of partially used switches.
- The cell selection should be implemented in a *distributed manner* and at the lowest level to allow the least latency in order to minimize the waiting time of the cells at the input buffers.

This approach requires that the global operation (slot operation) of the switch be performed by a central routing controller (CRC). The CRC will provide a mechanism to synchronize the operations of each banyan within a time slot. Cell path reservation operates in a serial fashion with respect to any given banyan. However, setting up the required paths according to the cell destination must be done in a selective and distributed manner. The selection of paths is based on maximizing the number of shared SEs. During this setup phase, only the cell headers need be examined. Cell routes are sequentially established. However, only the n-bit headers are propagated through during the route selection step. Furthermore, once routes for all the HOL cells are established, input-to-output circuits are established through the banyan switch so that cells are circuit-switched.

An HOL cell has one unique path (if any) for each available banyan. The path exists if all the links it needs and all the switch states, on the route, are set in the appropriate state. In general a cell may have different paths on distinct banyans. Each of these potential paths passthrough n SEs, where each can be partially or totally free. Selecting the Route that maximizes the number of shared internal SEs allows maximizing the number of SEs that remain free in a given banyan. This increases the probability of finding conflict-free paths in subsequent path allocation. Therefore, this path selection strategy is one approach to the minimization of future blocking.

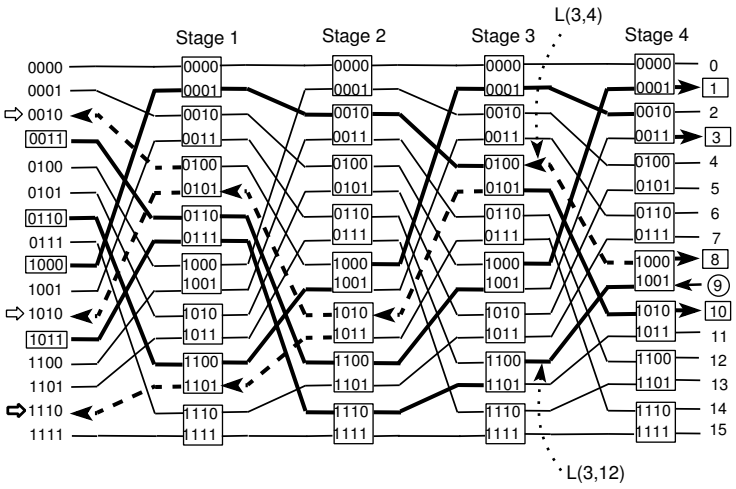

Figure 4: Finding conflict-free paths for a set of assigned paths

A group of cells that are sequentially issued to a banyan switch according to this approach is said to be *maximally compatible* (MC). A cell route decided on the basis of this criterion is referred to as *maximum compatible route* (MCR). The process of finding MCRs allows partitioning of the available cells into groups of non-conflicting cells for the purpose of maximizing the switch throughput.

The implementation of dynamic search and selection of MCRs requires the use of feedback from the banyan switch in order to select an MCR for each input cell. To minimize routing latency, the central controller should be implemented in hardware at the lowest level. In the following, we present the design strategy of the proposed path-clustering technique.

3.1 Finding conflict-free paths

Consider a banyan network with a number of established paths, the problem is to find out whether an HOL cell $C(s, d)$ that is present in buffer s can be switched without internal or external conflicts to output buffer d of a given banyan switch. For this we present an approach that provides information to all switch inputs from where a given switch output can be reached without conflicts with respect to a number of currently established paths. We further tune this approach so that it becomes possible to select the non-conflicting path that uses the largest number of partially busy switches among all non-conflicting paths.

Consider a $N = 2^n$ inputs and outputs banyan network (MIN) having n stages built of 2×2 SEs. There are $N/2$ switches in each stage. The switches of two successive stages are connected through N links which are subject to some permutation. The reader may refer to the *Appendix* for additional features about MINs. Figure 4 shows a 16 inputs and outputs banyan network (Omega) for which the stages are interconnected through a perfect-shuffle permutation σ. Consider the second link between stages 1 and 2. Its position at output of stage 1 is 0001, thus its position at input of stage 2 must be

$\sigma(0001) = 0010$.

Denote by $L(k, i)$ the link that is joining the ith output of stage k to input $\sigma(i)$ of next stage $k + 1$. Each link $L(k, i)$ is associated a status bit $st(k, i)$ that is 0 if $L(k, i)$ is being used for transferring some cell and 1 if $L(k, i)$ is free. Given a destination d, the problem is to find out all the inputs $\{s\}$ from where a cell $C(s, d)$ can be issued over a conflict-free path from source s to destination d. To solve this problem we need to find all conflict-free paths from destination d to all possible sources $\{s\}$.

Assuming all paths were originally free in the banyan, to exit stage 4 at output 1001 a cell should have come from either links $(L(3, 4), L(3, 12))$ of stage 3 and, either of links $(L(2, 2), L(2, 10), L(2, 6), L(2, 14))$ of stage 2 and, either of links $(L(1, 1), L(1, 9), L(1, 3), L(1, 11), L(1, 5), L(1, 13), L(1, 7), L(1, 15))$ of stage 1, and either of the 16 inputs. We assumed that all paths were originally free but if some of the paths were already reserved (cannot be used) then some of the potential paths cannot be used to reach the needed destination.

For example, the 16-inputs banyan network (Omega) shown in Figure 4 has four currently used paths whose source-destination pairs are $(3, 1)$, $(6, 3)$, $(8, 10)$, and $(11, 8)$ which are indicated by solid lines. The sources and destinations involved in these paths are indicated by boxes in the above figure. Assume we want to find all source inputs $\{s\}$ for which there is a conflict-free path $(s, 9)$. can be established without disturbing currently assigned paths. For this, we start from output 9 of stage 4 and move backward to find all potential paths from where to reach the same output 9. At output of stage 3, we only find link $L(3, 4)$ because the other link $(L(3, 12))$ is already reserved for path $(11, 8)$. At output of stage 2, we find only link $L(2, 10)$ because $L(2, 2)$ is reserved for path $(8, 10)$. At stage 1, we find two possible links which are $L(1, 5)$ and $L(1, 13)$. $L(1, 5)$ can be reached from the network inputs 2 and 10. Thus paths $(2, 9)$ and $(10, 9)$ are both conflict-free. Finally, link $L(1, 13)$ allows finding one additional conflict-free path which is $(14, 9)$. Notice, that to find all possible conflict-free paths, we move backward and select all free paths along a broadcast tree.

We can establish the rules for selecting these links and their status lines by using Lemma 2 in the *Appendix* which gives the tree shown on Figure 5. The summary of Lemma 2 is as follows. A cell that exits the banyan at position $d_{n-1} \dots d_0$ should pass the ith stage at any switch output whose position is $x_{n-i-1} \dots x_0 d_{n-1} \dots d_{n-i}$, where $x_{n-i-1} \dots x_0$ may take any possible binary combination out of 2^{n-i}. Therefore, to exit at destination $d = d_3 d_2 d_1 d_0$, a cell requires that link $L(3, d_3 d_2 d_1 d_0)$ be free and, links $L(3, 0 d_3 d_2 d_1)$ or $L(3, 1 d_3 d_2 d_1)$ be free, etc. Figure 5 shows how the links should be chosen for finding a path from some input to destination d. A path from some source s to destination d is formed by a sequence of links. In Figure 5, a vertex v having two incident edges connecting it to vertices u and w corresponds to the output of one SE and u and w correspond to the same switch inputs. A vertex v having a single edge connecting it to vertex u corresponds to a link connecting two stages.

A free path is characterized by status bit 1 for all of its links. Therefore, it is sufficient to *And* all the link's status bits along a path in order to find whether a path is free or not. For this we refer to Figure 5 and assume all the vertices represent status bits of the corresponding links and propagate a logic 1 from top to bottom in the tree. This cor-

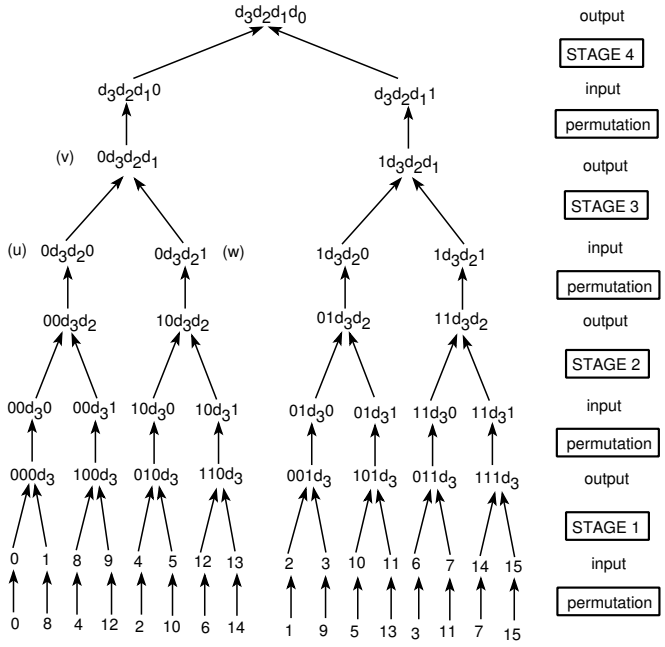

Figure 5: Symbolic relation of all links for a given destination

responds to a backward (right-to-left) propagation compared to the forward propagation of the cells through the banyan switch. All bottom entries that carry a logic 1 will then correspond to conflict-free paths. The minimum condition to exclude a given path is to find at least one busy link at any level of the same path.

More specifically, the destination $d = d_3d_2d_1d_0$ allows selecting one status bit out of 16 for the links of stage 4. There are 16 links connecting any two successive stages.

For stage 3, we divide the 16 status bits into 8 groups and each group consists of two status bits $st(3, i)$ and $st(3, j)$ so that the 3 least significant bits (lsb) of i and j are $d_3d_2d_1$ and their most significant bits are complement of each other. In other terms $d_3d_2d_1$ is to be used for selecting one group of status bits out of 8 as shown in the multiplexer (labeled 1 out-of-8) in Figure 6. A *memory plane* is an array of boolean memories that is used for the storage of status of all links in the banyan. The memory plane will be presented in more detail later.

For stage 2, we have 4 groups of status bits and each group consists of 4 status bits $st(2, i)$, $st(2, j)$, $st(2, k)$, $st(2, l)$ so that the two lsb bits of i, j, k, and l are d_3d_2 and their two msb bits take all possible combinations. Here, d_3d_2 can be used to select one group of four. This process continues to the bottom of the tree.

The above decision tree can then be dynamically generated using few multiplexers as shown in Figure 6 where the group of status bits are shown below the multiplexers. In Figure 6 the blocks denoted by E are called expanders and each is used here to stimulate one 2×2 SE. An expander corresponds to backward propagation of one 2×2 switch. Each expander has status inputs u, v, and w and generates two outputs O_{uv} and O_{uw} as shown in Figure 7-a and -b. Input u is the status of some output link (upper or lower output) to some 2×2 switch and v and w are the status bits of the incident links to the same

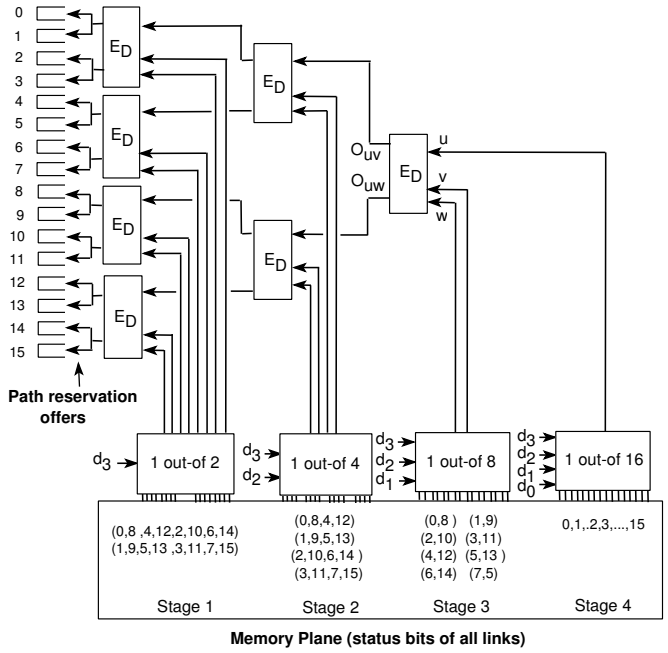

Figure 6: Finding free paths using backward propagation over a 16 inputs banyan

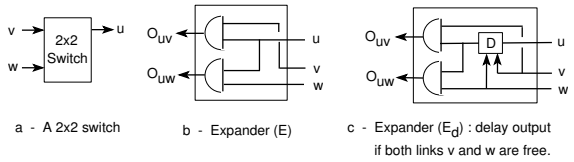

Figure 7: Correspondence between SE and backward propagation expanders

switch. Output $O_{uv} = 1$ only if $uv = 1$, which means that the link corresponding to v is free and there is a free-path from the current expander to the destination because $u = 1$. In other terms, when $O_{uv} = 1$ then there is a free-path from v to u in the corresponding 2×2 switch. A similar situation occurs when $O_{uw} = 1$. Therefore, the expander hardware shown in Figure 6 gives all the free-paths that can be reached from any source for which the overall expander output is 1. The expander structure allows finding all conflict-free paths from the input buffers where cells are waiting to be serviced to a given switch destination.

To find non-conflicting paths, working destinations d are generated from a given set so that feedbacks through the expanders are forwarded to each input buffer for each given destination. An input buffer that receives a feedback (conflict-free) from some expander knows that there is a free path from its output to the working destination. Therefore, the buffer can request reservation of offered conflict-free path if the destination of its HOL cell matches the working destination. This will guarantee that all cells issued to a given banyan are non-conflicting.

3.2 Finding maximally compatible paths

We now introduce an approach to select an input buffer so that the path (s, d) required by the output cell $C(s, d)$ shares the largest number of partially busy switches among all enabled buffers. Note that $u = 1$ occurs when a switch output is free (Figure 7-a and b). This indicates that the switch inputs v and w can be in one of the following states: 1) both switch inputs are free ($v = w = 1$), or 2) one of the switch inputs is free ($v \oplus w = 1$). The case $v = w = 0$ cannot occur because $u = 1$ necessarily implies that there is at least one free input. One needs to distinguish between the case of a completely free switch ($v = w = 1$) and the case of a partially busy switch ($v \oplus w = 1$). For this we delay the propagation of u when the condition $v = w = 1$ occurs.

The delay element shown in Figure 7-c causes u to be delayed by an amount of D seconds. The result is that the expanders feedback to an input buffer located at position s (Figure 6) will be delayed by an amount of time that is proportional to the number of fully free-switches that the path (s, d) has. In other terms, the earliest feedback to reach some input buffer located at s corresponds to a path (s, d) that uses the largest number of partially busy switches. As more than one such path can be found, ties can broken by enabling the lowest numbered input or randomly.

For example, the 16-inputs banyan network shown in Figure 4 has four used paths $(3, 1)$, $(6, 3)$, $(8, 10)$, and $(11, 8)$. Assume the expander structure is activated for destination $d = 9$. The conflict-free paths found are $(2, 9)$, $(10, 9)$, $(14, 9)$. Note that paths $(2, 9)$ and $(10, 9)$ pass through two partially busy switches (stages 3 and 4), but path $(14, 9)$ passes through three partially-busy switches located in stages 1, 3, and 4. Notice that conflict-free paths are marked by dotted lines in Figure 4. In this case, the expander output will be 1 for inputs 2, 10, and 14 that are the network inputs. These are marked by the arrows on Figure 4. The fastest (least delay) backward propagation reaches input 14 before the other logically valid paths that reach buffers 2 and 10. If input buffer 14 has an HOL cell with destination $d = 9$, then a path $(14, 9)$ can be reserved on the corresponding banyan. In this case the selected path is the one that uses the largest number of partially busy switches.

Therefore, the use of the expander with the delay element enables selection of an input buffer located at s from where a free-path (s, d) that uses the largest number of partially busy switches can then be established. This selection method is intended to promote future non-blocking because it leads to the clustering of currently established paths by using the largest number of partially busy switches, thus leaves unused the largest possible number of free 2×2 switches which is our proposed strategy to increase throughput of banyan networks.

3.3 Correctness of the output

The backward propagation scheme consists of $2^{n-1}-1$ expanders that are interconnected in a tree structure as shown in Figure 6. There are $n - 1$ expanders on each backward path. The wires interconnecting the expanders from first stage (backward) to last introduce delays, which depend on wire length. Therefore, the need to set up the gate delay D so that variations in wire length does not cause incorrect result. A path with the largest

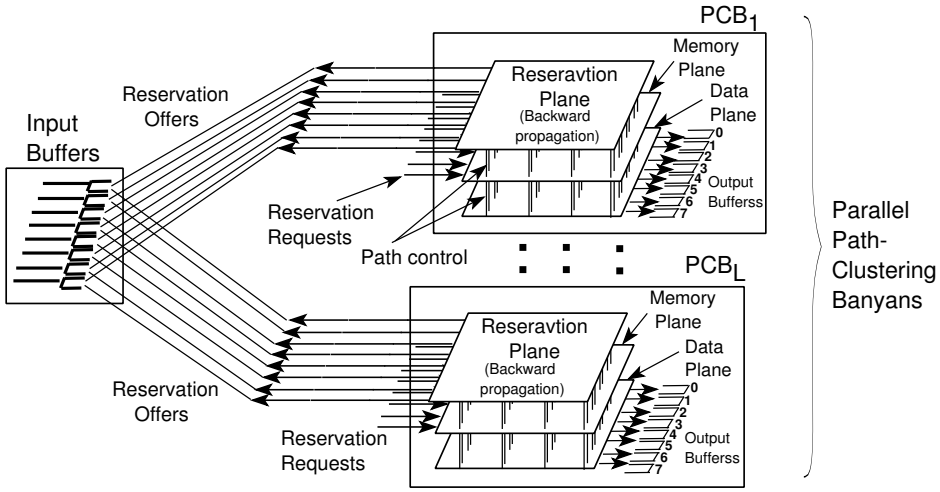

Figure 8: Architecture of the Path Clustering Banyan Switching Fabric

number of partially busy switches must have its output delayed the least compared to another path with a smaller number of partially busy switches. The regularity of the topology reduces the complexity of the wire delay problem.

Denote by t_{lmax} and t_{lmin} the time delays caused by the longest and shortest wires that connect two successive stages, respectively. We assume a delay t_{in} due to wiring for both outputs of the expander. The maximum variation of the delay due to wiring is then $t_{lmax} - t_{lmin}$ which should be below the gate delay D in order to yield correct result, i.e $t_{lmax} - t_{lmin} < D$. However, increasing the gate delay leads to increasing the response time of BPM. The least value of $D = t_{lmax} - t_{lmin} + \epsilon$, with ϵ as small as possible, should then be adopted in practical implementations in order to guarantee a correct result as well as the least response time.

It is easily seen that if the time ordering for the various signals is maintained from the input of one expander to the input of the next expander, then the order will also be maintained for the backward propagation module. This is true because all stages maintain time ordering and delays are additive along any path.

4 Operations of the switch

The design of the Path Clustering Banyan Switching Fabric PCBSF is based on three architectural features used as the basis to achieve high throughput which are: 1) parallel banyan planes with multiple outlets, 2) path-clustering to improve the throughput of each banyan, and 3) input buffering to reduce cell loss during transient traffic patterns. The PCBSF has a number of parallel banyan planes. Each banyan plane with this organization is called *Path Clustering Banyan* (PCB) which is the basic building block in the PCBSF. A PCB consists of a back-propagation module, a state switch memory, and a data plane to switch cells from inputs to outputs as shown in Figure 8. The PCBSF shown has 8 input buffers, L PCBs, and $8 \times L$ output buffers.

The back-propagation module selects subsets of non-conflicting cells from sources to destinations with respect to already reserved paths. The memory plane is used to storing the status information (free/busy) of all inter-stage links. The data plane has 2^n inputs and 2^n outputs. The data plane is a pure combinational circuit since the state of all 2^{n-1} switches have already been set during the path reservation phase which follows the back-propagation phase. The reservation offers and reservation requests shown in Figure 8 will be described later.

The switch operates in two modes: 1) path reservation phase, and 2) cell routing phase. In the path reservation phase, all PCBs compete in performing path-clustering in an attempt to maximize their load as the main strategy to maximizing overall throughput. In cell routing phase, the data payload of non-conflicting cells are routed through specific data planes by using the routes reserved during the path reservation phase. The input buffers supply the path reservation phase with destination of their HOL cells whose required paths can actually get reserved on some data plane. A loss occurs when a cell is issued from external trunk to some full input buffer. The combination of input buffering, parallel arrangement, and path-clustering allows PCBSF to deliver high throughput. In the next subsection we describe the general organization of the PCBSF.

4.1 Switch organization

Cells arrive at some rate to the input buffers which are FIFO buffers of depth (M). Arriving cells are queued and only HOL cells are candidates for routing. The PCBSF follows a slotted operation, where the arrivals of new cells are considered only at the beginning of a new time slot. To load balance L parallel PCBs of the switch, the N input buffers are divided into L groups, where the ith group of input buffers is denoted by G_i. The allocation of a given group G_i to a given Path Clustering Banyan PCB_j is done in a round robin manner with respect to reservation slots. Figure 9 shows the reservation cycles k and $k+1$ for a 128-input PCBSF having 4 PCBs. Each reservation slot consists of L steps $(0, \ldots, L-1)$. In step 0 of reservation slot k, PCB_0 is allocated input buffers $0, \ldots, 31$, PCB_1 is allocated $32, \ldots, 63$, and so on. The assignment follows a round robin distribution in each reservation slot as shown in Figure 9. Therefore, in one reservation slot all sets of input buffers would have been allocated to each PCB_j. Note that while the back-propagation of each PCB is busy during reservation slot k the payload of reserved slot $k-1$ is being switched on the PCB data planes. The slot time is the sum of reservation slot time and payload switching time which are assumed to be equal in Figure 9. Since the reservation slot overlaps with payload switching, the PCBSF is pipelined.

In reservation slot k, PCB_j is allocated set G_i of input buffers such that $i = j + l \ mod \ L$, where l is the step number $0 \leq l \leq L-1$. For example when $N = 16$ and $L = 4$, in the first reservation the allocation is (PCB_j, G_j) and $0 \leq j \leq 3$, in second reservation $(PCB_i, G_{i+1 \ mod \ 4})$, etc.

During one reservation slot, each PCB sequentially polls all the input buffers of its current set of input buffers with the objective of making the largest number of possible path reservations. In a reservation slot, every PCB_j performs exactly N/L back-propagation attempts on the basis of the destinations of the HOL cells in its allocated group of input

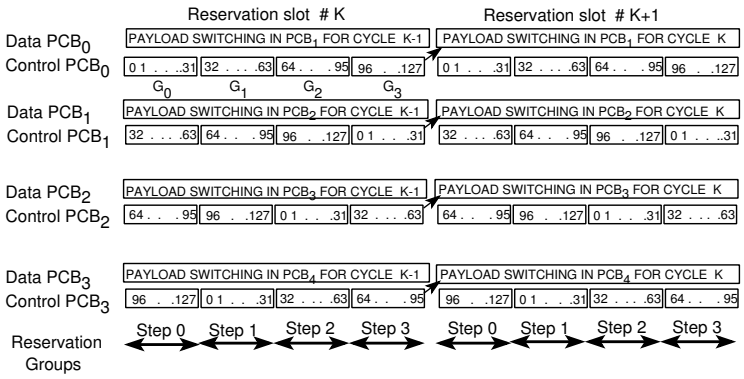

Figure 9: Reservation slots and pipelined operations of the PCBSF

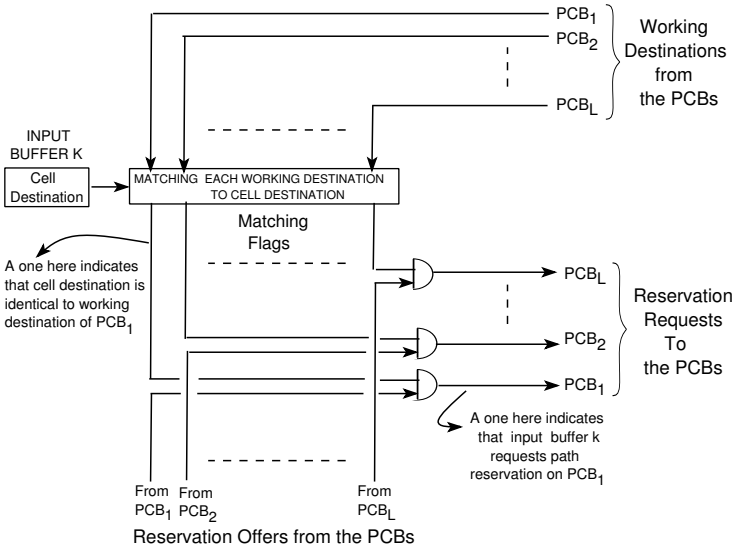

Figure 10: Reservation offers and reservation requests for one input buffer

buffers G_i. In other words, the destinations of the cells for each input buffer of G_i are used to initiate back-propagation on PCB_j. Following back-propagation, the generated feedback out of PCB_j is sent to all input buffers, not only those buffers within G_i, and the HOL cell which is maximally clustered with respect to current path reservations is selected and must be allocated a path on PCB_j. In other words, the reservation slots are only intended to assign non-overlapping working sets of destinations to avoid concurrent back-propagation on different PCBs for the destination of HOL of the same input buffer.

Since all PCBs have identical back propagation hardware, offers for conflict free path reservation will be concurrently presented at all inputs buffers. Figure 10 shows the reservation offers for one input buffer holding an HOL cell. The destination port numbers used to initiate each back-propagation plane of each PCB are forwarded to all input buffers to enable matching of currently working destinations with the cell destination. This matching is done in block *destination matching* of Figure 10. Each input buffer receives offers for path reservation from potentially all the PCBs. If there is a destination

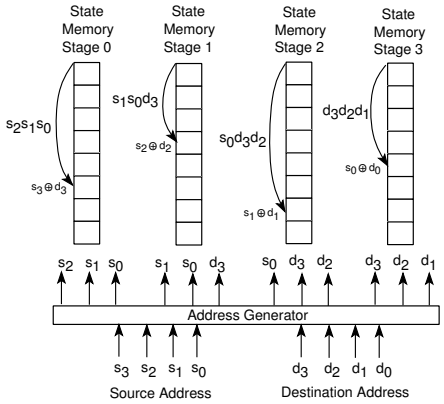

Figure 11: Updating switch state memory for path reservation

matching for a given input buffer and there is a reservation offer from the corresponding PCB, then a reservation request is generated for the current HOL cell. Concurrent offers from different PCBs usually arrive at different times to a given input buffer, and the reservation offer that comes first is accepted, which is the basis of the path-clustering strategy described in the previous section. Notice that an offer from some PCB to some input buffer means that a path from this input to the destination of its HOL is conflict free with respect to current reservation on this PCB. Therefore, if an offer is accepted by some input buffer, it must inhibit all other input buffers from performing concurrent reservation for the same PCB. The path with the largest number of partially used switches from some input buffer to its desired destination causes its offer to reach earlier than the offers of other paths with a smaller number of partially busy switches. The route corresponding to the earliest reservation offer is set, which in turn inhibits all later reservation offers for the same PCB destination.

4.2 Reservation of paths

Input buffers receiving a reservation offer at the end of a back-propagation cycle request path reservation on the PCB that generated the routing offer. Assume an input buffer s receives a routing offer from PCB_j for its HOL cell whose destination is d. This means that a path (s, d) is conflict free on PCB_j which requires that buffer s attempts reservation of the desired path on PCB_j prior to servicing another HOL cell from this input buffer. Notice that the first input buffer to receive a routing offer from a given PCB will immediately perform path reservation, thus disabling all other buffers which received delayed offers from the same PCB from proceeding into concurrent path reservation. Now we explain the process of path reservation.

Each switching element (SE) in the PCB can perform the straight or swap permutation. Thus, its state is only one bit. There are 2^{n-1} switches in each of the n stages of the PCB which allow organizing the state memory of the PCB switches as an array of $2^{n-1} \times n$ flip-flops, where 2^{n-1} is the number of SEs in each stage and there are n stages.

Figure 11 shows the state memory as an array of (4×8)-bits memory for a 16×16

banyan (Ω network). Each stage is one column of 2^{n-1} one-bit memories. The state memories are located in the memory plane shown in Figure 8.

Assume a cell reaches a switch which is already set in one of its two possible states (straight or swap). We need to find out under what condition the cell can pass the switch without conflicts. In other words, the cell may arrive to the switch at its lower or upper input and may request to exit the switch at its upper or lower output. Lemma- 3 of the *Appendix* determines the relationship between the cell entry and exit positions and the previous state of the switch in order not to cause any conflict. The state of a 2×2 switch can either be 0 (straight) or 1 (swap). The summary of Lemma 3 is as follows. In a banyan network (Ω_n), a cell that reaches the ith stage at some input of switch $s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1}$ passes the switch without conflict if and only if the switch state is $s_{n-i} \oplus d_{n-1}$.

Assume the path for which the reservation is to be done is (s, d). The addressing of the state memory is done according to Lemma 3 which tells us that we must set the state of the 2×2 switch that belongs to path (s, d) in the ith stage to $s_{n-i} \oplus d_{n-1}$. This completely defines the value to be stored in the switch memory, but now, we need to find the address of the one-bit state flip-flop.

Using Lemma 1, we know that the position of the cell at the output of the ith stage is $pos_i(s, d) = s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i-1} d_{n-i}$. Each 2×2 switch has two inputs. Thus, the position of the switch is $pos_i(s, d)/2$ which is $pos_i(sw) = s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i-1}$. The switch address is taken here for the address of the switch state bit in the ith state memory column. The position of the path at each stage is used here as a pointer to the state memory. Specifically, Lemma- 1 is applied here to the generation of addresses of the state memory for each SE that belongs to the path. This is shown in Figure 11 in block *address generator* that is passive block which combines s and d to generate the needed addresses. Now the address of each SE memory of a path (s, d) is available for setting up a reservation or for cancellation of paths.

Figure 11 shows the addresses of the state memories that are generated for the reservation of a path (s, d), where $s = s_3 s_2 s_1 s_0$ and $d = d_3 d_2 d_1 d_0$ Therefore, the position of the state memory bit is obtained by simple combination of s and d as shown in the figure. In summary, path reservation consists of setting the state of flip-flop $pos_i(sw)$ to $s_{n-i} \oplus d_{n-1}$ in the ith stage, in parallel on all the n stages or memory columns.

Notice that switch $pos_i(sw)$ of the ith stage is either (1) already set to value of $s_{n-i} \oplus d_{n-1}$ by some previous path reservation or (2) being set for the first time during this time slot. In the first case, overwriting $s_{n-i} \oplus d_{n-1}$ is an indicator of non-conflicting path reservation. Notice that it is impossible to find the state of this switch already set in a value other than $s_{n-i} \oplus d_{n-1}$ because the current reservation of (s, d) is conflict-free for the PCB. In the second case, the value of $s_{n-i} \oplus d_{n-1}$ is being written for the first time during the current time slot which means that a future path reservation may use the same switch without changing its state. Finally, we notice that one clock is required to make path reservation according to the above parallel state setting scheme.

4.3 Cell routing

Initially all the switch states are set to straight at the start of the reservation phase. At the end of reservation, most of the switches in any given PCB have already been set as required by all buffered cells that were HOL or became HOL during the current time slot. The cell routing phase consists of routing the payload of all the cells for which path reservation succeeded on one of the available data planes. The payload of each such cell is routed from data plane input to data plane output through a combinational logic path. The path is established based on the state of all the switches in all the stages which have been written in the previous reservation phase. The routing delay is the same for all routed cells because all payloads in all data planes traverse the same number of gates. The routing delay depends on the number of stages n, gate delay in each stage, and size of the payload (P). One stage delay is identical to one switch delay which is two gates because the switch can either be in straight or swap states. An estimation of the delay is $2log(N)P$.

5 Performance evaluation

In this section we study the effect of various architectural features of the proposed switch on its performance. The performance measure of interest here is the cell loss ratio. The features that are studied are,

1. Effect of switch size on cell loss.
2. Effect of the number of data planes on cell loss.
3. Effect of path-clustering on the cell loss.
4. Sensitivity of the switch cell loss ratio to the traffic intensity.
5. Sensitivity of the switch cell loss ratio to the number of input buffers.

Because of the complex structure of the switch router, we resort to simulation for a thorough study of the aforementioned architectural features. A simulator was implemented in the C language and used to estimate the cell loss ratio of the switch under uniform as well as ATM traffic load conditions. Figures 12 to 18 summarize simulation results collected under varying uniform traffic loads, while Figures 19 to 21 show performance estimations under ATM traffic loads. Notice that all plots are semi-log plots. Because of excessive runtimes, we were forced to limit each simulation run to 10^8 cells.

5.1 Performance under uniform traffic

Figure 12 shows the variation of cell loss ratio with the number of data planes, for different switch sizes and at full load. Results are for the above *path-clustering* concept. The probability of cell loss drops from 0.65 with one data plane to about 10^{-2} with five data planes. This is a gain of two orders of magnitude. No cell loss was observed with six data

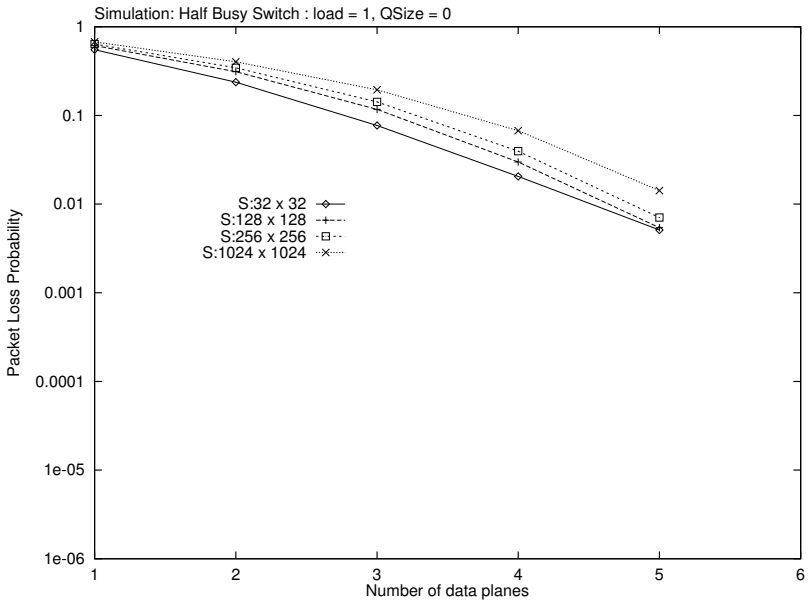

Figure 12: Loss in PCBSF pipeline without input buffering versus the number of data planes

planes and for a simulation load of 10^8 cells. Hence, at full load, and with six data planes, the cell loss ratio is predicted to be less than 10^{-8}.

Figure 13 illustrates the cell loss performance of the *path-clustering* switch as a function of the number of data planes, when each input port is equipped with a buffer of five cells.

The simulation was conducted at full load and for different switch sizes. In contrast to Figure 12, where input buffering was not available, we notice dramatic improvement in performance only when more than one data plane is used. When the switch consisted of a single data plane, there was no noticeable decrease in cell loss probability because of the availability of five buffers at each input port. Input buffering is advantageous only when two or more data planes are available. Each additional plane improves the cell loss ratio by several orders of magnitude. For example, for a 1024x1024 switch with 3 data planes, cell loss was equal to 10^{-1}, and with 4 data planes it decreased to 10^{-4} (see Figure 13). Contrast these numbers with those plotted in Figure 12.

Figures 14 and 15 show simulation results of the switch when a simple banyan is used instead of the *path-clustering* banyan. We notice that the degree of cell loss improvement is a function of the number of data planes and to a lesser extent of switch size. Comparing Figures 12 and 13 with Figures 14 and 15, we observe there is noticeable drop in cell loss ratio. This performance improvement is attributed solely to the path-clustering feature. For example, for the path-clustering strategy (refer to Figure 13), the cell loss ratio for a 32x32 switch with 4 data planes is less than 10^{-8}. In contrast, for the simple banyan control strategy, the cell loss ratio was larger than 10^{-6} under the same load condition. This is in a way expected, since a path clustering banyan (PCB) has better throughput characteristic than a simple banyan. Hence, the addition of each PCB would result in a larger boost of throughput performance than the addition of a simple banyan. Therefore, a smaller number data planes would be required if PCBs are used instead than simple

banyans.

Figures 14 and 17 give cell loss simulation results for the simple banyan and path-clustering strategies respectively, for various sizes of the input buffers. A close examination of Figures 14 and 17 reveals another desirable performance behavior of the path-clustering strategy. The Figures indicate that, with the path-clustering strategy, the addition of each data plane leads to a much larger reduction of cell loss than when simple banyans were used. For example, for a 32x32 switch with 3 data planes and a buffer size $B = 2$ cells per port, cell loss ratio is in the order of 10^{-2} and 10^{-1} when path-clustering banyan and simple banyan were used respectively. Now, when buffer size is increased to $B = 10$, cell loss ratio decreased to about 10^{-8} when path-clustering is adopted and to about 10^{-4} for the simple banyan strategy.

6 Evaluation under ATM traffic

ATM networks are expected to support a wide range of traffic sources requiring one of the five service categories, CBR, VBR-rt, VBR-nrt, ABR, or UBR. During the lifetime of a virtual connection, a traffic source will be in one of two states, *active* or *idle*. During the active state the source is transmitting cells at some given rate. Depending on the type of source, each active state may be followed by an idle period during which the source is silent. This model is known as the *ON-OFF model*. There is a general belief that, with the exception of CBR sources, this *ON-OFF* model provides an acceptable approximation to ATM traffic sources. For CBR-sources, there is no idle period.

The cells generated during the same ON-period form a *burst*. Successive active and idle periods are assumed statistically independent. Furthermore, the length of the active period as well as that of the idle period are exponentially distributed, with average lengths $\frac{1}{a}$ and $\frac{1}{b}$ respectively.

Several parameters have been identified, which together, completely characterize an ON-OFF A traffic source. These are,

p**: Peak cell arrival Rate.** This is the cell arrival rate when the source is in the ON state. $p = \frac{1}{T}$, where T is the time between two consecutive cell arrivals during the ON period.

m**: Average cell arrival Rate.** This is the cell arrival rate over the entire lifetime of the connection of the source. $m = p \times \frac{a^{-1}}{a^{-1}+b^{-1}}$

β**: Traffic burstiness** $= \frac{p}{m}$**.** A large value for this parameter indicates a very bursty source.

t_{on}**: Average duration of active state (Burst).** This average is computed over the entire lifetime of the connection. $t_{on} = \frac{1}{a}$

Typical values for the traffic parameters for the various traffic source types are summarized in Table 6 [17, 18]. In our simulation study, we assumed that the PCR, t_{ON}, and β are known for each source. Furthermore, as recommended by ITU-T, we assumed that

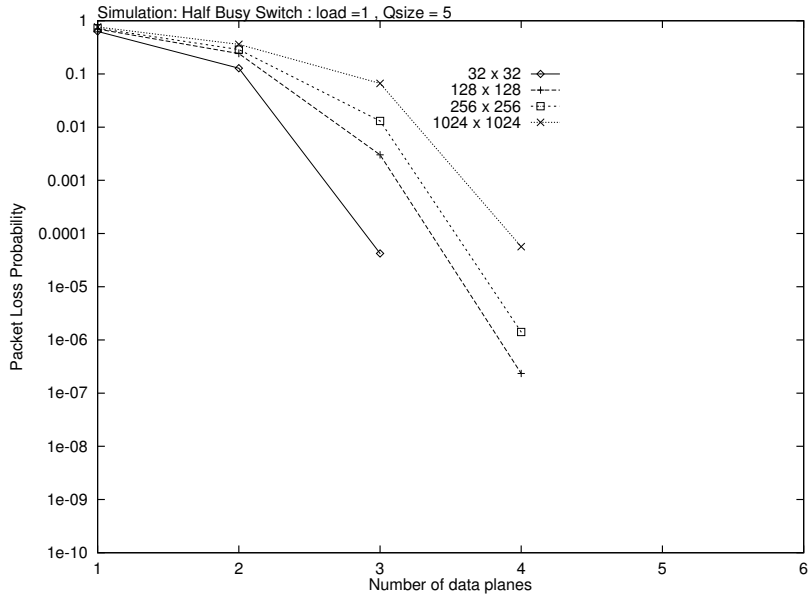

Figure 13: Loss in PCBSF pipeline with 5 input buffers versus the number of data planes

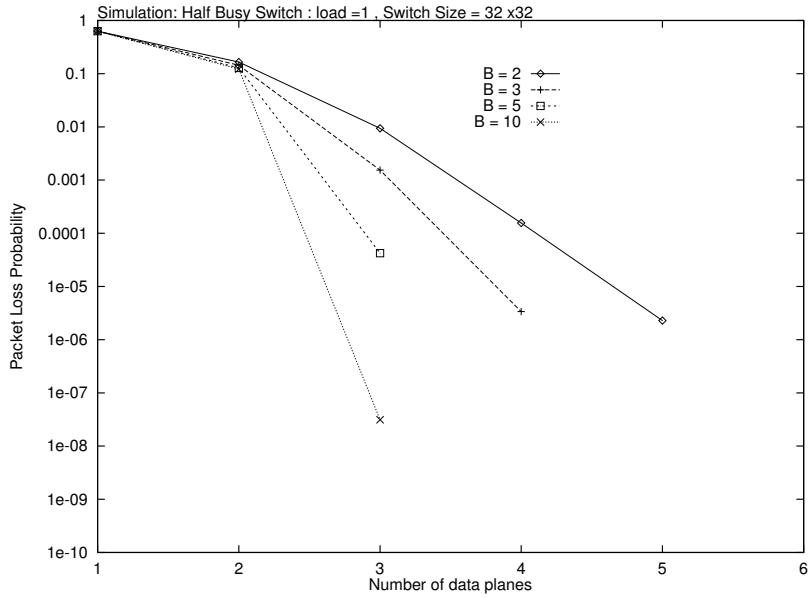

Figure 14: Loss in a 32×32 PCBSF pipeline versus the buffer size and the number of data planes

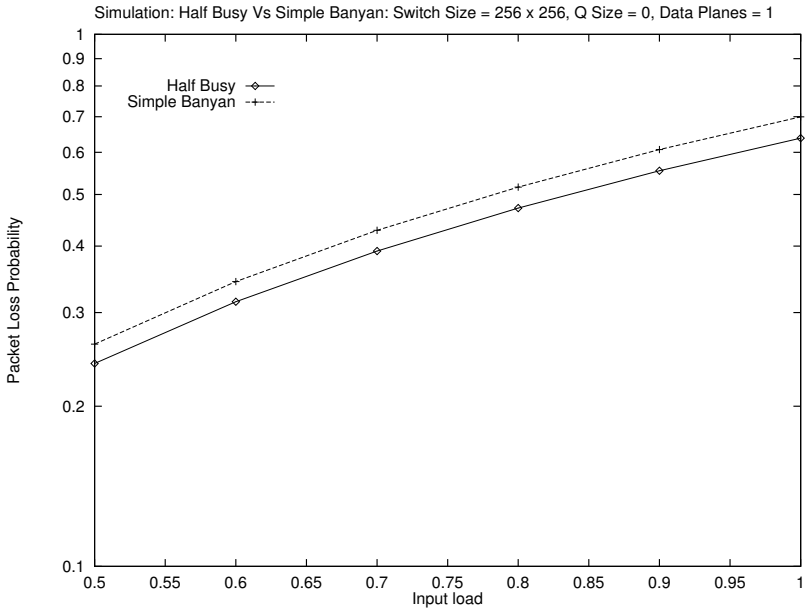

Figure 15: Loss in a 256×256 pipeline banyan and PCBSF versus input load

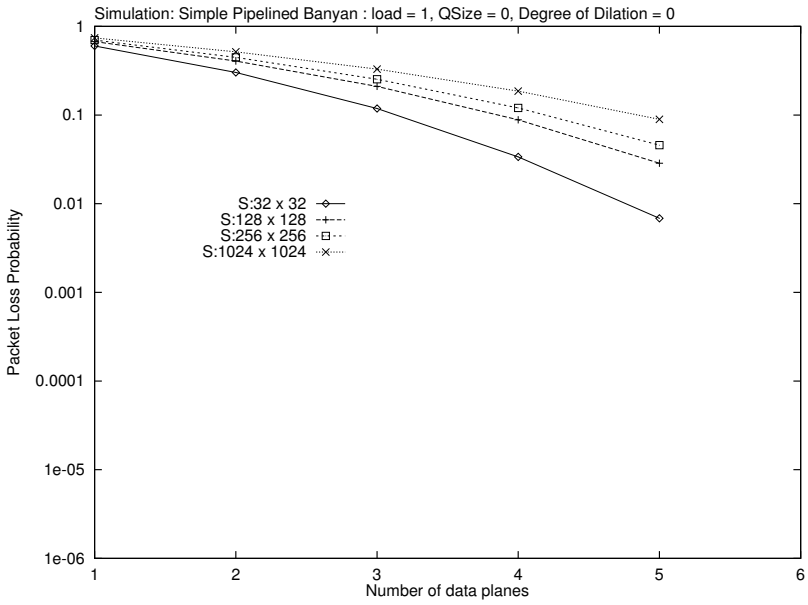

Figure 16: Loss in pipeline banyan versus the number of data planes

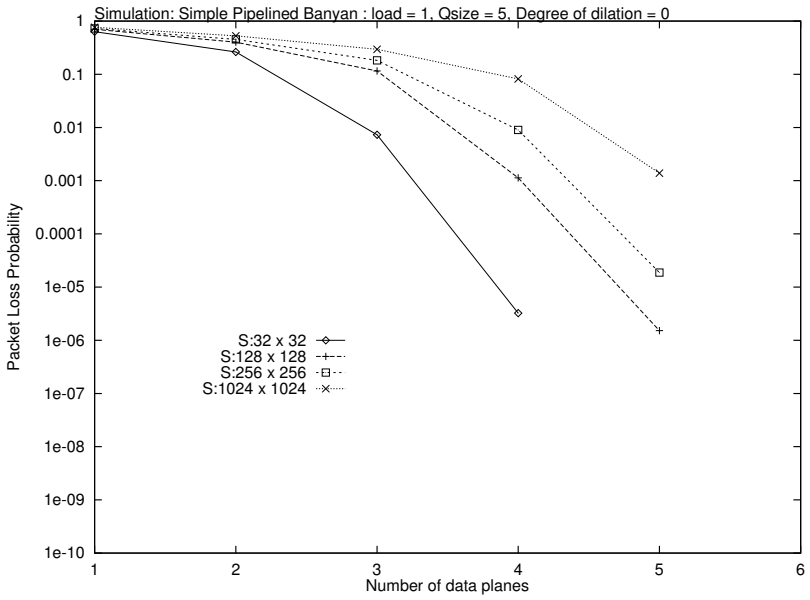

Figure 17: Loss in a 32×32 pipeline banyan versus the buffer size and the number of data planes

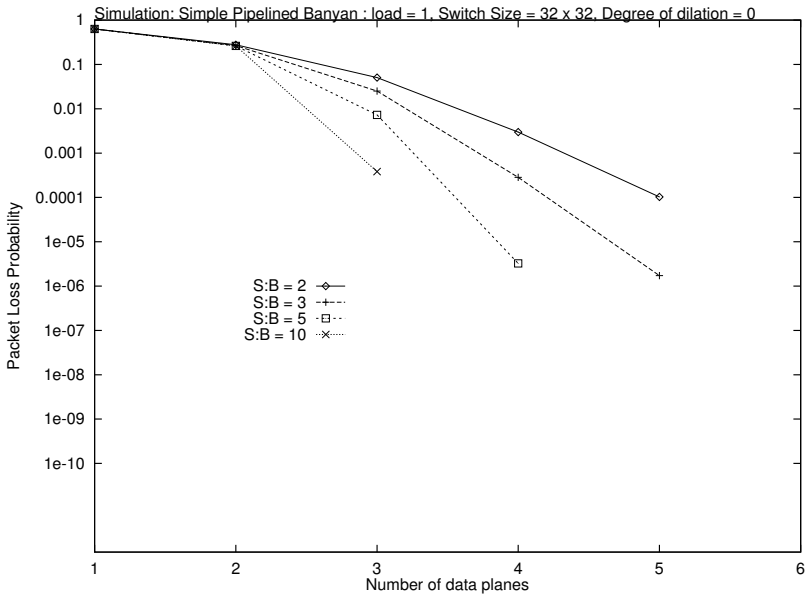

Figure 18: Loss in a 32×32 pipeline banyan versus the buffer size and the number of data planes

Type of Source	Mean burst size in cells	Average bit rate $SCR \times 384$ *bps*	Burstiness β	Cell Loss Tolerance
CBR (Voice)	N/A	64 Kbps	1	10^{-4} to 10^{-6}
Connectionless data	200	700 Kbps	as high as 1000	10^{-12}
Connection oriented data	200	25 Mbps	as high as 1000	10^{-12}
VBR video	2	25 Mbps	2 to 5	10^{-10}
Background data/video	3	1 Mbps	2 to 5	10^{-9} to 10^{-10}
VBR video/data	30	21 Mbps	2 to 5	10^{-9}

Table 1: Various traffic source types and their traffic descriptors.

the active and idle periods are exponentially distributed with parameters $a = \frac{1}{t_{ON}}$ and $b = \frac{1}{t_{OFF}}$ respectively.

We experimented with the two traffic mixes, *Traffic 1* and *Traffic 2*, described below,

Traffic	Source type	Peak arrival rate (Mbps)	Average cell arrival (Mbps)	Mean burst length (cells)	Burstiness	Percentage channels
Traffic 1	CBR	0.064				10%
	CBR	1.4				10%
	VBR		0.7	200	5	20%
	VBR		25	20	5	20%
	VBR		21	30	4	40%
Traffic 2	CBR	0.064				25%
	CBR	1.4				25%
	VBR		0.7	200	5	12%
	VBR		20	25	5	13%
	VBR		2	25	10	6%
	VBR		3	1	5	6%
	VBR		30	21	4	6%
	VBR		3	6	5	7%

To generate traffic sources according to the ON-OFF model VBR traffic sources require specification of four input parameters m, t_{ON}, and β in addition to percentage of input channels assigned to these sources. For CBR sources, the peak cell rate p and the percentage of input channels are sufficient.

Figures 19 to 21 show cell loss performance estimations of the path clustering pipelined banyan, under ATM traffic mix "Traffic 1". Similar results were obtained for the second traffic mix. The plots show that with a buffer size as low as $B = 5$, and three data planes, the cell loss ratio is less than 10^{-8} for switch sizes varying between 32x32 and 1024x1024. We were unable to simulate larger switch sizes because of excessive runtime requirement. The reader should notice that the switch cell loss performance under simulated ATM load is better than for the full uniform workload. The reason is that the ATM load was a lighter load than the uniform load. Furthermore, unlike the ATM traffic, cell destinations were completely random for the uniform load.

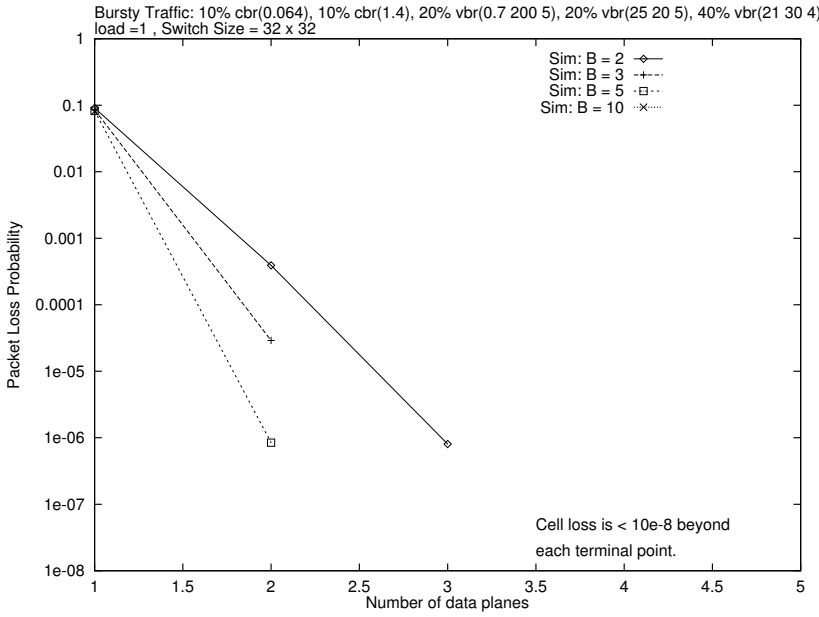

Figure 19: Effect of varying buffer size on a 32 x 32 switch

7 Conclusion

The operation of multi-banyan based ATM switches mainly consists of partitioning the HOL cells into subsets so that cells within each subset can be routed without conflicts in one single banyan. This strategy allows parallel path reservation within each banyan but multiple banyan planes must be reserved in a sequential fashion.

In this paper, we proposed a different method based on performing incremental path reservation in each banyan on the basis of previously established connections. This approach enables multiple banyan planes be concurrently reserved. For this we proposed efficient hardware to each banyan for the generation of conflict-free path offers to head of the line cells waiting in input buffers. An offer is allocated to the cell whose source to destination path uses the largest number of partially allocated switching elements. This approach maximizes the sharing of switching elements in each banyan, thus leaving the largest number of free switches for subsequent reservation which reduces potential of future conflicts. Based on the above technique, a pipelined switch architecture called *Path Clustering Banyan Switching Fabric* PCBSF was proposed. We showed that PCBSF can deliver very high throughput under simulated *uniform traffic* and *ATM traffic mixes*. By varying the size of the switch and that of input buffers, we showed that performance and robustness of PCBSF compare favorably to other pipelined banyan switches.

8 Appendix

Consider the class of dynamic, full access, unique path, multistage networks (MINs) that use 2×2 switches, 2^n inputs and 2^n outputs for each of the n stages. Interstage interconnection represents some permutations. Examples of well known permutations are the *perfect shuffle* (σ), *Butterfly* (β), and *Exchange* (e), which are defined by $\sigma(x_{n-1}, \ldots, x_0) =$

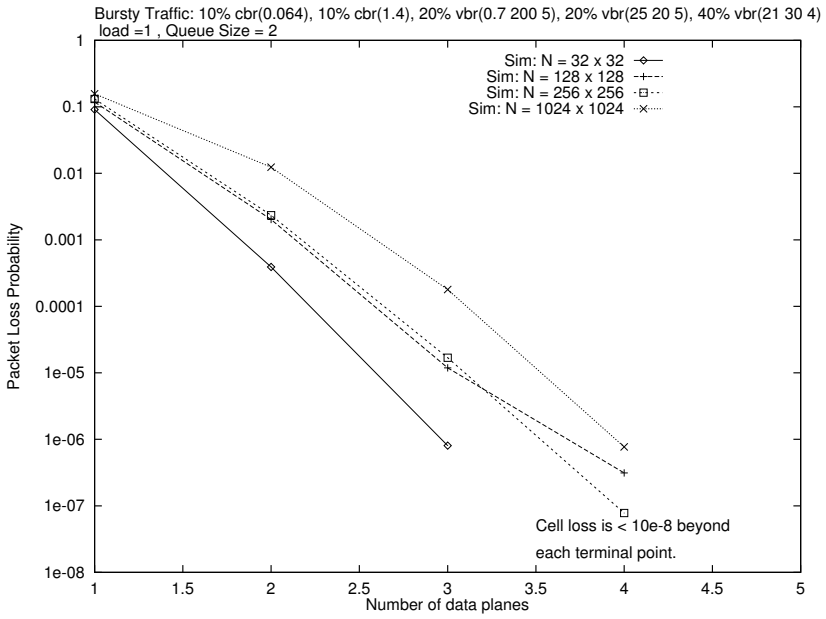

Figure 20: Performance of buffered path-clustering switches with queue size 2

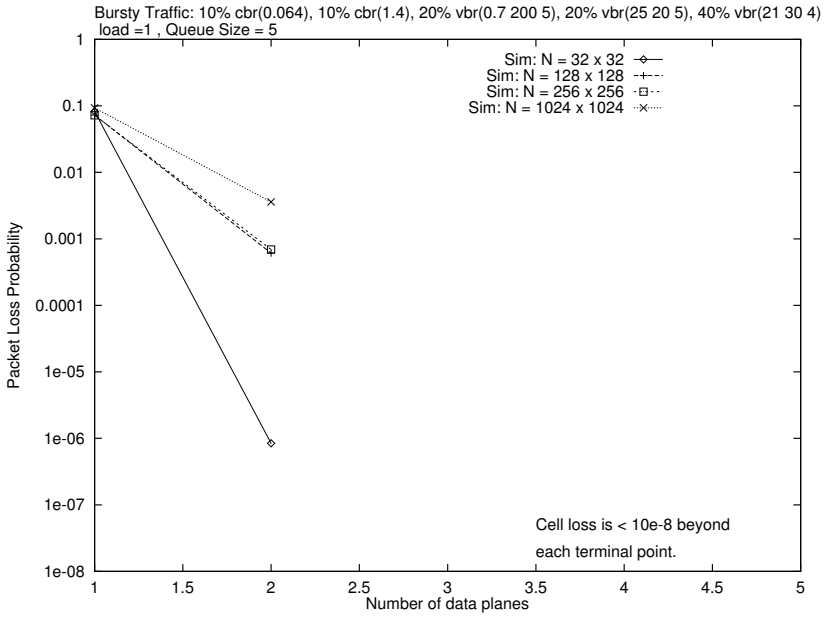

Figure 21: performance of buffered path-clustering switches with queue size 5

$x_{n-2}, \ldots, x_0, x_{n-1}$, $\beta(x_{n-1}, \ldots, x_0) = x_0, x_{n-2}, \ldots, x_1, x_{n-1}$ and $e(x_{n-1}, \ldots, x_0) = x_{n-1}, \ldots, x_1, \bar{x}_0$.

A MIN with 2^n inputs and 2^n outputs has 2^{n-1} switches in each stage for a total of $n2^{n-1}$ switches. Each switch has two states *straight* and *exchange* and, therefore, it can perform $(N^N)^{1/2}$ permutations each corresponds to one state of the $n2^{n-1}$ switches, where $N = 2^n$. A MIN is said to be *blocking* because it cannot perform all possible $N!$ permutations of the inputs. A network can perform a given permutation π if there exists a setting of the switches in the network so that the ith input is connected to the $\pi(i)$th output, where π is a permutation defined over the integers $\{0, \ldots, 2^n - 1\}$ and $0 \leq i \leq 2^n - 1$. A crossbar switch can achieve all possible permutations but its drawback is the cost of the N^2 switches it requires.

The interstage interconnection is defined by a permutation of the addresses. This allows recursive definition of networks by means of the above permutations and stages. Examples of well know networks are *Omega* (Ω_n), *Baseline* (B_n), *Cube* (C_n), and *Delta* (Δ_n), which can be defined as follows $\Omega_n = (\sigma E)^n$, $B_n = E\sigma_n^{-1} \ldots E\sigma_2^{-1}E$, $C_n = \sigma E\beta_n E \ldots E\beta_2 E$, and $\Delta_n = (E\sigma)^{n-1}E$, respectively, where E denotes a particular stage.

Since there are 2^n inputs (sources) and outputs (destinations) in an Ω_n network, routing a source $s = s_{n-1} \ldots s_0$ to destination $d = d_{n-1} \ldots d_0$ consists of finding a path of switches (w) that connect s to d. Each switch w receives two incident sources s and s', with destinations d and d', on its upper and lower inputs, respectively. A switch can achieve either straight or exchange permutation. The routing bit (d) directs the cell to either the upper output ($d = 0$) or lower output ($d = 1$). A collision occurs at the switch when both incident cells require exiting the switch at the same output.

In a unique path network a cell that is issued at source $s = (s_{n-1} \ldots s_i \ldots s_0)$ with its destination $d = (d_{n-1} \ldots d_i \ldots d_0)$ is to route along a unique path that is defined by pair (s, d). We first need to find the position of a cell at the output of each of each stage of the network which completely identifies the paths (s, d).

Lemma 1 *Given a n-stage Omega network, let $pos_i(s, d)$ be the position of a cell passing from input $s = (s_{n-1} \ldots s_i \ldots s_0)$ to destination $d = (d_{n-1} \ldots d_i \ldots d_0)$ after the ith stage. Then:*

$$pos_i(s, d) = s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i}$$

in binary representation, where s_i and d_i are the ith source and destination bits, and $0 \leq i \leq n - 1$.

Proof The proof is by induction. The base case is $pos_0(s, d) = s_{n-1} \ldots s_0$, which is the position of the cell at input s of the network. We assume that $pos_{i-1}(s, d) = s_{n-i}s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1}$ and we need to prove that $pos_i(s, d) = s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1}d_{n-i}$.

Since the cell enters some switch at stage i following a σ permutation, the position of the cell at the *ith* stage input is:

$$\sigma(pos_{i-1}(s, d)) = s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1}s_{n-i}$$

At every switch of ith stage, the position $pos_i(s, d)$ of the cell is determined by d_{n-i} that is $s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1}0$ if $d_{n-i} = 0$ and $s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1}1$ if $d_{n-i} = 1$.

Therefore, $pos_i(s,d)$ must be $s_{n-i-1}\dots s_0 d_{n-1}\dots d_{n-i+1}d_{n-i}$ ∎

This result is applicable to any single-path MIN which means that we can find $pos_i(s,d)$ for all other MINs. Assume a number of paths have already been reserved within the network and a new cell is to be routed from s to d. One needs to find whether the above cell passes the network without conflicts with respect to currently reserved paths. For this, one may identify all the source inputs that a cell could have been issued from in order to reach the desired destination d. In other words, the knowledge of d must be sufficient to find all possible conflict-free paths (s,d) by finding all the possible values of s. We first find all possible cell positions at the ith stage to exit the network at destination d. At this level we only find all potential paths to exit at d and later we will provide a mechanism to cancel any possible path that conflicts with currently reserved paths.

Lemma 2 *A cell that exits Ω_n at position $d_{n-1}\dots d_0$ should pass the ith stage at any switch output whose position is $x_{n-i-1}\dots x_0 d_{n-1}\dots d_{n-i}$, where $x_{n-i-1}\dots x_0$ may take any possible binary combination out of 2^{n-i}.*

Proof The proof is by induction. The base case is a cell that exits stage $n-1$ (last stage) at $out_{n-1} = d_{n-1}\dots d_0$ of switch $sw = d_{n-1}\dots d_1$. This cell should have entered the switch at inputs $in_{n-1} = d_{n-1}\dots d_1 0$ or $in_{n-1} = d_{n-1}\dots d_1 1$ which correspond to the output position $out_{n-2} = 0 d_{n-1}\dots d_1$ or $out'_{n-2} = 1 d_{n-1}\dots d_1$ of stage $n-2$. Here, x_0 may take the values 0 or 1 as shown in out_{n-2} and out'_{n-2}. This results from backward propagation through Ω_n which is obtained by applying an inverse perfect shuffle permutation (σ^{-1}).

We assume the cell passed through any of the outputs of stage $i+1$ whose positions are given by:

$$x_{n-i-2}x_{n-i-3}\dots x_0 d_{n-1}\dots d_{n-i}d_{n-i-1}$$

where $x_{n-i-2}x_{n-i-3}\dots x_0$ may take any combination out of 2^{n-i-1}, and we need to prove that the cell should have passed the ith stage at outputs $out_i = 0 x_{n-i-1}\dots x_0 d_{n-1}\dots d_{n-i+1}$ d_{n-i} or $out'_i = 1 x_{n-i-1}\dots x_0 d_{n-1}\dots d_{n-i+1}d_{n-i}$.

By definition of the stage operation, the position of the cell at the input of stage $i+1$ is necessarily $in_{i+1} = x_{n-i-2}x_{n-i-3}\dots x_0 d_{n-1}\dots d_{n-i}0$ or $in'_{i+1} = x_{n-i-2}x_{n-i-3}\dots x_0$ $d_{n-1}\dots d_{n-i}1$. By definition of Ω_n, we apply a σ^{-1} over in_{i+1} and in'_{i+1} which gives positions of the cell at the output of stage i that are identical to out_i or out'_i, respectively. Therefore, for any combination of $x_{n-i-2}x_{n-i-3}\dots x_0$, for the output of stage $i+1$, there exists two possible combinations for the ith stage that are out_i and out'_i.

Any path from output $x_{n-i}x_{n-i-1}\dots x_0 d_{n-1}\dots d_{n-i+1}d_{n-i}$ of the i stage can then be used to route to switch output $d_{n-1}\dots d_0$ of last stage, where $x_{n-i}x_{n-i-1}\dots x_0$ is any possible binary combination. The total number of paths from the ith stage to some destination $d_{n-1}\dots d_0$ is then $2^{n-i-1}2 = 2^{n-i}$. ∎

Assume a cell reaches a switch which is already set in one of its two possible states (straight or swap). We need to find out under what condition the cell can pass the switch without conflicts. In other words, the cell may arrive to the switch at its lower or upper input and may request to exit the switch at its upper or lower output. The following

Lemma gives the relationships between the cell entry and exit positions and the previous state of the switch in order not to cause any conflict. The state of a 2×2 switch can either be 0 or 1, which denote that the switch is in state "straight" or "swap", respectively.

Lemma 3 *In an Ω_n network, a cell that reaches the ith stage at some input of switch $s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1}$ passes the switch without conflict if and only if the switch state is $s_{n-i} \oplus d_{n-1}$.*

Proof Using Lemma 1 we know that the position of a cell at the output of the ith stage is $s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1} d_{n-i}$. The cell exits the switch at its upper output if $d_{n-i} = 0$ and at its lower output if $d_{n-i} = 1$. On the other hand, the position of the same cell at the input of the ith stage must be $s_{n-i-1} \ldots s_0 d_{n-1} \ldots d_{n-i+1} s_{n-i}$ The cell entered the switch at its upper input if $s_{n-i} = 0$ and at its lower input if $s_{n-i} = 1$. Therefore, the cell passes the switch without conflicts if the switch is in a straight state and $s_{n-i} = d_{n-1}$ or if the switch is in the swap state and $s_{n-i} \neq d_{n-1}$. Therefore, the cell passes the switch without conflict if and only if its state is identical to $s_{n-i} \oplus d_{n-1}$. ∎

In summary, Lemma 2 shows how one can find all inter-stage inputs from where a cell can reach a given destination. In Lemma 3 we found the condition that source s and destination d of a cell must satisfy to traverse a 2×2 switch without conflicts. Notice that the previous state of the same switch may have been previously assigned. Thus Lemmas 2 and 3 provide the basis for finding whether a source-destination path conflicts with another one or not. In the next section we will use the above results for designing a switch that selectively assigns paths to cells based on the concept of maximizing the sharing (non-conflicting) of switches with respect to some previously reserved paths.

References

[1] M. Kawarasaki and B. Jabbari. B-ISDN architecture and protocol. *IEEE J. Selected Areas in Communications*, 9(9):1405–1415, Dec. 1991.

[2] D. Delisle and L. Pelamourgues. B-ISDN and how it works. *IEEE Spectrum*, 28(8):39–42, Aug. 1991.

[3] Martin de Prycker. *Asynchronous Transfer Mode - solution for broadband ISDN.* Ellis Horwood, 1991.

[4] Ronald J. Vetter. Atm concepts, architectures and protocols. *Communications of the ACM*, 38(2):31–38, Feb 1995.

[5] Reza Rooholamini, Vladimir Cherkassky, and Mark Garver. Finding the right ATM switch for the market. *IEEE Computer*, 27(4):17–28, Apr. 1994.

[6] A. Huang and S. Knauer. Starlite: A wideband digital switch. *Proc. GLOBECOM 84, Atlanta, GA*, pages 121–125, 1984.

[7] J. Giacopelli et al. Sunshine: A high performance self-routing broadband packet-switch architecture. *IEEE J. Selected Areas in Communications*, pages 1289–1298, Oct 1991.

[8] M. J. Narasimha. The Batcher-banyan self routing network: universality and simplification. *IEEE Trans. Commun.*, 36(10):1175–1178, Oct 1988.

[9] Fouad A. Tobagi, Timothy Kwok, and Fabio M. Chiussi. Architecture, performance, and implementation of the tandem Banyan fast packet switch. *IEEE J. Selected Areas in Communications*, 9(8):1173–1193, Oct. 1991.

[10] Toshihiro Hanawa et al. Multistage interconnection networks with multiple outlets. *1994 International Conference on Parallel Processing*, I:1–8, 1994.

[11] M. Al-Mouhamed, H. Yousef, and W. Hassan. A Parallel-Tree Switch Architecture for ATM Networks. *Journal of Communication Systems*, Vol. 11, issue 1, 1998.

[12] P.C.Wong and M.S. Yeung. Design and analysis of a Novel Fast Packet Switch - Pipeline Banyan. *IEEE/ACM Transcations on Networking*, 3(1):63–69, february 1995.

[13] H. Ahmadi and W. Denzel. A survey of modern high-performance switching techniques. *IEEE J. Selected Areas in Communications*, 7(7):1091–1103, Sept 1989.

[14] A. Saha and M. D. Wagh. Performance analysis of banyan networks based on buffers of various sizes. *IEEE Proc. INFOCOM'90*, 1:157–164, 1990.

[15] K . Shiomoto et al. Performance evaluation of cell bypass queueing discipline for buffered banyan type ATM switches. *Proc. IEEE INFOCOM'90*, 2:677–685, 1990.

[16] K. E. Batcher. Sorting networks and their applications. *AFIPS Proc. 1968 Spring Joint Computer Conf.*, 32:307–314, 1968.

[17] G. D. Stamoulis, M. E. Anagnostou, and A. D. Georgantas. Traffic source models for ATM networks: a survey. *Computer Communications, Butterworth-Heinemann Ltd*, 17(6):428–438, June 1994.

[18] Ken Dubose and Hyong S. Kim. An Effective Bit Rate/Table Lookup Based Admission Control Algorithm for the ATM B-ISDN. *Proceedings of the* 17^{th} *Conference on Local Computer Networks, Minneapolis*, pages 20–29, Sept. 1992.