Designing SANs to Support Low-Fanout Multicasts

Rajendra V. Boppana[*,1], Rajesh Boppana[1], and Suresh Chalasani[2]

[1] CS Department,The Univ. of Texas at San Antonio, San Antonio, TX 78249
{boppana, brajesh}@cs.utsa.edu
[2] School of Business and Technology, Univ. of Wisconsin-Parkside, Kenosha, WI 53141
suresh.chalasani@uwp.edu

Abstract. System area networks (SANs) need to support low-fanout multicasts efficiently in addition to broadcasts and unicasts. A critical component in SANs is the switch, which is commonly designed around crossbars. We present new switch designs using a combination of low-cost multistage switching fabrics and input and output buffering with hardware based packet scheduling mechanism. Using detailed simulations, we show that the proposed designs can scale to 512-ports and outperform crossbar based designs in a 4-switch SAN.

1 Introduction

System area networks (SANs), designed using high-speed switches with direct links, are used to interconnect and provide high-throughput, low-latency communication among workstations. Similar switched networks are also used for high-speed local area networks (LANs) and storage area networks (SANs)to accommodate a variety of needs ranging from parallel computing, storage area networking, to video conferencing. These applications often use low-fanout multicast messages in addition to unicasts and broadcasts. It is important that future switches be designed to handle these diverse communication needs.

A multicast on a switched network is facilitated by replicating a multicast packet at appropriate switches as the packet progresses from its source to destinations. A multicast packet arriving at the input port of a switch may require more than one output port of that switch. Since multicast packets from different inputs may have one or more common outputs, the conflicts for the output ports of a switch increase. Often, a packet could be sent to some but not all output ports it needs to reach, owing to the output conflicts. A number of different architectures have been proposed to handle this problem.

In this paper, we focus on designing switches that can handle multicast traffic efficiently without compromising the unicast traffic. The key component of a switch is the switching fabric used to provide the datapath for packets to move within a switch from input ports to output ports. The most commonly used choices are shared-bus and crossbar. The shared-bus designs are not scalable owing to performance constraints, and the crossbar-based designs are not scalable owing to cost considerations. Our design uses a multistage network, specifically an Omega multistage interconnection network, as the switching fabric with input and output buffers and a hardware based packet scheduling. This design handles both unicast and multicast traffic. We also look into

[*] Boppana's research has been partially supported by NSF grant EIA-0117255.

Presented at International Conference on High Performance Computing, Dec. 2003.

the performance of SANs based on proposed switches. For a simple network of four switches, we show that the proposed switches give as much as 50% more throughput than crossbar-based switches.

2 Background and Related Work

We assume that fixed-size packets (called cells) are used for communication. If a message is too large to fit in a single packet, then multiple cells are sent from source to intended set of destinations. Such a message appears as bursty traffic or correlated traffic to the network. The amount of time a cell takes to move from input ports of a switch to its output ports is called a time-slot. For balanced design, this should be about the same as the transmission time for a cell. For example, the slot-time for a 100-byte cell with a line rate of 10 GHz is 80 ns. We are primarily interested in cut-through or store-and-forward switching with best-effort delivery. So excess packets may be dropped by switches when traffic load is high. For applications that require reliable delivery of data, a transport layer such as TCP will need to be used. It is noteworthy that the majority of parallel computing applications are currently designed to work on top of a standard TCP/IP protocol stack.

We assume switches are of size $N \times N$. Many switched networks and standards for the same were proposed in the literature [8–12]. These networks use crossbar based switches designed primarily to handle unicast traffic. Several designs to handle multicasts were proposed in the computer networks community [1–6]. Of particular interest is the Weight-based algorithm (WBA) for crossbar-based switches with input buffering, which is shown to achieve high throughput for multicasts [1]. In this design, each multicast cell is assigned a weight using the following formula $a \times \text{age} - f \times \text{fanout}$, where age is the number of time-slots for which the cell is queued in the current switch input buffers and fanout is the number of output ports of the switch it needs; a and f are tunable parameters and are often chosen as 1 and 2, respectively. However, the cost of crossbars in terms of crosspoints increases at $O(N^2)$. For work conservation, a technique known as cell-splitting is often used. In cell-splitting, when a multicast cell competes for outputs with other cells and is granted to reach only a partial set of its outputs, a copy of the cell is retained in the input buffer with the remaining outputs as its destination list.

3 *Ω* Switch

Our design attempts to simplify the design complexity of switching fabric and output ports at the cost of more complex cell scheduling. We use a combination of input and output buffers and multiple copies of the Omega multistage interconnection network (MIN) [7] as the data network (switching fabric). We denote the switch as the *Ω* switch. Figure 1 shows the block diagram of *Ω* switch.

The cells arriving from the input lines are buffered at the input port buffers. Once a cell is in the input buffer, it is guaranteed to reach all of its destinations (switch outputs). Hence there is no cell loss within the switch fabric or output

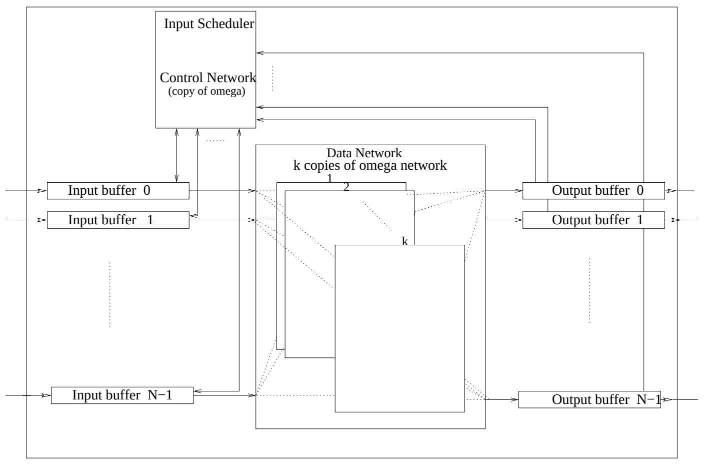

Fig. 1. Switch architecture. The switch constitutes of input and output queues, control network and data network. The control network is based on the Omega MIN and acts as input scheduler. The data network consists of k copies Omega MIN for routing the cells through the switch. Normally $k \leq 3$.

buffers. Cells are lost only when they arrive at the switch and are rejected by the input ports. This happens when the switch is saturated. Based upon the header of the cell, the destinations of the cell are determined by table lookup and a routing tag is assigned for routing through the switch fabric. For the purpose of the discussion, we assume the routing tag to be an N bit vector with each bit denoting a particular output port. A cell is destined to an output port if its corresponding bit in the vector is set. Routing tags could be precomputed or prepared as a cell is placed in an input buffer.

We select cells in such a way that there are no path conflicts within the switch fabric. Because cells are carefully selected to eliminate path conflicts, there is no need for buffering at any of the switching elements inside the switch fabric. We believe that, by moving all the necessary buffering away from the data path of the switch, we can simplify the switch fabric design.

The input scheduler determines the inputs and the cells they need to send through the switch fabric. Contention for paths within the switch fabric and outputs affect the performance of the switch. When crossbars are used as switch fabrics, there is no contention for data paths through the switch fabrics. It is a severe problem, however, when blocking switch fabrics like Omega MIN are used. On the other hand using crossbars could be expensive. To alleviate this, we use multiple copies say, k, of the Omega MIN in the switch fabric. So conflicts for a path in the switch fabric can be handled by sending contending cells through different copies of the network. This increases the bandwidth inside the switch and achieves high throughput.

Since we use k copies of the Omega MIN as the switch fabric, each output could receive up to k cells during a single slot time. Hence the output buffers need to operate at k times the line speed. We can restrict the maximum number of cells sent to an output port in a single slot time to some l, where $l \leq k$. When k is small, it is simpler to have $l = k$. In each slot time, one cell is sent out of the output port. Excess cells are buffered in the output queues. This would achieve high switch utilization since any output that did not receive a cell due to blocking nature of the switch fabric could send out a cell as long as its buffer is not empty. This is particularly useful for correlated traffic in which the demand for an output tends to be bursty. Although, we use multiple copies of Omega MIN we limit the complexity or speedup requirements of output buffers by controlling the output port contention through cell scheduling so that the switch is scalable.

3.1 Scheduling Cells

It is known that multistage interconnection network based switches need elaborate cell selection techniques to achieve high throughput. Software based cell selection schemes do not work well for high line rates or large switches. Therefore, we use the idea of hardware specific selection technique for unicast cells by Boppana and Raghavendra [15]. In this method, a copy of the underlying network for the switch fabric (that is, the Omega multistage network itself) as the control network to aid the selection of cells. However this control copy network needs to route the routing tags (N-bit tags for an $N \times N$ switch) of the cells for scheduling purposes. The conflicts for data paths through a switch fabric will appear as contention for a switching element's output links in the Omega MIN. These conflicts are resolved randomly or by using a suitable priority mechanism. Cells whose routing tags go through all the stages of the control network successfully will be actually routed through the switch fabric in the following time slot. The self-routing techniques can be used to set up the switching elements in the switch fabric since the cells going through the data network will not have conflicts. To improve the number of cells that can be sent in a time slot, this selection process may be repeated several times for each copy of the data network.

Round-Robin Scheduling The simplest way to schedule multicast cells is to allow one or more input ports to send their multicast cells in a round-robin (RR) fashion during each slot time. Since our proposed switch has k data networks, its intuitive to select k input ports to route their multicast cells through the k data networks (one through each copy).

Hence, the simplest scheduling policy is a prioritized round robin algorithm in which, during each slot, k input ports have the priority and can send their multicast cell unrestricted through a distinct copy of the data network. If a prioritized input port doesn't have a multicast cell the next input port (in a circular fashion) gets a chance. For the next slot, next k inputs will be the priority inputs. This policy basically routes k multicast cells through the switch during each slot time. So switch utilization $\rho = \frac{k \times f}{N}$, where k is number of copies of data network, f is the fanout of the multicast traffic and N is the size of the switch. This scheme does not work well if f is small and N is large.

Scheduling Additional Cells To improve the performance, we attempt to schedule more than k multicast cells during each slot time; some Omega MINs send two or more multicast cells without path conflicts. This can be achieved by performing a round of selection for additional multicast cells that could go through each copy of the data network. So, after determining the k input ports that would send their multicast cells through the k data networks, using the round-robin scheduling, we let the remaining input ports send their output requests (routing tags) through the control network and determine if any other multicast cells could be scheduled through each of the data networks without conflicting with the internal route of the earlier selected multicast cell. Hence there is one round of selection for each of the k copies. It is noteworthy that this additional scheduling improves the switch utilization by using additional data paths. Each input request is either satisfied or rejected in entirety. That is, there is no cell splitting. If the fanout of the multicast cells is high, the probability of finding a cell which uses paths that do not conflict with those of an already selected multicast cell (using round-robin scheme) decreases and this scheduling does not achieve any improvement over the simple round-robin policy. So for multicast traffic with larger fanout, it becomes necessary to split the cell to achieve high switch utilization.

Scheduling Additional Cells with Fanout Splitting In this policy, after determining the k input ports that would send their multicast cells through the k data networks, the remaining input ports send their routing tags through the control network to determine if any other multicast cells could be scheduled through each of the data networks without conflicting with the internal routes of earlier selected multicast cells. If a multicast cell is able to obtain paths to reach some but not all of its destinations, we split the destination list and let a copy of the multicast cell go to the available outputs while retaining a copy with the remaining destinations at the input. Once an input port obtains paths for one or more destinations for its multicast cell, it will not compete for paths in the other copies of the data network. A split cell with reduced destination list is treated as a normal multicast cell for scheduling in later rounds.

We use output buffers to store cells to accommodate multiple cells arriving (via multiple data networks) at an output in a slot time. So the output queues could overflow and result in high cell loss. To prevent this, a back pressure mechanism is used. If the output queue size exceeds a threshold it accepts only one cell during each slot time until the number of cells at the output is less than the threshold.

4 Performance Analysis of Switch Designs

We simulated a $N \times N$, $8 \leq N \leq 1024$, Ω- based and crossbar-based switches with separate buffers for unicast and multicast traffic. Owing to space considerations, we present only a subset of the results; for additional results see [16].

While each input queue of crossbar can buffer up to 256 cells, Ω can buffer 128 cells at the input and 128 cells at the output. So the total buffer space used

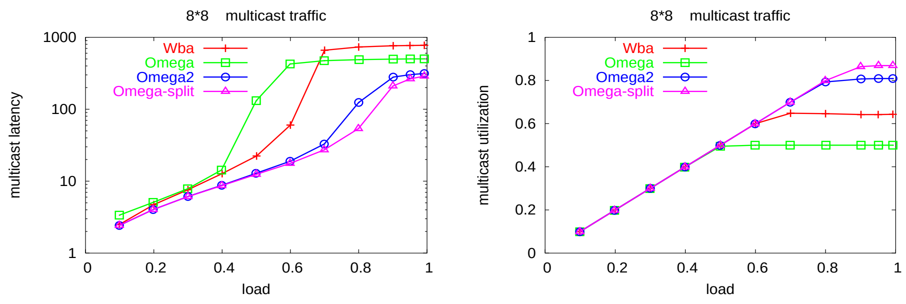

Fig. 2. Performance of 8×8 switches for correlated multicast traffic with fixed fanout of 2.

was the same for both designs. The control network of the Ω switch used two rounds (one for each copy) for additional multicast cell selection. Weights of 1 and 2 were used the age and fanout for WBA simulations.

Each simulation was run for multiple batches of 100,000 cycles, after a warmup of 50,000 cycles. Simulation was stopped when the half width of 95% confidence interval fell within 5% the computed mean latencies and throughputs.

Traffic patterns In this paper, we present results for correlated traffic, which is considered to be more representative of the traffic on computer networks [13, 14]. In correlated arrivals multicast cells generated in 16 consecutive slot times. Each injected cell has a constant fanout, specified as input to the simulation. The first cell of a stream of correlated cells chooses the destinations randomly; the remaining cells in the stream use the same destinations.

Performance metrics We use average cell latency and output port utilization as the performance metrics. Cell latency is the time elapsed from the time a cell is injected to the time it is delivered to all of its destinations. Output port utilization is the average number of cells delivered by the switch (or SAN) per output per slot-time. The maximum possible utilization is 1. Load offered to the switch (or SAN) is expressed as a fraction of the maximum utilization.

We use the following notations for all the plots. Omega, Omega2 and Omega-split denote, respectively, the simple round-robin, round-robin with one round of scheduling and Omega2 with fanout splitting. WBA denotes the weight-based scheduling for crossbar-based switches.

4.1 Small Switches

Figure 2 presents the results for correlated traffic. WBA saturates at about 64% while Omega2 can attain up to 80% and Omega-split saturates at 86%. The delay curves for Omega2 and Omega-split are significantly lower than that of WBA when the loads are in the range 50% to 90%. We observed that with uncorrelated arrivals (not shown here), utilizations improve for Omega2, Omega-split and WBA [16]. The utilization of round-robin Omega switch is oblivious

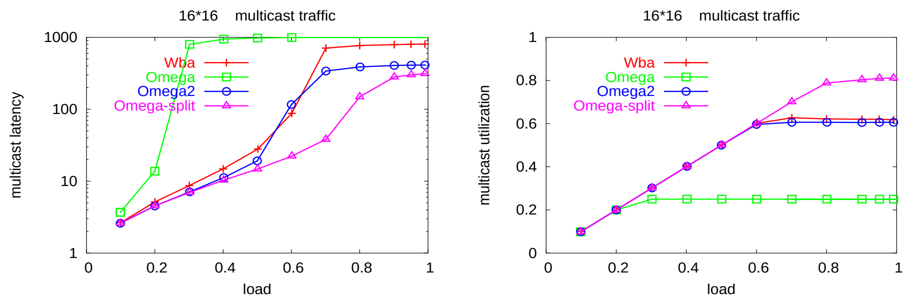

Fig. 3. Performance of 16×16 switches for correlated multicast traffic with fixed fanout of 2.

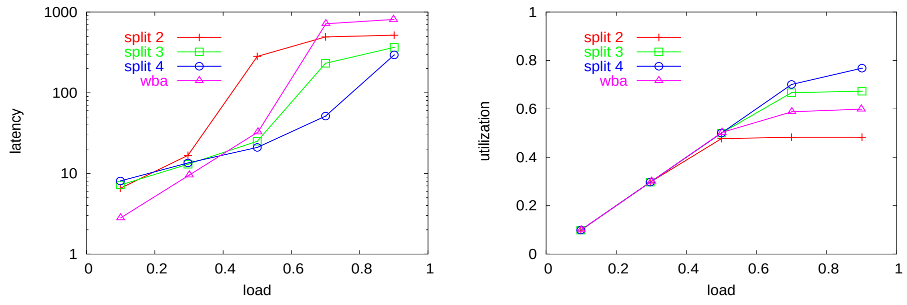

Fig. 4. Performance of 512×512 switches for correlated multicast traffic with fixed fanout of 2. Split-n indicates Omega-split with n copies of switch fabric.

to traffic pattern. Similar performances were observed for 16×16 switches (see Figure 3).

4.2 Large Switches

For small switch sizes, the Ω switch with fanout splitting performs well compared to the crossbar-based WBA switch. To see if this holds for larger switch sizes, we simulated 512×512 Omega-split and WBA switches. For Ω-switch simulations, we varied k, the number of data networks, to determine the benefit of more copies of the data network. Since the cell selection in the Ω switch is more complicated than that in a WBA switch, we assume that cell selection is pipelined using additional hardware and that it takes k time slots to determine the cells that can go through a single slot. So in Ω switch designs, cell selection starts k time-slots prior to the time slot for which the scheduling is done. For WBA the cell delay is still 2 slots in the absence of contention or waiting.

Figure 4 gives the simulation results for 512×512 switches. We observe that, while 2 copies are not sufficient to obtain good performance, one or two additional copies of switch fabric provide much higher utilizations. The utilization is improved by 20 percentage points when 3 copies (split-3) are used instead of

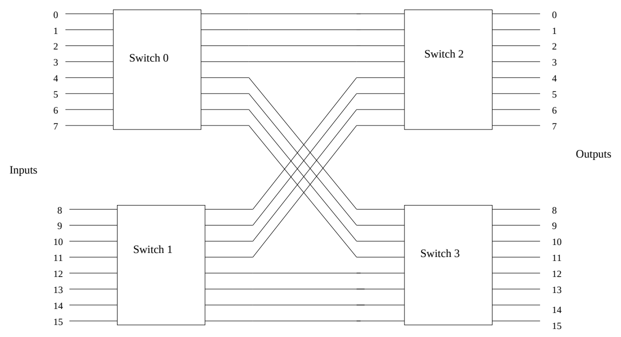

Fig. 5. Topology of SAN 1.

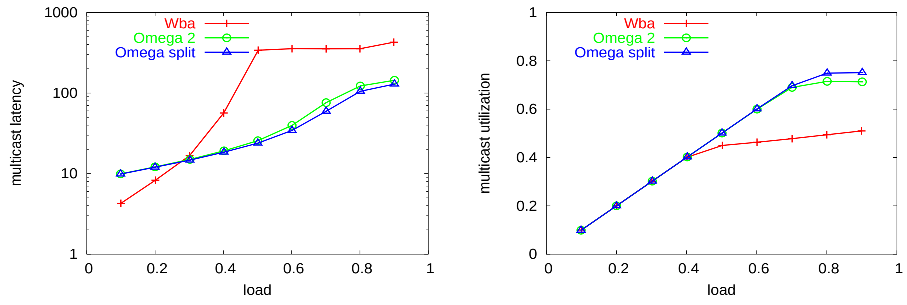

Fig. 6. Performance of SAN 1 for correlated multicast traffic with fixed fanout of 2

2 (split-2). The difference in the performances of 3 and 4 copies decreases with increasing fanout. For fanout 8 and above, they are almost identical [16].

Though Ω switches with fanout splitting outperform WBA for large switches, they are less expensive in terms of number of crosspoints: Split-3 requires fewer than 11% ($\frac{3\times\frac{512}{2}\times 9\times 4}{512\times 512}$) of the crosspoints used for a crossbar.

5 Performance Analysis of Switched Networks

Most published studies evaluate proposed switch designs as single components, but do not evaluate their switch designs in the context of a SAN. We used Omega and WBA switches as building blocks to form SANs. We have evaluated the performance of the SANs so designed. In particular, we analyzed the performance of the switches for two network topologies. The performance of one of the networks, shown in Figure 5, which provides the functionality of a 16×16 switch, is presented here. The performance of the second network is given in [16].

Figure 6 presents the network utilizations and latencies achieved by the switches for network in Figure 5 for multicast traffic of fanout 2. Though it uses more complex cell splitting, Omega-split performs only marginally better than Omega2. On the other hand, both Omega designs give 50% higher throughput than WBA. It is instructive to compare the performance of a single 16×16 switch given in Figure 3 with that of the network simulated. We see that Omega2 achieves a utilization of over 70% in the SAN configuration, but only 60% utilization as a single switch. On the other hand, WBA switch performs worse in a SAN configuration (at 40% utilization) than as a single switch (at 60% utilization). The reasons for the performance divergence is as follows.

The network utilization depends on utilization of each of the switches in the network. By arranging the switches in stages we have reduced the effective fanout of the cells passing through each stage of the switches. Conversely, the rate of cell arrivals at the second stage of switches is increased. For a multicast load of 0.4 with fanout 2 on the network in Figure 5, the cell rate (rate at which cells are injected in to the inputs) at switches 0 and 1 is $\frac{0.4}{2} = 0.2$. Since the effective fanout at these switches is 1.53, assuming all outputs are uniformly used, the cell rate at switches 2 and 3 is $0.2 \times 1.53 = 0.31$. This is about the cell rate at which a single 8×8 WBA switch starts to lose cells for fanout 2 (see Figure 2). This shows the inherent limitation on the cell rate that the WBA can sustain on the crossbars irrespective of the fanout for correlated traffic. On the other hand, Omega2, which can handle a higher rate of cells in 8×8 size, clearly benefits from the reduced fanout seen by each stage of switches and provides higher utilization than it does as a single 16×16 switch.

This clearly demonstrates the inherent weakness of input buffered crossbar designs for correlated traffic. This also illustrates the advantage of using limited output buffering, especially for correlated multicast traffic. For this reason, the Ω switch designs take advantage of reduced effective fanout in the traffic that occurs as cells go through multiple switches.

6 Conclusions

The increasing demand for various multicast applications requires system area networks based on high-speed multicast switches. These switches should handle both unicast and multicast traffic efficiently. In this paper, we have presented the design of a multicast switch based on multistage interconnection network with input and output buffers. Being based on multistage network, the design is cost effective and scalable.

We have illustrated three possible designs based on the cell selection policy. They are Omega, which has a simple round-robin scheme, Omega2, which has an additional round of scheduling in addition to the round-robin scheduling, and Omega-split, which is basically Omega2 with cell splitting. We have developed a modular simulator in java to evaluate the performance of the switches for the proposed scheduling policies. We have observed that the Omega-split outperforms WBA on crossbar for switch sizes ranging from 8×8 to 512×512.

We have analyzed the performance of a simple 4-switch SAN built from the Ω and WBA switches. In particular, we simulated a 16×16 switch by interconnecting four 8×8 switches and observed that the throughput achieved is more than that of a single 16×16 switch for Omega based designs, while WBA performs worse in the SAN configuration. An interesting outcome of our study is that Omega2, which does not use cell-splitting, performs as well as Omega-split and better than WBA for small switches. On the other hand published results indicate that crossbar-based switches do not work well without fanout splitting [1]. Another contribution of our study is that WBA which uses crossbars as switch fabrics suffers as its not able to sustain the cell rate for low fanout multicast traffic that occurs as cells go through multiple switches in a network.

References

1. Balaji Prabhakar, Nick McKeown, Ritesh Ahuja, “Multicast Scheduling for Input-Queued Switches,” IEEE Journal on Selected Areas in Communications, vol 15, No. 15, pp. 885-866, June 1997.
2. Y. Yeh, M. Hluchyj, and A. Acampora, “The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching,” IEEE Journal on Selected Areas in Communications, Vol SAC-5, No. 8, October 1987, pp. 1274-1283.
3. Rein J. F. de Vries, “ATM Multicast connections using the Gauss switch,” in Proc. GLOBECOM 1990, pp. 211-217.
4. D. X. Chen and J. W. Mark, “Multicasting in SCOQ Switch,” INFOCOM ’94, pp. 290-297, 1994.
5. H. J. Chao and B. S. Choe, “Design and Analysis of large-scale multicast output buffered ATM switch,” IEEE/ACM Trans. Networking, vol. 3, pp. 126-138, April 1995.
6. K. L. E. Law and A. Leon-Garcia, “A large scalable ATM multicast switch,” IEEE J. Selected Areas Commun., vol15, no. 5, pp. 844-854, 1997.
7. Duncan H. Lawrie, “Access and Alignment of Data in an Array Processor” IEEE Transactions on Computers, Dec. 1975, pp 1145-1155.
8. Michael D. Schroeder et. al., “Autonet: A High-Speed, Self-Con figuring Local Area Network Using Point-to-Point links,” IEEE Journal of selected areas in communications. vol. 9. no.8, Oct. 1991.
9. N. J. Boden et. al., “Myrinet: A Gigabit-per-second local area network,” IEEE Micro, pages 29-36, Feb. 1995.
10. D.Garcia and W. Watson, “Servernet II,” Proceedings of the 1997 Parallel Computer, Routing and Communication Workshop, June 1997.
11. R. Sheifert, “Gigabit Ethernet,” Addison-Wesley, April 1998.
12. Infiniband Trade Assoc., “Infiniband Architecture Specification, Release 1.0, ” Infiniband Trade Association, 2000.
13. W. E. leland, M. S. Taqqu, W. Willinger and D. V. Willson, “On the self-similar nature of Ethernet traffic (extended version),” IEEE/ACM Transactions on Networking, vol. 2, pp 1-15, Feb. 1994.
14. V. Paxson and S. Floyd, “Wide area traffic: The failure of Poisson modeling,” IEEE/ACM Transactions on Networking, vol. 3, pp. 226-244, June 1995.
15. Rajendra V. Boppana, C. S. Raghavendra, “Designing efficient Benes and Banyan based input buffered ATM switches,” ICC 1999.
16. Rajesh Boppana, ”Design of Multicast Switches for SANs.” Masters thesis, University of Texas at San Antonio, May 2003.