A Practical Version of Lee's Multicast Switch Architecture

Jonathan S. Turner

WUCS-91-46

August 26, 1991

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

This paper describes several improvements to Lee's multicast switch architecture. Our improvements make Lee's architecture practical, allowing it to achieve maximum network throughput under worst-case conditions and drastically reducing the amount of memory required for addressing of multicast cells. These improvements allow multicast to be added to a 256 port switch with 150 Mb/s links at a cost of about two additional chips per port.

A Practical Version of Lee's Multicast Switch Architecture

Jonathan S. Turner

One interesting class of ATM (*Asynchronous Transfer Mode*) switching systems uses sorting networks as a key component. Starlite is the name given to a particular architecture for such a switching system that was developed by Alan Huang and Scott Knauer at AT&T Bell Labs [5, 6, 7]. The Starlite architecture was motivated by the observation that sorting networks, can be used to construct rearrangeably non-blocking switching networks with distributed control. This observation was first put forward by Batcher [1] in 1968 in his seminal paper describing his *bitonic sorter* that sorts a set of n numbers using a network of approximately $(n/4)(\log n)^2$ simple comparison elements. For circuit switching applications, this observation leads to switching networks that are non-blocking, operationally very simple and eminently suited to VLSI implementation. To accommodate packet switching, mechanisms are needed to resolve contention between cells that arrive concurrently and are destined for the same output port. Multipoint communication requires additional mechanisms for cell replication.

A group at Bellcore has adopted this switching technique in an experimental ATM switching system referred to as Sunshine [2, 3, 4]. The Sunshine architecture can be extended to support multicast as described in [8]. Figure 1 includes a point-to-point switching system on the right preceded by several additional components to handle multicast. Cells are received at the *Port Processors* (PP) on the left where they are assigned a *Fanout* and a *Broadcast Channel Number* (BCN). The cells then pass through a concentrator network, which places the cells on consecutive outputs so as to ensure non-blocking operation of the subsequent networks. Next, the cells pass through a running adder network, which for the cell on output port i computes the sum of the fanouts of all cells entering on ports 0 through $i-1$ and places this sum in a field of the cell (this is done for all output ports, using a network with $n \log_2 n$ simple processing elements). Following the adder, the cells enter a set of *dummy address encoders* (DAC), which perform two functions. First, the DAC at output i sets a variable lo to be the sum computed by the adder network for output i and lets $hi = lo + f_i - 1$ where f_i is the fanout of the cell at output i. If hi exceeds the number of outputs of the entire network, the cell is discarded. The DACs return an acknowledgement to the sending input for each cell that is not discarded. The discarded cells are retransmitted later.

For the cells that are not discarded, lo and hi are inserted into the cell header as the cell is sent to the copy network. The copy network sends copies of each cell to all the outputs in

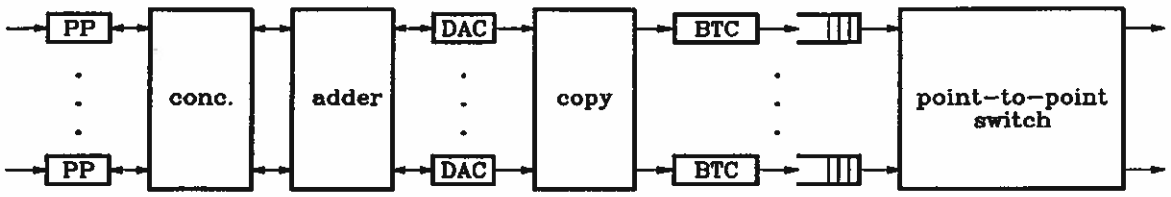

Figure 1: Lee's Multicast Switch Architecture

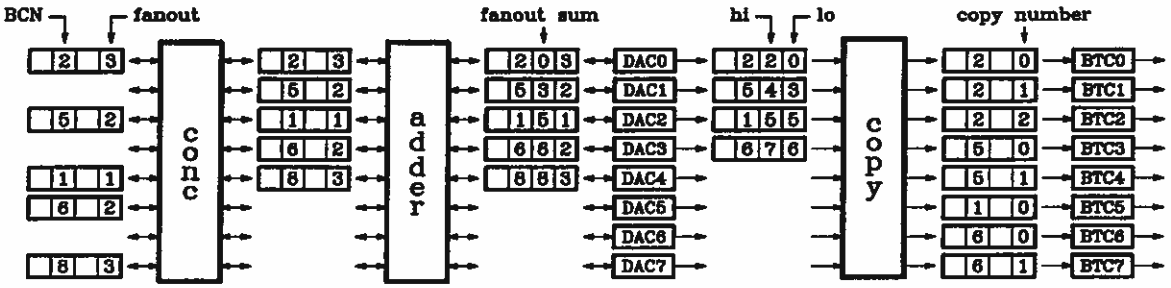

Figure 2: Example of Operation of Lee's Multicast Architecture

the range from *lo* to *hi*. The copy network also labels the copies and when each copy reaches a *Broadcast Translation Circuit* (BTC) its broadcast channel number and the copy number are used to produce an output address that the point-to-point switch uses to guide the cell to its ultimate destination. An example, illustrating the operation of Lee's architecture is shown in Figure 2.

Because of the way copying is managed, a single connection with a large fanout (say n) can prevent most cells that enter during a given cycle from passing through the network. For example, suppose the first input of the network has a cell with a fanout of 1 and the second input has a cell with a fanout of n, where n is the number of inputs to the switch. In this case, only the cell with the fanout of 1 passes through the DACs and all other cells are blocked. In the worst-case then, the DACs could pass just a single cell in each operation cycle. While this is admittedly a worst-case situation, it is not so far-fetched that it can be easily ignored.

There are essentially two ways to eliminate this problem. The first is to restrict fanouts to some value $< n$. For example, if we let the maximum fanout be αn where $0 < \alpha \leq 1$, then the number of cells that pass through the switch in a given operational cycle is guaranteed to be at least min $\{(1-\alpha)n, F\}$, where F is the sum of the fanouts of all the cells that enter the network in a particular cycle. So for example, if we limit the fanout of any single cell to $n/2$ the network can support a sustained throughput of at least $n/2$ cells per cycle.

The second way to improve the throughput of the network is to expand the copy network so that it can output more than n cells per cycle. In this approach, we use a copy network with n inputs and N outputs, followed by N BTCs, which then feed into n buffers (each buffer having $\lceil N/n \rceil$ inputs). Now, if we allow the maximum fanout to be αN, the network can support a sustained throughput of $(1-\alpha)N$ cells per cycle, so for example, with $N = 2n$,

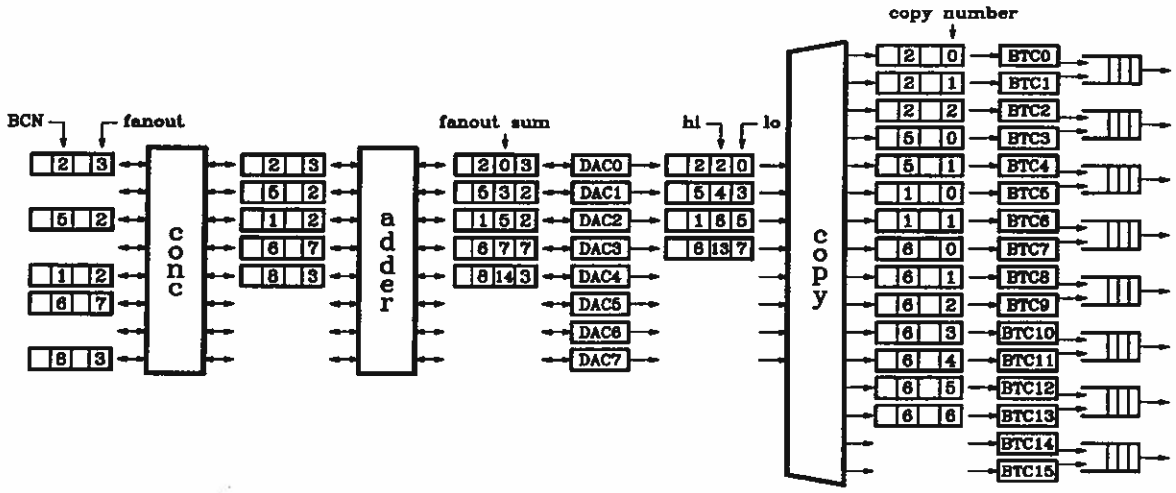

Figure 3: Example of System with Expanding Copy Network

we can support a maximum fanout of n and fully loaded input links. This is illustrated in Figure 3.

A second problem with Lee's original multicast architecture is that it requires very large memories for multicast address translation. Because each copy of a cell can go to any one of the broadcast translation circuits, the BTCs must each contain the information required to translate any one of up to n copies. If the total number of broadcast channel numbers is B (allowing the system to support B simultaneous multicast connections) and the number of bits in each BTC entry is W, each table requires nBW bits of storage. So, for example with $n = 256$, $B = 2^{16}$ and $W = 16$, each table would require 256 Mbits of memory. The revised version, using a copy network with $2n$ outputs, requires the same number of bits per BTC, but since it has twice the number of BTCs, the number of bits per input port of the system is $2nBW$ or 512 Mbits in the example given above.

There are three ways to reduce the memory requirement. The first is just to observe that with an ATM cell size of 424 bits, serial data paths and a clock rate of about 150 MHz, each BTC can be shared by a large number of lines. With 32 lines accessing a BTC sequentially, there is over 80 ns available for each memory access. Figure 4 illustrates a multichannel BTC, in which the arriving cells are delayed by different amounts on the left to allow sequential access to the common table, then delayed complementary amounts on the right to maintain the overall system synchronization. If s is the number of copy network outputs that share a given BTC then the memory requirement drops to $2nBW/s$. For $s = 32$, this reduces the memory requirement in the example configuration to 16 Mbits per input to the network.

The second step in reducing the memory requirement is to subdivide the set of broadcast channel numbers into two parts, one for *small fanout connections*, that is connections having a fanout no larger than some critical value f, and one part for *large fanout connections*. Notice that inherently, a network can support fewer large fanout connections than it can small fanout connections, so it makes sense to allocate most of the BCNs to small fanout connections. In particular, if B_s is the number of small fanout BCNs and B_ℓ is the number

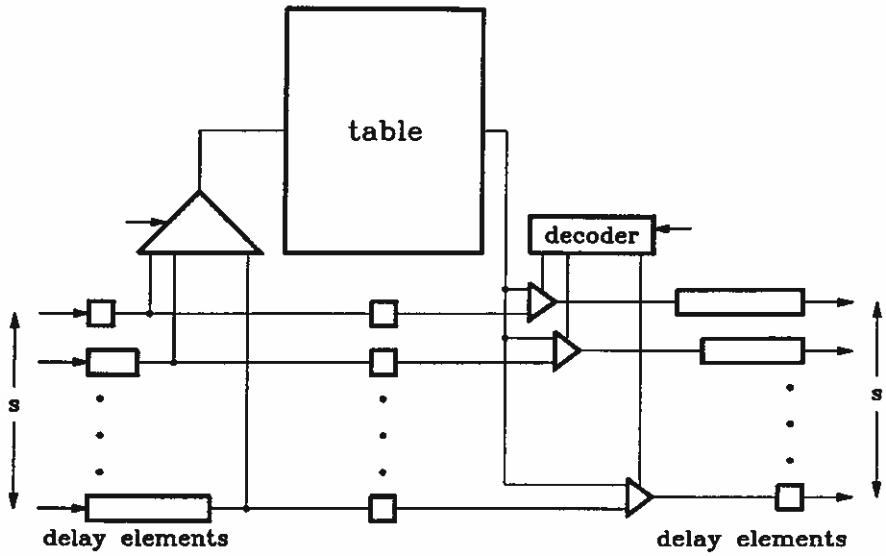

Figure 4: Sharing BTC's Among Copy Network Outputs

of large fanout BCNs, a reasonable choice is to let $B_s = fB_\ell$. To take advantage of this, the BTC memory is divided into two tables, one for the small fanout connections and one for large fanout connections. Since we need store the information for only f copies in the case of small fanout connections, the memory requirement in this case is $2W(fB_s + nB_\ell)/s$ per network input. When $B_s = fB_\ell$, this is minimized by taking $f \approx \sqrt{n}$. Since $B_s + B_\ell = B$, the total memory requirement is then approximately $4\sqrt{n}BW/s$ per input port or 2 Mbits in the example configuration.

The third improvement we can make trades off additional capacity in the network stages preceding the BTCs for reduced memory. The idea here is to constrain the routing of connections through the copy network so that only a subset of the BTCs can receive any given copy, hence eliminating the need to store the information about that copy in the remaining BTCs. We accomplish this by modifying the dummy address encoders to use "fanout aligned addresses." Let f_i be the fanout of the cell appearing on the input to DAC$_i$ and let $F_i = \sum_{j=0}^{i-1} f_i$ be the fanout sum which was computed by the adder network and placed in the cell header. As we have seen, the DAC replaces these two fields with values *lo* and *hi* and the copy network then copies the cell to all outputs in the range from *lo* to *hi*. In the original design $lo = F_i$ and $hi = F_{i+1} - 1$. We modify the scheme as follows. First, let $\overline{f}_i$ be the smallest power of 2 that is at least as big as f_i. Next, let *lo* be the smallest multiple of $\overline{f}_i$ that is at least as big as $3F_i$ and let $hi = lo + f_i - 1$. As before, the cell is accepted by the DAC if *hi* is no larger than N. Figure 5 shows an example of this algorithm. Observe that

$$3F_i \le lo < 3F_i + \overline{f}_i < 3F_i + 2f_i$$

and so

$$hi < 3F_i + 3f_i = 3F_{i+1}$$

This ensures that the range of copy network outputs selected by different cells are disjoint from one another. Notice also that if i is the index of the first DAC to reject a cell, then,

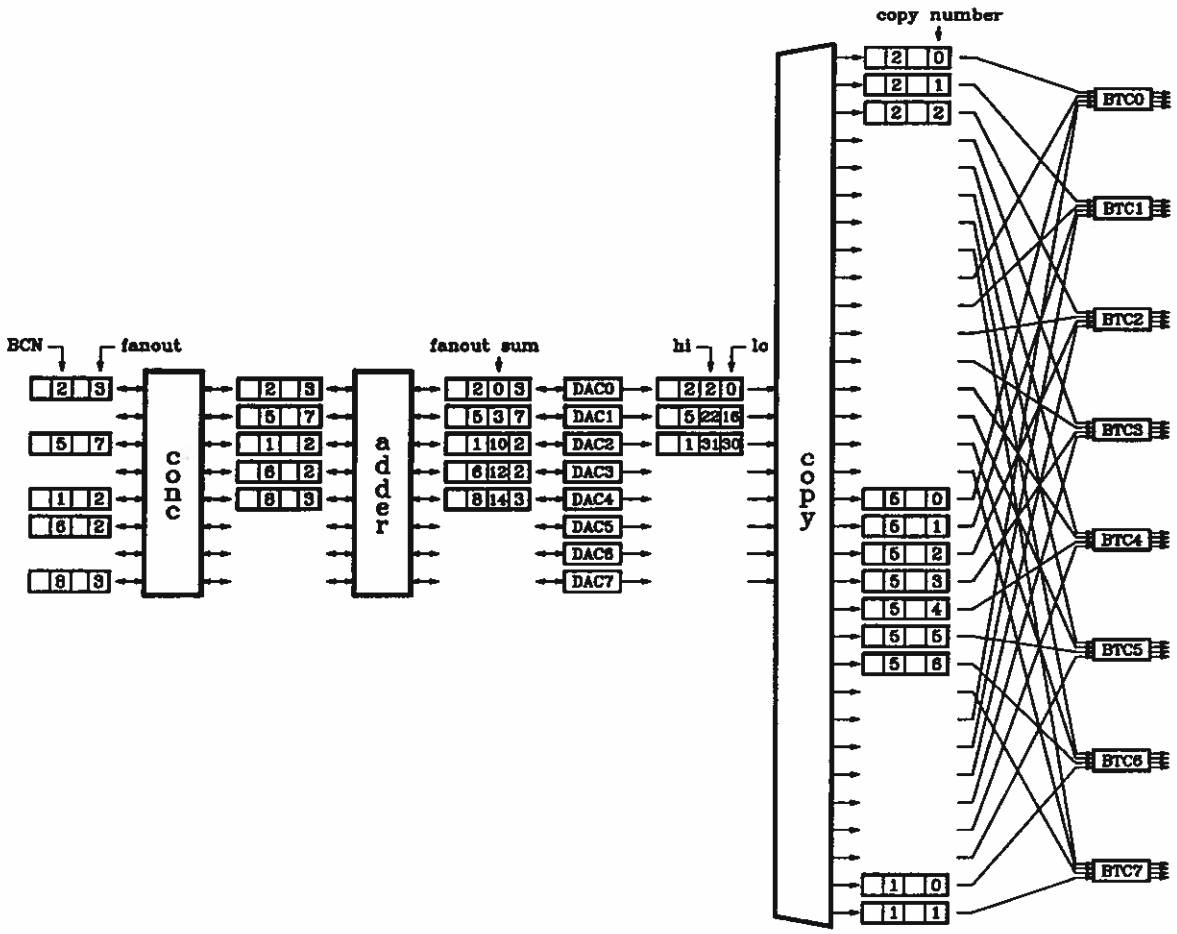

Figure 5: Copying with Fanout Aligned Addresses

$3F_i + (3f_i - 1) > N$, so F_i, which is the number of successful cells output by the copy network during that cycle is greater than $(N + 1 - 3f_i)/3$. Hence, if we let $N = 6n$, we can sustain a throughput of n cells per cycle.

The effect of using fanout-aligned addresses is that copy j of any given cell can only appear at a copy network output with index k where $k \bmod f_i = j$. This means that only the BTCs connected to those copy network outputs require the information about how to translate copy j. To obtain the largest possible reduction in the total amount of memory as a result of this change, the copy network outputs that share a common BTC must be spaced apart from one another, by a distance N/s positions and N/s must be constrained to be a power of 2. That is, copy network output k should be connected to BTC_h where $h = k \bmod N/s$. This is shown in Figure 5.

The number of bits of memory per input port, using fanout-aligned addresses is

$$\frac{WN}{sn}\left(B_s\left\lceil\frac{f}{N/s}\right\rceil + B_\ell\left\lceil\frac{n}{N/s}\right\rceil\right) \approx \frac{WN}{sn}\left(B\left\lceil\frac{f}{N/s}\right\rceil + \frac{B}{f}\left\lceil\frac{n}{N/s}\right\rceil\right)$$

For the example configuration, if we let $N = 6n$ and $s = 24$ (this choice ensures that N/s is

a power of two), the best choice for f is 64, giving a memory requirement of approximately 272 Kbits per input port.

To close, let's estimate the overall chip count required to add the multicast capability to a point-to-point network with 256 inputs and outputs. We will assume that for the chips that have little internal memory, we are constrained only by pin count, that each chip has 128 pins available for inputs and outputs and that the data paths are one bit wide. For the example configuration then, 32 chips are required for the concentrator, 16 for the adder, 6 for the DACs and 56 for the copy network. Assuming that each of the 64 BTCs consists of one memory chip and one control chip, we have 128 more chips for the BTCs. This gives a total of 238 chips or less than one per port. The FIFOs following the BTCs would add perhaps another chip each. Hence, the cost of adding a multicast capability to Lee's architecture is about two chips per port.

References

[1] Batcher, K. E. "Sorting Networks and Their Applications," *Proceedings of the Spring Joint Computer Conference*, 1968, 307–314.

[2] Beckner, M. W., T. T. Lee, S. E. Minzer. "A Protocol and Prototype for Broadband Subscriber Access to ISDNs," *International Switching Symposium*, 3/87.

[3] Day, C., J. N. Giacopelli and J. Hickey. "Applications of Self-Routing Switches to LATA Fiber Optic Networks," *International Switching Symposium*, 1987.

[4] Giacopelli, J. N., W. D. Sincoskie and M. Littlewood. "Sunshine: a High Performance Self Routing Broadband Packet Switch Architecture," *Proceedings of the International Switching Symposium*, 6/90.

[5] Huang, Alan and Scott Knauer. "Starlite: a Wideband Digital Switch," *Proceedings of Globecom 84*, 12/84, 121–125.

[6] Huang, Alan. "Distributed Prioritized Concentrator," U.S. Patent 4,472,801, 1984.

[7] Huang, Alan and Scott Knauer. "Wideband Digital Switching Network," U.S. Patent 4,542,497, 1985.

[8] Lee, Tony T. "Non-Blocking Copy Networks for Multicast Packet Switching," *IEEE Journal on Selected Areas in Communications*, 1455–1467, 12/88.