AN OPTIMAL NONBLOCKING MULTICAST VIRTUAL CIRCUIT SWITCH

Jonathan S. Turner
Washington University, St. Louis

Abstract

This paper describes an architecture for a multicast virtual circuit switch using cell recycling. This is the first nonblocking switch architecture that is optimal in both the switching network complexity and the amount of memory required for routing cells in multicast virtual circuits. Furthermore, it is optimal in the amount of effort required for multicast virtual circuit modification. This architecture makes it both technically and economically feasible to construct the large switching systems that will ultimately be needed for wide scale deployment of Broadband ISDN to residential users.

1 Introduction

Multicast virtual circuit networks support communication paths from a sender to an arbitrary number of receivers, as illustrated in Figure 1. As shown, multicast virtual circuits induce a tree in a network connecting a sender to one or more receivers. Switching systems participating in the virtual circuit replicate received cells using *virtual circuit identifiers* in the cell headers to access control information stored in the switching system's internal control tables, then use this information to identify the outputs the cells are to be sent to and relabel the copies before forwarding them on to other switching systems.

Figure 2 illustrates in more detail, the function of a multicast virtual circuit switch. The switch includes control information, shown here as a table, which for each incoming virtual circuit provides a list of outputs and outgoing virtual circuit identifiers. For a cell received on input link i and virtual circuit z, the switch forwards copies to outputs $j_1, j_2, \ldots$ after relabeling them with new virtual circuit identifiers, $y_1, y_2, \ldots$ Notice that if the switch has n inputs and outputs and each output supports up to m virtual circuits, one can describe any collection of multicast virtual circuits with mn words of memory. One simply provides for each (output,VCI) pair, the identity of the (input,VCI) pair from which it is to receive cells. Unfortunately, this method of defining a set of multicast connections is not particularly helpful in switching, as it does not give one a way to go from an (input,VCI) pair to the desired list of (output,VCI) pairs. Existing virtual circuit switch architectures describe multicast virtual circuits in different ways, which while suitable for switching, use far more than mn words of memory. The broadcast packet switch [5, 6], for example, requires $mn^2/2$ words of memory under worst-case conditions (although refinements described in [9] can reduce this to $O(mn^{3/2})$ or less, giving acceptable complexity for systems with up to a few hundred ports). Moreover, the time required to update a multicast connection grows with the size of the connection. Other architectures require even greater amounts of memory. For example, Lee's multicast switching system [3] requires $mn^3/2$ words of memory under worst-case conditions.

This paper describes a multicast switch architecture with $O(n \log n)$ hardware complexity that is nonblocking, in the sense that it is always possible to accommodate a new multicast connection or augment an existing one, so long as the required bandwidth is available at the external links, and which requires $< 2mn$ words of memory for routing cells in multicast virtual circuits. Moreover, the overhead for establishing or modifying a multicast connection is independent of the size of the connection or the switching network.

2 Basic Operation

The basic principle behind the recycling architecture is illustrated in Figure 3. To implement a mul-

[0] This work was supported by the National Science Foundation (grant DCI 8600947), Ascom Timeplex, Bell Communications Research, Bell Northern Research, Goldstar Information and Communications, Italtel SIT, NEC, NTT and SynOptics.

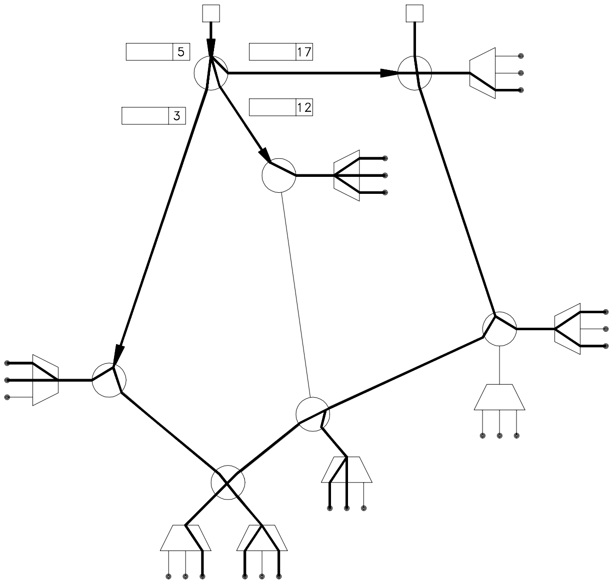

Figure 1: Multicast Virtual Circuit Switching

ticast connection, a binary tree is constructed with the source switch port at its root and the destination switch ports at its leaves. Internal nodes represent switch ports acting as relay points, which accept cells from the switch but then recycle them back into the switch after relabeling the cells with a destination pair identifying the next two switch ports they are to be sent to. There are many possibilities for constructing the switching network. A Beneš network in which the switches in the first half of the network distribute cells dynamically in order to balance the load, and in which local buffers are used to resolve contention, provides the lowest cost solution. This is illustrated in Figure 4 which shows a 16 port network of binary switches in which two cells with two destinations each are forwarded from inputs to outputs. Note that cells are copied at the latest possible point in the network and this point is easily determined by bit-wise consideration of the destination addresses. This scheme can easily be extended to networks constructed from larger switches. We show in a later section that given any collection of virtual circuits, the load placed on any of the switching network's internal links is at most equal to the load on the most heavily loaded external port. In other words, there is no collection of virtual circuits that can be handled by the external links that cannot also be handled by the network. That is, this network is nonblocking. Other switching networks, suitably extended to provide the copy-by-two function, can also be used in the recycling architecture.

The lower part of Figure 3 details the hardware

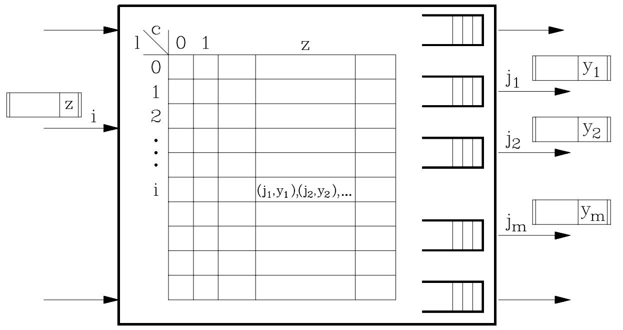

Figure 2: Multicast Switch Functionality

associated with each port of the switching system. Given a virtual circuit identifier, obtained from a cell's header, the *Virtual Circuit Translation Table* (VXT) provides two (output,VCI) pairs that are added to the cell header plus two additional bits that indicate, for each pair, whether it is to be recirculated another time, or not. The *Receive Buffer* (RCB) holds cells received from the input link that are waiting to enter the switching network, while the *Transmit Buffer* (XMB) holds cells waiting to be transmitted on the outgoing link.

Because networks that perform dynamic load distribution may deliver cells in a different order than that by which they enter, the ports are typically augmented with a resequencing buffer to restore the proper ordering on output [7]. The simplest resequencer implementations measure the time that cells spend in transit through the network and delay cells that pass through quickly in a resequencing buffer for a long enough period to ensure that cells that are delayed an unusally long period of time have a chance to catch up. In the recycling architecture, the resequencer also ensures proper ordering of cells during additions and deletions to multicast connections. The resequencing buffer is labeled RSQ in Figure 3.

Figure 5 illustrates the operation of the recycling switch in more detail. In this example, a multicast connection delivers cells from input a to outputs b, c, d and e, using ports x and y as relay points. In the lower part of the diagram, the implementation of the connection is shown in an 'unrolled' form, to clarify the flow of cells through the system. It should be understood however, that this is purely illustrative. There is in fact just one switching network, not three, and cells are simply sent through it multiple times in order to reach all the destinations. In the example, cells entering at input a with VCI i, are forwarded to output e, VCI k and output x, VCI j. At x, the cell is recycled, with VCI j used to select a new table entry

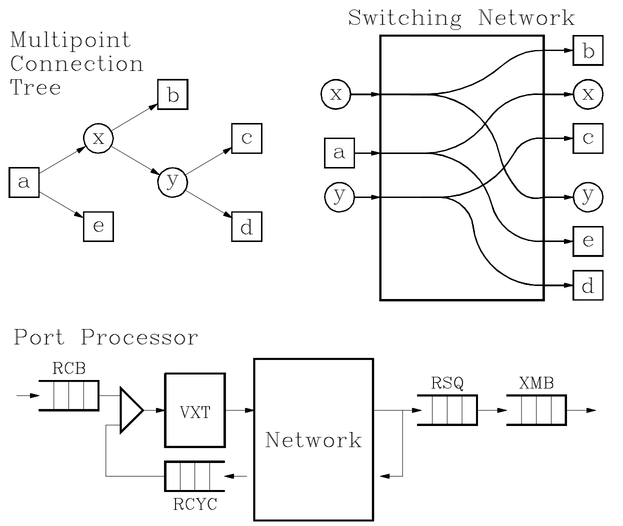

Figure 3: Multicasting by Recycling Cells

from x's VXT. The resulting information causes the cell to be forwarded to output b, VCI n and output y, VCI m. At y, the cell is recycled again, with the resulting copies delivered to c and d.

We can also construct multicast connections to which multiple input ports can send cells. One simply sets up the virtual circuit tables of each of the source input ports so that they forward cells to the port at the root of the tree, which then recycles them along the tree. Of course, the total traffic from all the source ports must be limited to the total bandwidth allocated to the connection. In a connection where a port is both a source and a destination, we will usually not want to send a source a copy of a cell that it sent in the first place (although we do want the other participants to receive it). This is easily accomplished by including the identity of the original source in the cell and checking this at the destination in order to discard unwanted copies. To add an endpoint to a multicast connection, some rearrangement of the connection is needed. This is illustrated in Figure 6. Let d be the output that is to be added to a connection, let c be an output closest to the root of the tree and let a be its parent. Select a switch port x with a minimum amount of recycling traffic. Enter c and d in an unused VXT entry at x and then replace c with x in a's VXT entry. These changes have the effect of inserting x into the tree, with children c and d, as illustrated in the figure. Dropping an endpoint is similar. Let c be the output to be removed from a connection and let d be its sibling in the tree, x be its parent and a its grandparent. In a's VXT entry, replace x with d. If the output to be removed has no grandparent but its sibling has children, replace the parent's VXT entry with the sibling's children. If the output to be removed has no grandparent and its sibling has no children, then we simply drop the output to be removed from its parent's VXT entry, and the connection reverts to a simple point-to-point connection.

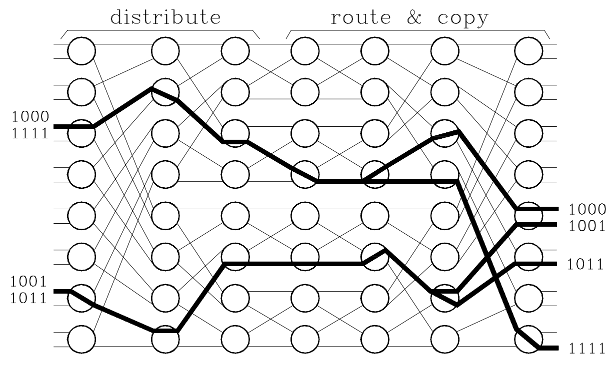

Figure 4: Beneš Network with Copy-Twice Routing

3 Resequencing Options

Because cells are resequenced when they exit from the system, the resequencing buffer must be dimensioned to delay cells long enough so that slow cells have a chance to catch up with fast cells. That is, the resequencing buffer must be at least as large as the largest variation expected in the delay of cells through the system, when they recycle the maximum number of times. Since, both the total delay and the delay variation can change over time, the most practical approach appears to be to dimension the buffer to be equal to the maximum delay that would be expected under the heaviest loading conditions.

A naive analysis reveals how the delay grows with the number of inputs and outputs to the system (n). Let μ and σ be the mean and standard deviation of the delay in each stage of the switching network. Let μ_t and σ_t be the mean and standard deviation for cells passing through the network the maximum number of times. Let r be number of stages of switching that these cells pass through, altogether. Then $\mu_t = r\mu$ and if the delays in each stage are independent (often a reasonable approximation), then $\sigma_t = \sqrt{r}\sigma$. A reasonable engineering rule is to select the resequencer depth equal to the mean delay plus some number h of standard deviations past the mean. This gives a resequencer depth of $\mu_t + h\sigma_t = r\mu + h\sqrt{r}\sigma$. Consequently, the depth grows in proportion to r and for a Beneš network, $r = (2(\log_d n) - 1)(\log_2 F)$ where F

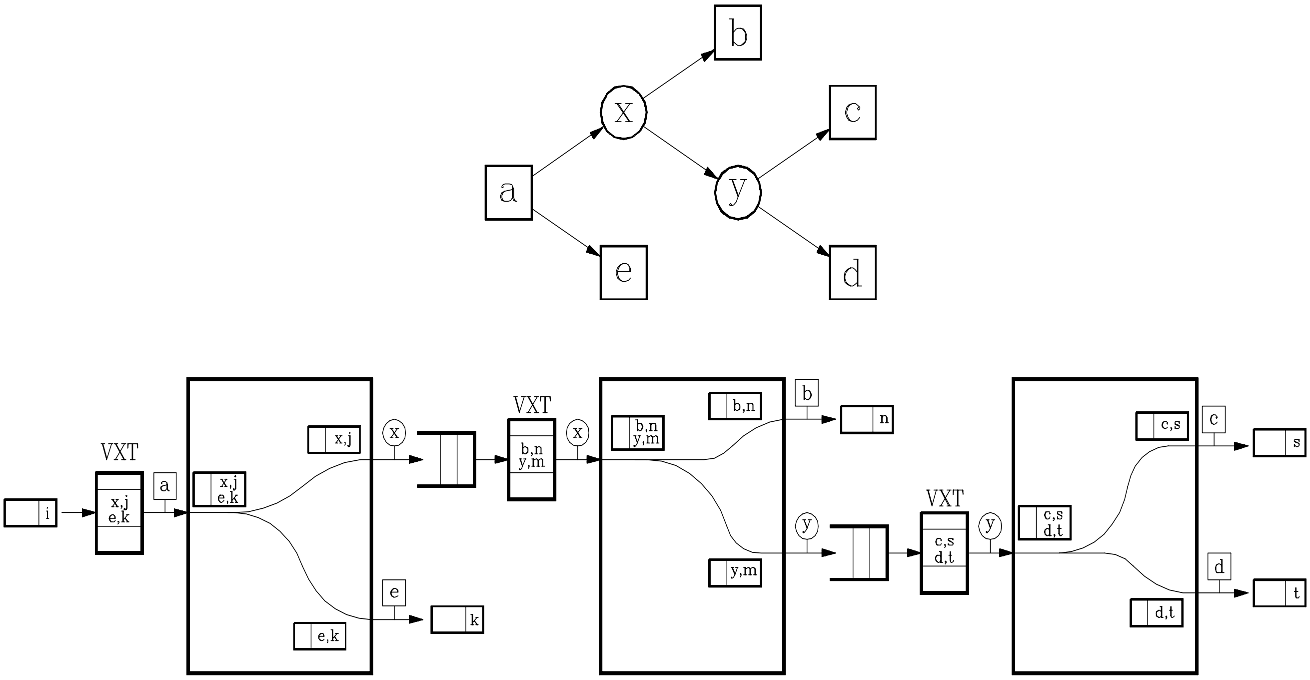

Figure 5: Example of Multicast Connection

is the maximum fanout. For $d = 2$ and $F = n$, this is too much if we are to obtain an overall system cost that grows in proportion to $n \log n$.

We can obtain the desired complexity by resequencing cells after every pass, rather than waiting until the cells exit. This raises a new issue however, in that when we modify a connection, we potentially change the depth of the tree. This means that cells take a different number of passes through the network and introduces the possibility of cells getting out of sequence (even though they are correctly sequenced on each pass). When an endpoint is added to a connection its new sibling becomes repositioned in the tree and its cells experience a longer delay, because of the additional pass through the network. Consequently, there is a momentary gap in the flow of cells to the output, but the ordering of the cells is unaffected. However, when an endpoint is removed from a connection, outputs immediately following the *cut point*, are moved closer to the root of the tree and so the cells being sent to them experience a shorter delay and are at risk of being mis-sequenced with cells that left the cut point just before the change.

To prevent cells from being delivered out of order, the resequencer must provide an extra delay for cells forwarded immediately after the cut occurs. Let T be the maximum delay we expect to see in one pass through the network (by the naive analysis, this would be equal to $(2(\log_d n)-1)\mu+\sqrt{2(\log_d n)-1}h\sigma$ for the Beneš network). Let τ be the moment when the VXT at the cut point is changed and let R be a new register included in the time stamping circuit of every input port processor. Assume the clock used for time stamping is incremented once for every operational cycle of the system (one cell time) and assume also that the time stamp field of the cell and the register R include an extra low order bit that can be used to represent a "half-step." Normally, cells are time stamped with the current time value. We modify this process for the affected virtual circuit in the time period immediately following the change in the following way. At time τ, the register R is set equal to $\tau + T$. After that time, cells in the affected virtual circuit are time stamped with either the current time or the value of R, whichever is larger. If R is chosen, we also add $1/2$ to R. This process compresses the time stamps in the period of length $2T$ following the transition into the time period $[\tau+T, \tau+2T]$ (see Figure 7). This ensures that cells immediately following the transition are delayed for an extra time period in the resequencer, giving cells that entered just before the transition, time to catch up and get placed in the proper sequence. The time stamping process returns to normal no later than $2T$ cycles following the transition. These same ideas can be generalized to allow, resequencing after every p passes for some p. See [8] for details.

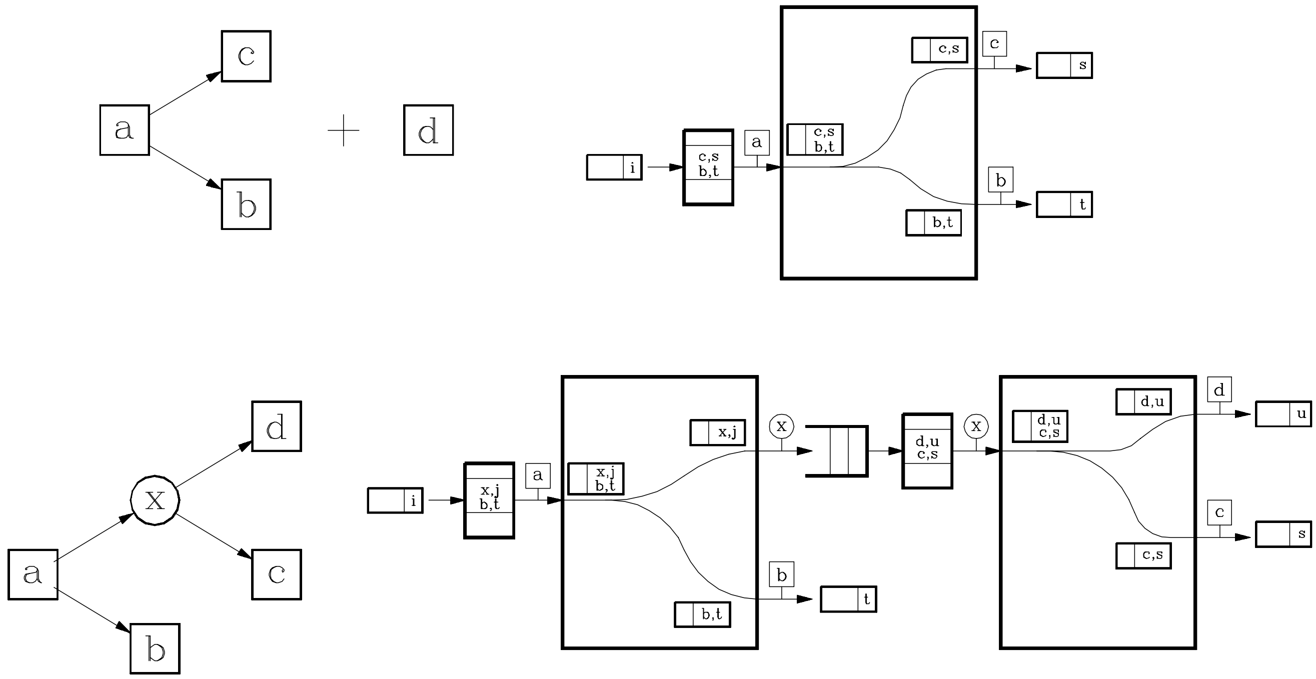

Figure 6: Adding an Endpoint to a Connection

			per pass reseq		multipass
n	r	$\sqrt{r}$	depth	max delay	$r\mu + h\sqrt{r}\sigma$
16	4	2	46	92	12+40=52
256	24	4.9	87	348	72+98=170
4K	60	7.7	119	714	180+155=335
65K	112	10.6	148	1184	336+212=548

Figure 8: Comparison of Per Pass Resequencing and Resequence on Exit

For the largest system, the per pass resequencing delay is 1184 cell times, or under 700 μs for a system configured to support external link speeds of 620 Mb/s. To put things in perspective, this is less than the delay in many existing digital telephone switches, so even the largest value in the table is quite reasonable. The resequencer depth in the largest case is getting fairly large, although it's arguably still acceptable, since the transmit buffer of the output port is likely to be at least as large. We'll introduce mechanisms in the next section which can improve both of these cases, but the point to be made here is that even without further refinements, excellent performance is possible.

4 Configuring the Network to Avoid Blocking

In this section we show that the recycling architecture can be configured so that it never blocks a new connection request if the rate of the network's internal data paths is sufficiently higher than that of the external links. We show that the necessary *speed advantage* is modest, making the recycling architecture practically useful. The analysis is in two parts. First we consider how the loading on the network's internal links depends on the loading of the network's ports (including the recycled traffic). Then we consider how the recycled traffic depends on the external traffic. By combining these two analyses, we obtain the speed advantage needed to make the system nonblocking.

To study how the internal link loading depends on the port loading, we apply an analytical technique developed in [4]. For the recycling architecture, we subdivide each multicast connection into "one-pass" segments, consisting of a single input and two outputs. With this understanding, we denote a connection by a triple (x, y, z, ω), where x is an input to the network, y and z are outputs and $0 < \omega < 1$ is a *weight*, which denotes the fraction of one of the network's internal data paths that would be consumed by the connection if it were to use that data path. (So, if the internal data paths operate at a speed of 800 Mb/s, a constant

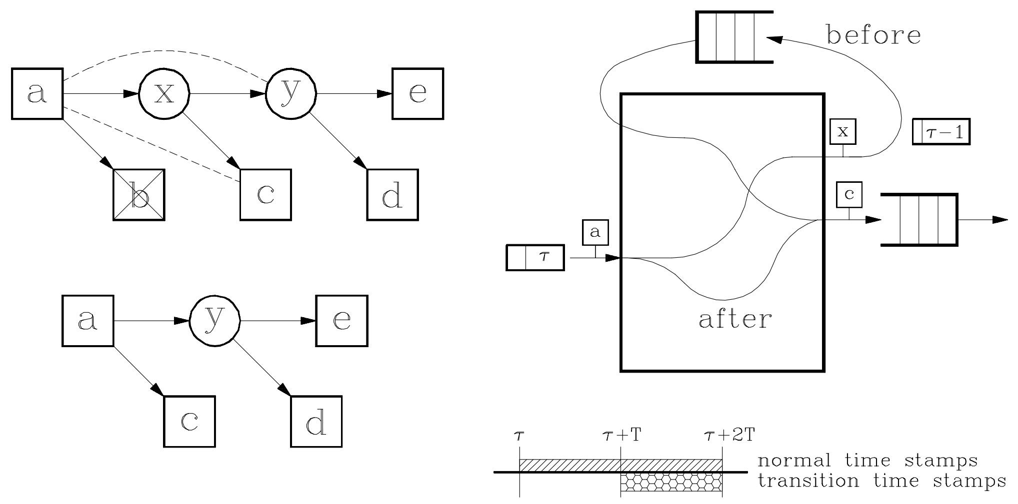

Figure 7: Maintaining Sequence During Transitions

rate virtual circuit operating at a rate of 100 Mb/s would have $\omega = 1/8$.)

We say that a connection *induces a load* on the links that lie on paths joining the connection's input and output ports. In particular, we assume that the load is distributed evenly whenever there are multiple paths to the desired destination. We let $0 < \alpha \leq 1$ denote the maximum load allowed on any of the network's input or output ports. A *connection assignment* C is a set of connections that satisfies this constraint and the load induced by C on link ℓ is denoted $\lambda_\ell(C)$. We say a network is *nonblocking* if for all connection assignments, $\lambda_\ell(C) \leq 1$ for *all* links ℓ.

The Beneš network is a special case of a class of networks known as *extended delta networks.* Extended delta networks can be defined using a combination of the *series* and *parallel* constructions of Cantor [2]. Let N_1 be a network with n_1 outputs, N_2 be a network with n_2 inputs and n_3 outputs and N_3 be a network with n_1 inputs. The series connection of the two networks N_1 and N_2 is obtained by taking n_2 copies of N_1, n_1 copies of N_2, numbering the copies of N_1 sequentially from 0 $(N_1(0), \ldots, N_1(n_2-1))$, numbering the copies of N_2 similarly and then connecting output j of $N_1(i)$ to input i of $N_2(j)$, for all pairs i, j. The resulting network is denoted $N_1 \times N_2$. The parallel connection of N_1, N_2 and N_3 is obtained by taking the series network just constructed plus n_3 copies of N_3 and connecting output j of $N_2(i)$ to input i of $N_3(j)$. This network is denoted $N_1 \bowtie N_2 \bowtie N_3$. We also let X_{d_1,d_2} denote a switch element with d_1 inputs and d_2 outputs. Let $d \geq 2$, $k \geq 0$, $0 \leq h < k$ be integers and let $n = d^k$. The extended delta network $D^*_{n,d,h}$ is defined by

$$D^*_{n,d,h} = \begin{cases} X_{d,d} & n = d \\ X_{d,d} \times D^*_{n/d,d,0} & n > d,\ h = 0 \\ X_{d,d} \bowtie D^*_{n/d,d,h-1} \bowtie X_{d,d} & n > d,\ h > 0 \end{cases}$$

When $h = 0$, the extended delta network is equivalent to the ordinary delta network and when $h = k-1$, it is equivalent to the Beneš network. The number of stages in the extended delta network is $h + k$.

A network with the extended delta topology can be used for copy-twice routing by using the first h stages to distribute cells dynamically across the network and using the last k stages to route cells to the desired output. Hence, for $h \geq 1$, if a connection places a load ω on some input x of the network, the load on the links exiting from x's first stage switch is $1/d$, the load on the links exiting from the subsequent second stage switches is $1/d^2$ and so forth. This spreading of the load continues for the first h stages and then stops. In the last h stages, the load builds up again, increasing by a factor of d at every stage. Let ℓ be a link in stage i of an extended delta network and let $c = (x, y, z, \omega)$ be a connection. From the above discussion, if there is a path in the network from x to ℓ and a path from ℓ to either y or z then

$$\lambda_\ell(c) = \begin{cases} \omega_j d^{-i} & 0 \leq i \leq h \\ \omega_j d^{-h} & h \leq i \leq k \\ \omega_j d^{-(k+h-i)} & k \leq i \leq k+h \end{cases}$$

Let C be any set of connections on $D^*_{n,d,h}$ and let C_ℓ be the subset of connections $c_j = (x_j, y_j, z_j, \omega_j)$ for which there is a path from x to ℓ and a path from ℓ to either y or z. Note that there are paths to ℓ from at most d^i inputs and from ℓ to at most d^{h+k-i} outputs. Because the load on every input and output is limited to α,

$$\sum_{c_j \in C_\ell} \omega_j \leq \alpha d^i \quad \text{and} \quad \sum_{c_j \in C_\ell} \omega_j \leq \alpha d^{k+h-i}$$

Thus, for $0 \leq i \leq h$,

$$\lambda_\ell(C) = \sum_{c_j \in C_\ell} \lambda_\ell(c_j) \leq d^{-i} \sum_{c_j \in C_\ell} \omega_j \leq \alpha$$

For $k \leq i \leq k+h$,

$$\lambda_\ell(C) = \sum_{c_j \in C_\ell} \lambda_\ell(c_j) \leq d^{-(k+h-i)} \sum_{c_j \in C_\ell} \omega_j \leq \alpha$$

For $h \leq i \leq k$,

$$\begin{aligned} \lambda_\ell(C) &\leq d^{-h} \sum_{c_j \in C_\ell} \omega_j \leq \alpha d^{-h} \max_{h \leq r \leq k} \min \left\{ d^r, d^{k+h-r} \right\} \\ &= \alpha d^{-h} d^{\lfloor (k+h)/2 \rfloor} = \alpha d^{\lfloor (k-h)/2 \rfloor} \end{aligned}$$

Hence we have proved the following theorem.

Theorem 4.1 *For any assignment* C *on* $D^*_{n,d,h}$, $\lambda_\ell(C) \leq \alpha d^{\lfloor (k-h)/2 \rfloor}$ *for all links* ℓ.

For $h = k-1$, this is simply $\lambda_\ell(C) \leq \alpha$. That is, the load on the internal links of the network is bounded by the load on the inputs and outputs. Hence, the only speedup needed in the network is that required to accommodate queueing effects. From [1] it is apparent that if the network is constructed from shared buffer switches with $d \geq 8$ and $4d$ buffer slots, a speedup of 20% can be sufficient.

We now consider the impact of the recycling strategy on the total traffic in the network. A binary tree with r leaves and in which every internal node has two children, has exactly $r-1$ internal nodes. Hence, a one-to-many connection with rate ω and r leaves, places a total load of ωr on the outgoing links and generates a recycling load of $(r-2)\omega$. A many-to-many connection (one with multiple transmitters as well as multiple receivers) creates an output load of $(r-1)\omega$ and a recycling load of $(r-1)\omega$. That is, the recycling load never exceeds the exiting load. Also, notice that recycling is used only for one-to-many connections with at least three destinations and for many-to-many connections with at least two. Call these the *recycling connections*.

Let β denote the ratio of the external link rate to the switch's internal data path speed and let δn denote the total traffic exiting the system from the recycling connections (where n is the number of inputs and outputs of the network). From the above discussion it is clear that the total recycling traffic is at most δn, so there is always some output port where the recycling traffic is $\leq \delta$. If $B \leq \beta$ is the maximum weight for a single connection then we can always accommodate a new connection if $\beta + \delta + B \leq 1$. The worst-case occurs when $\delta = B = \beta$; in this case a three-to-one speed advantage is needed to ensure that a new connection does not block. If however, $\delta = B = \beta/2$, a two-to-one speed advantage suffices. Note that since the required speed advantage is independent of n, the complexity of the switching network is $O(n \log n)$.

There are some variations on the recycling architecture that are worth considering. One variation is to add a distribution pass before copying begins, for those cells that must be recycled. In effect, this replaces the single connection tree with a group of γ parallel trees, all with the same leaves, but different internal nodes. The input port (or ports) sends each cell to the root of one of the trees, distributing them in a round-robin fashion. This means that the impact of single connection on a recycling port is reduced by a factor of γ, making the system nonblocking if $\beta + \delta + B/\gamma \leq 1$. If $\delta = B = \beta/2$ and $\gamma = 8$, we require a speed advantage of about 1.56. Obviously, this increases the amount of memory needed for forwarding multicast cells by a factor of γ, so there is a trade-off of memory vs. bandwidth that can be exploited to meet specific application requirements.

Another variation is to do the cell replication at the recycling port instead of within the switching network. This would allow a network designed for point-to-point switching to be used for recycling. However, there is the obvious drawback of increased traffic at the ports and a larger speed advantage. In particular, the required speed advantage goes up to $\beta + 2(\delta + B/\gamma)$.

A third option is to design the system so that it can make more than two copies per pass. A larger branching factor reduces the number of passes required to produce a given number of copies and can reduce the total recycling traffic. In particular, if the branching factor is b, it suffices to have $\beta + \delta/(b-1) + B/\gamma \leq 1$ if copying is done in the network and $\beta + b(\delta/(b-1) + B/\gamma) \leq 1$ if copying is done at the recycling ports. When multiple copies are separately addressed, we require additional space in virtual circuit translation tables and in cell headers. Another option is to make more than two copies in a pass but simply send all the copies to a contiguous set of outputs, so that separate addresses are not needed for each copy. This is easy to accomplish by designing the switch hardware to (optionally) interpret the two addresses in the header as lower and upper bounds for a range of outputs, rather than using the usual copy-by-two algorithm. All the copies would use the same virtual circuit identifier to recycle through the network, meaning that some portion of the table entries (and bandwidth) must be managed across groups of ports. Using this approach, large branching factors become practical and it's possible to dramatically reduce the number of passes required to support a given connection. In particular, if we use a fixed branching factor of b copies per pass (in order to simplify management of recycling bandwidth and table entries) along with the original copy-by-two mechanism, it's possible to reduce the number of passes from $\log_2 F$ (where F is the maximum fanout), to approximately $\log_2 b + \log_b F$. Additional reductions are possible with more sophisticated connection management algorithms.

5 Summary

The recycling architecture is the first nonblocking multicast virtual circuit switch architecture that is optimal in the switching network complexity, the amount of memory required for routing multicast cells and the amount of effort required for multicast connection modification. While recycling cells leads to suboptimal delay, this has a negligible impact in most practical applications, since the delays are well under one millisecond in the basic architecture and can be cut even further using refinements to the basic scheme.

This approach makes it both technically and economically feasible to construct switching systems with tens of thousands of high speed ports. Our group at Washington University is currently designing a system that can support up to 32,768 links with speeds of 620 Mb/s and 2.48 Gb/s. When configured with one third of the links operating at 2.4 Gb/s it is nonblocking for all connection assignments in which the maximimum bandwidth for a single virtual circuit is 1.2 Gb/s and one third of the total traffic is associated with multicast virtual circuits. The number of CMOS integrated circuits needed to implement the port processors and the switching fabric in a system with n ports is $(1.5 + (1/3)\log_2 n)n$, giving a cost of under 4.2 chips per port when $n = 256$ and 6.5 chips per port when $n = 32,768$. With chip costs of \$30 each, even the largest system has costs due to the switching components of under \$200 per port. The cost of other components (particularly the optical devices needed to interface to fiber optic transmission links) could drive the total system cost to well over \$1,000 per port, but future reductions in the cost of these other components can be expected to bring total costs down to a level that would permit widespread deployment of such gigabit switching systems.

References

[1] Bianchi, Giusseppe and Jonathan Turner. "Improved Queueing Analysis of Shared Buffer Switching Networks," *IEEE/ACM Transactions on Networking*, August 1993.

[2] Cantor, D. G. "On Non-Blocking Switching Networks," *Networks*, vol. 1, 1971, pp. 367–377.

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

[4] Turner, Jonathan S, "Fluid Flow Loading Analysis of Packet Switching Networks," *Proceedings of the International Teletraffic Congress*, June 1988.

[5] Turner, Jonathan S., "Broadcast Packet Switching Network," United States Patent #4,734,907, March, 1988.

[6] Turner, Jonathan S., "Design of a Broadcast Packet Network," *IEEE Transactions on Communications*, June, 1988.

[7] Turner, Jonathan S., "Resequencing Cells in an ATM Switch," Washington University Computer Science Department, WUCS-91-21.

[8] Turner, Jonathan S., "An Optimal Nonblocking Multicast Virtual Circuit Switch," Washington University Computer Science Department, WUCS-93-30.

[9] Turner, Jonathan S., "Progress Towards Optimal Nonblocking Multipoint Virtual Circuit Switching Networks," *Proceedings of the Allerton Conference*, 9/93, pp. 760–769.