Switching Systems

David K. Hunter

1. Introduction

Switching systems connect and disconnect users in a communications network by activating and deactivating switching devices. At first sight, this may seem trivial, but there is, in fact, a vast literature on the subject, much of which is highly technical. The purpose of this article is to provide a flavor of this subject and identify some of its principal features. The article starts by looking at some key results from the mathematical theory of switching, providing idealized models of real switching systems and yielding useful insights into them. After that, time switching and packet switching are discussed and some switching systems currently in the research stage using a new technology – optical switching – are described.

Before continuing, it is worthwhile to ask why switching systems are necessary at all. Suppose there is a communications network with N users. If there were no switching, it would be necessary to connect every user to every other user (Figure 1). Each user is connected to $N - 1$ other users, and the total number of connections is hence $N(N - 1)/2$. Division by 2 takes place because each link is assumed to be bi-directional. For a few users (say 5) this is quite feasible, requiring 10 links, but for a large telecommunications network with say 25 million subscribers, this evaluates to a staggering 312 499 987 500 000 links. Not only is this completely impractical and incredibly expensive, it is also extremely inefficient since individual subscribers tend to call only a small circle of other subscribers and hence very few links would ever be used.

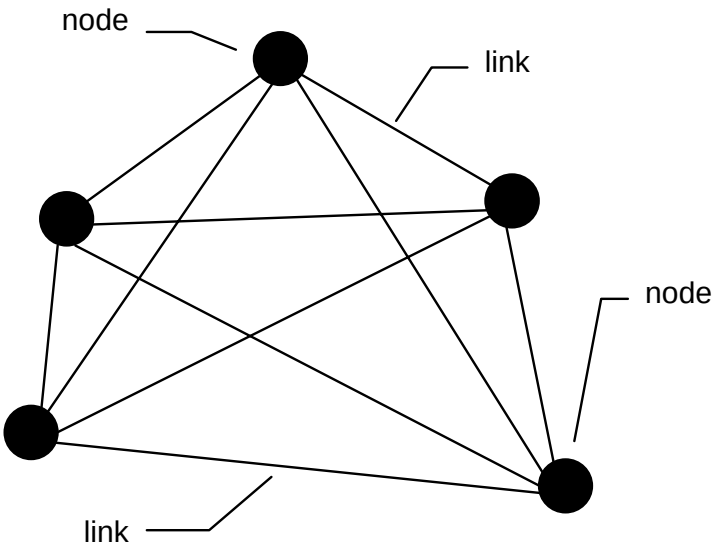

Figure 1: A fully interconnected network with $N = 5$ nodes.

One solution involves connecting all the nodes to a central node (or a switching center – Figure 2), requiring only N links and overcoming the problems mentioned above. In practice, in large networks, to further optimize the cost, there is a hierarchy of switches so that 25 million subscribers would be distributed over many switching centers. For example, switching centers such as that in Figure 2 would constitute nodes in the next tier of the network. The switches which connect directly to the subscribers are referred to as being at the lowest tier of the network hierarchy, while those furthermost removed are on the top tier.

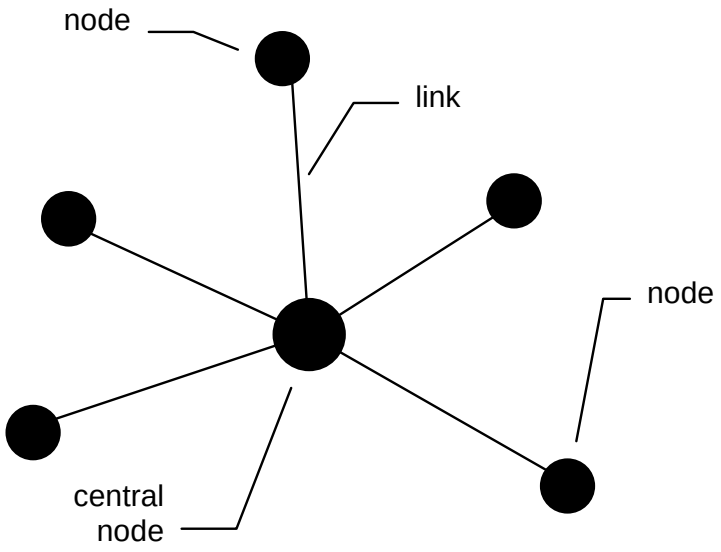

Figure 2: A star network with $N = 5$ nodes and a central node or switching center.

In telecommunications networks, there are two types of node:

- *crossconnects*, representing the highest tiers of the network, which set up semi-permanent *paths* in response to faults and changing traffic patterns over the network, and

- *switches*, which connect up *circuits*, corresponding to customers' calls.

Many circuits may be combined into a higher capacity signal and carried over a single path. The architectural design and performance evaluation of switches and crossconnects are similar, and what follows applies equally to both. For the remainder of this article, the term "switch" will often be used loosely in both cases.

2. The mathematical theory of switching networks

Many terms used in telecommunications switching have several subtly different meanings, depending on context. Above, the term "network" meant a telecommunications network, that is, a network of switching nodes and subscribers, interconnected by links. It may also mean a "switching network", which is a collection of small elementary switching units which constitute an entire switching

system. Confusingly, the term "switch" can either mean one of these elementary units or a whole switching system.

2.1 Terminology

The simplest type of switching network is the *crossbar switch* (Figure 3), a rectangular array of switches, called *crosspoints*, each of which can connect one of the N input lines to one of the M output lines. The black circles in the diagram each represent a crosspoint. This is referred to as an $N \times M$ switch, and is drawn as a box with N inputs and M outputs. 2×2 switches are of particular interest since they often result in the most economical implementation when used as "building blocks" in other architectures.

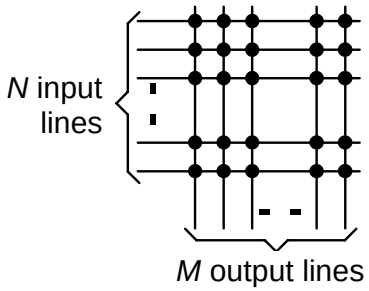

Figure 3: An $N \times M$ crossbar switch.

Most switching networks are organized into *stages*. Each stage consists of a column of switches. In each stage, except for the last one, each switch output is connected to a switch input on the next stage. The set of connections between adjacent stages is known as an *interconnect*. The inputs of the first stage switches form the input terminals of the switching network, and the outputs of the final stage switches form the output terminals. All the switching networks that are considered here have the same number of inputs as outputs. An example of a three-stage network is shown in Figure 4; stages 1 and 3 are made from 2×2 switches and stage 2 is made from 3×3 switches. It is assumed throughout that signals travel from left to right through a switching network.

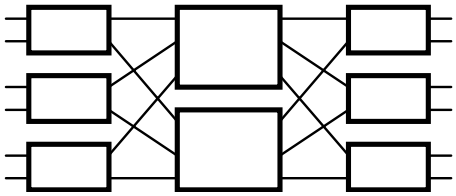

Figure 4: A three-stage switching network.

A *call* in a switching network is a connection between an input terminal and an output terminal; here, this may also mean a path in a crossconnects. The switching network has a controller associated with it which accommodates new calls as they arrive, and disconnects old calls once they finish. An *assignment* is a set of calls which are in progress, where each input or output terminal can carry at most one call. An input or output terminal is *free* if it does not carry a call. A *maximal assignment* is an assignment where no input (or output) terminals are free.

A switching network is said to be *blocking* if there are one or more assignments that it cannot realize, so it is not always possible to set up a new call between a pair of free input and output terminals. Switching networks which are not blocking are said to be *nonblocking*; here they are subdivided into two types. In a *strict-sense nonblocking* network, there is always at least one free route through the network for a new call, where the call may be set up without rerouting existing calls. Any free route may be used without blocking ever taking place. In a *rearrangeably nonblocking* network, new calls can always be accommodated, but it may be necessary to reroute existing calls through the network to do this.

2.2 Clos networks

Many switching networks are based on a type of three-stage network that was first studied by Clos, and are generally referred to as *Clos networks* (Figure 5). The first stage consists of r $n \times m$ switches, the second stage m $r \times r$ switches, and the third stage r $m \times n$ switches. Each switch in stages 1 and 2 has exactly one link to each switch in the following stage.

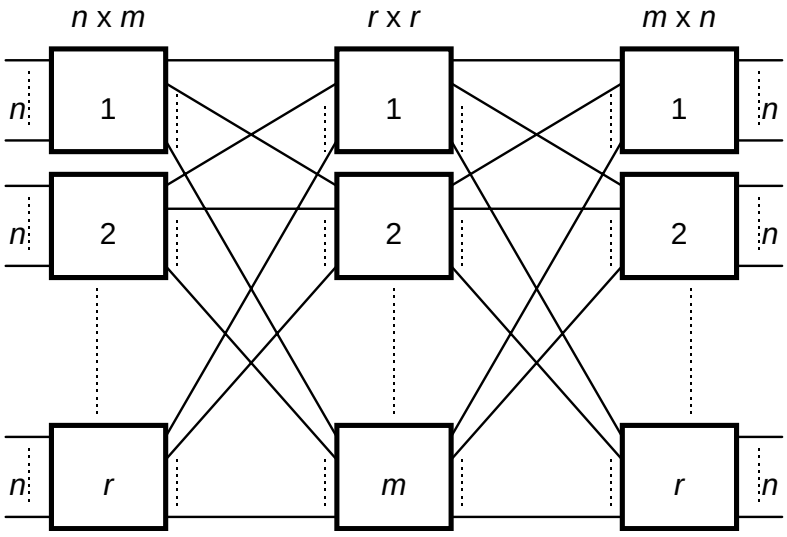

Figure 5: A three-stage Clos network.

If $m \geq 2n - 1$, a Clos network is strict-sense nonblocking [1]. To prove this, it is sufficient to show that, irrespective of how the existing calls are set up, there is always a free path through the network for a new call. Suppose a new call is to be set up between an input on a first stage switch **A** and an output on an output switch **B**. The worst case is shown in Figure 6; $n - 1$ calls will be already carried through switch **A**, so $n - 1$ center stage switches are used up. Also, $n - 1$ calls pass through **B**, each of them using up a further center stage switch. Since it is the worst case, none of the center stage switches used by calls in **A** are used by calls in **B**. Thus $2n - 2$ center stage switches are used in the center stage. To provide a free path from **A** to **B**, a further center stage switch must be provided, so there must be at least $2n - 2 + 1 = 2n - 1$ center stage switches.

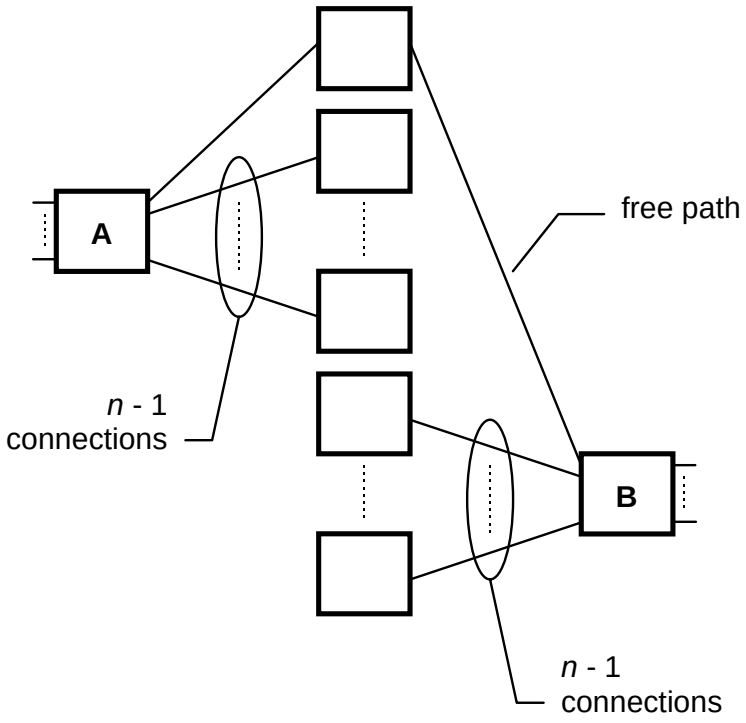

Figure 6: The worst case number of center stage switches required in a Clos network: $m = 2n - 1$.

In the days of early electromechanical systems, the attraction in using this switch architecture, rather than one large crossbar switch, was in the reduction in the number of crosspoints, since this was then a good indication of cost. Nowadays, this cost reduction benefit has become less pronounced with the advent of VLSI (Very Large Scale Integration), although crosspoint count is still a crude but useful guide to cost.

For large systems, a further reduction in crosspoint count can be made by replacing each center stage switch by another three-stage Clos network, or subnetwork. The resulting network would have five stages. By repeatedly applying this procedure, networks with 7, 9, 11,... stages may be produced. As the number of inputs and outputs become larger, the number of stages required to obtain an optimal crosspoint saving increases.

2.3 Blocking in Clos networks; the Lee and Jacobaeus methods

The blocking probability of a switching network is the proportion of time that a free input-output pair cannot be connected. To calculate the blocking probability, a simplified model of the three-stage Clos network is used (Figure 7). Let $u = n - 1$, the number of potentially active links on the input and output switches which will carry the new call, if it can be carried.

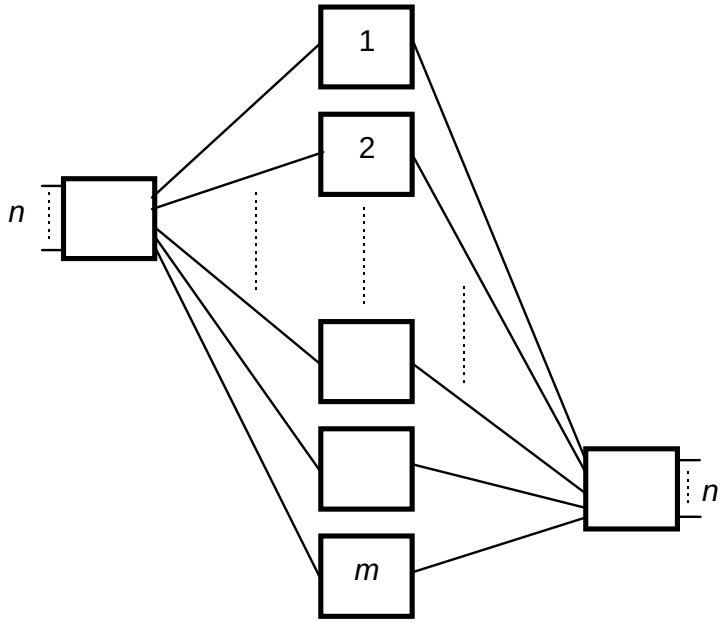

Figure 7: Simplified model of a three-stage Clos network for blocking calculations.

The first approximation to be considered was proposed by Lee [2]. If p is the probability that a given input or output link is busy, the probability that a given link is busy between the input or output switches and the center stage is up/m. The probability of a particular path being free between the input and output switches in question is $(1 - up/m)^2$. Since there are m potential paths through the Clos network, the blocking probability is approximately $[1 - (1 - up/m)^2]^m$. The assumption has been made that all links within the network are independent; if a certain link is being used, it does not affect whether other links are used. This gives a high estimate of blocking probability, but the Jacobaeus method described below [2] is more accurate.

In the Jacobaeus method, the "independent link assumption" made above is relaxed, so that the traffic on one side of the network is only independent from that on the other side. As usual, the number of ways that b things can be chosen from a things is indicated by:

$$\binom{a}{b} = \frac{a!}{b!(b-a)!}.$$

Initially, assume there are i input calls already entering via the same first stage switch as the free input terminal to be connected, and there are j calls already leaving the Clos network (output calls) via the same output stage switch as the free output terminal to be connected. Hence $0 \le i \le u$, and $0 \le j \le u$. Furthermore, assume that for the purposes of the calculation, any assignment of output calls to center stage switches is possible, but that the assignment of input calls to center stage switches has already been decided. This reflects the fact that the *relative* configurations of ingress switch and egress switches is of relevance.

There are $\binom{m}{j}$ ways of assigning the j output calls to the m center stage switches. The number of these assignments which result in blocking is $\binom{i}{m-j}$. This is the number of cases in which the remaining $m - j$ center stage switches coincide with $m - j$ of the i input calls, which is the number of subsets containing $m - j$ of these calls. The probability that blocking occurs is hence:

$$\beta_{ij} = \frac{\binom{i}{m-j}}{\binom{m}{j}} = \frac{i!\,j!}{(i+j-m)!\,m!}$$

The formula is only valid if $i + j \geq m$; if $i + j < m$, $\beta_{ij} = 0$. This implicitly includes the Clos result since $i \leq u$ and $j \leq u$, so $i + j$ cannot be greater than $2u = 2n - 2$, and for a strict-sense nonblocking Clos network, $\beta_{ij} = 0$ always. Let f_i be the probability that there are already i active links on the input switch and g_j be the probability that there are already j active links on the output switch. The blocking probability is:

$$P_B = \sum_{i=0}^{u} \sum_{j=0}^{u} f_i g_j \beta_{ij}$$

The probability of i links from the input switch being busy may be assumed to follow the binomial distribution:

$$f_i = \binom{u}{i} p^i (1-p)^{u-i}$$

where p is the probability that a particular link entering the input switch is busy. The expression for the links leading to the output switch is also assumed to follow the binomial distribution:

$$g_j = \binom{u}{j} p^j (1-p)^{u-j}$$

After much algebraic manipulation, the probability of blocking can be written as:

$$P_B = \frac{(u!)^2 (2-p)^{2u-m} p^m}{m!(2u-m)!}$$

Blocking probabilities for the Lee and Jacobaeus approximations are tabulated in Table 1. Not surprisingly, the blocking probability decreases as m (the number of center stage switches) is increased, and it increases as the load on the switch is increased. The table also confirms that the Lee approximation yields results that are consistently higher than the Jacobaeus method.

n	m	$p = 0.6$, Lee	$p = 0.6$, Jacobaeus	$p = 0.75$, Lee	$p = 0.75$, Jacobaeus	$p = 0.9$, Lee	$p = 0.9$, Jacobaeus
64	64	7.97×10^{-6}	7.16×10^{-6}	0.0107	0.0101	0.433	0.428
64	68	3.25×10^{-7}	1.65×10^{-7}	0.00130	0.000899	0.149	0.131
64	72	1.01×10^{-8}	2.30×10^{-9}	0.000117	4.81×10^{-5}	0.0359	0.0243
64	76	2.45×10^{-10}	1.91×10^{-11}	7.99×10^{-6}	1.54×10^{-6}	0.00631	0.00268
120	128	8.04×10^{-13}	2.41×10^{-13}	4.53×10^{-6}	2.36×10^{-6}	0.0315	0.0252
240	256	1.25×10^{-24}	1.33×10^{-25}	3.38×10^{-11}	1.01×10^{-11}	0.00133	0.000893

Table 1: Blocking probabilities for Clos networks under the Lee and Jacobaeus approximations, with different traffic intensities.

2.4 Rearrangeably nonblocking Clos networks

The Slepian-Duguid theorem [3] states that a Clos network is rearrangeably nonblocking if $m \geq n$. To prove this, it is sufficient to consider a maximal assignment with $m = n$.

This result relies on a combinatorial theorem due to Hall [4], which is often called "Hall's marriage theorem" because it may be stated informally like this. Suppose there are r boys and r girls, and that each boy must be paired off with a girl that he knows. For this to be possible, it is a sufficient and necessary condition that if any group of boys (say k) are selected, the number of girls they know between them must always be at least k. It is clear that this condition is necessary, but it is surprising that it is also sufficient.

Each of the r boys corresponds to a different input stage switch in the Clos network and each girl corresponds to a different output stage switch. A boy is said to know a girl if a call passing through his switch also passes through the girl's switch. Each subset of k boys must collectively know at least k girls, because k input switches carry kn calls (remember a maximal assignment is being considered with n calls in each input stage switch). These kn calls cannot be distributed over fewer than k output switches.

Hence each input stage switch can be paired off with an output stage switch carrying one of its calls, via a one-to-one mapping, and the r calls that this represents (one for each pair of switches) can be routed through the top center stage switch since one call comes from each input stage switch and one call goes to each output stage switch. Now remove this top center stage switch from the network and reduce all the input and output stage switches to $n - 1 \times n - 1$. Continue likewise until the input and output stage switches are all of size 1×1, and all calls in the Clos network have been assigned to a center stage switch.

2.5 Beneš networks

Beneš networks [3] (Figure 8) are rearrangeably nonblocking Clos networks with $m = n = 2$. Assume that N in Figure 8 is an integral power of 2. If $N = 4$, each center stage switch is a single 2×2 switch. If $N > 4$, each center stage switch can be replaced by a $N/2 \times N/2$ Beneš network. By repeating this substitution, a rearrangeably nonblocking Beneš network of arbitrary size can be produced. Figure 9

shows a Beneš network with 16 inputs and 16 outputs. These networks are organized into $2\log_2 N - 1$ stages of $N/2$ switches, where $\log_2 N$ is a logarithm to base 2. For example, $\log_2 2 = 1$, $\log_2 4 = 2$ and $\log_2 8 = 3$. The total number of switches required is $N\log_2 N - N/2$; it can be shown that this is very close to the lower bound for nonblocking operation [5].

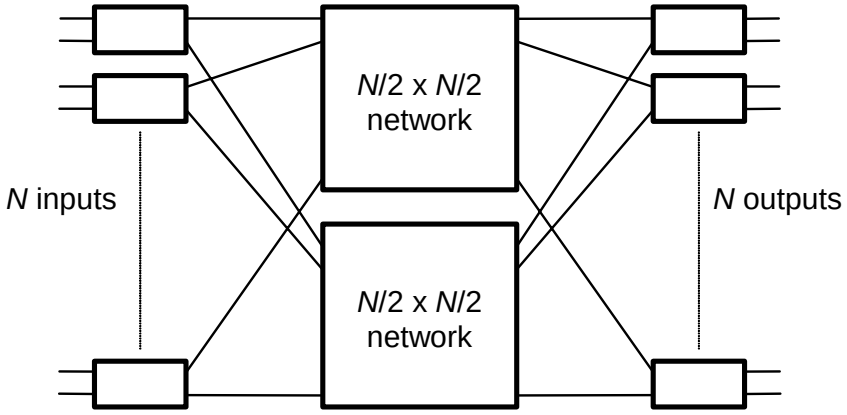

Figure 8: Recursive definition of an $N \times N$ Beneš network.

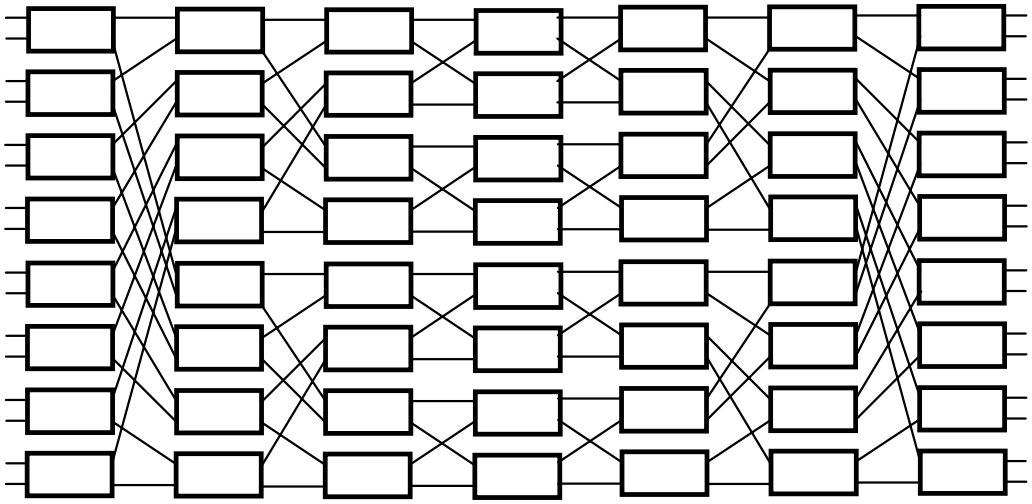

Figure 9: A 16 × 16 Beneš network.

2.6 Routing in Beneš networks

Because a Beneš network is rearrangeable, the switch setting may be dramatically changed even if only one new call is set up, which can be as complicated as re-connecting all input-output pairs. The *looping algorithm* is the simplest algorithm to do this [6]. It will be assumed that a maximal assignment is involved; the algorithm is easily modified if this is not the case.

Let the inputs be denoted by $u_1,\ldots,u_N$ and the outputs by $v_1,\ldots,v_N$. π is a mapping of the inputs to the outputs, representing a maximal assignment, and π^{-1} is its inverse, mapping outputs onto inputs. For any input u_i, the other input sharing the same first-stage switch is denoted by "co u_i", and similarly, "co v_i" is the output

sharing the same output switch as v_i. Initially, all the inputs and outputs are unconnected; the algorithm connects them up as follows, where the variables *S* and *T* represent various inputs and outputs respectively while the algorithm runs:

1. Select any unconnected u_i and set $S = u_i$. If no such input exists then the algorithm terminates since all the inputs are now connected to an output.

2. Connect S to $\pi(S)$ through the top subnetwork in the center stage.

3. Set $T = \text{co } \pi(S)$.

4. Connect T to $\pi^{-1}(T)$ via the lower subnetwork in the center stage.

5. Set $S = \text{co } \pi^{-1}(T)$.

6. If *S* has not been connected to an input, go back to step 2. Otherwise, a "loop" has been completed, so go to step 1 to start a new loop.

The algorithm works by traversing the network between the inputs and the outputs, until it gets back to its starting point, thus forming a loop. An example of a loop, with arrows showing how it was created, is shown in Figure 10; Table 2 and Table 3 show the complete assignment that is to be realized. For each input $u_1,\ldots,u_8$, Table 2 shows the corresponding output (one of $v_1,\ldots,v_8$) which it is to be connected to. Table 3 relates each output to its corresponding input in a similar way. The loop in Figure 10 contains 4 calls, but a loop may vary in size from 2 to N calls, and there can be from 1 to $N/2$ loops, depending on the maximal assignment being realized.

u_i	$\pi(u_i)$
u_1	v_4
u_2	v_2
u_3	v_6
u_4	v_8
u_5	v_5
u_6	v_7
u_7	v_3
u_8	v_1

Table 2: Assignment to be realized; for each Beneš network input, the corresponding output is shown.

v_i	$\pi^{-1}(v_i)$
v_1	u_8
v_2	u_2
v_3	u_7
v_4	u_1
v_5	u_5
v_6	u_3
v_7	u_6
v_8	u_4

Table 3: Inverse of assignment to be realized for the 8-input Beneš network.

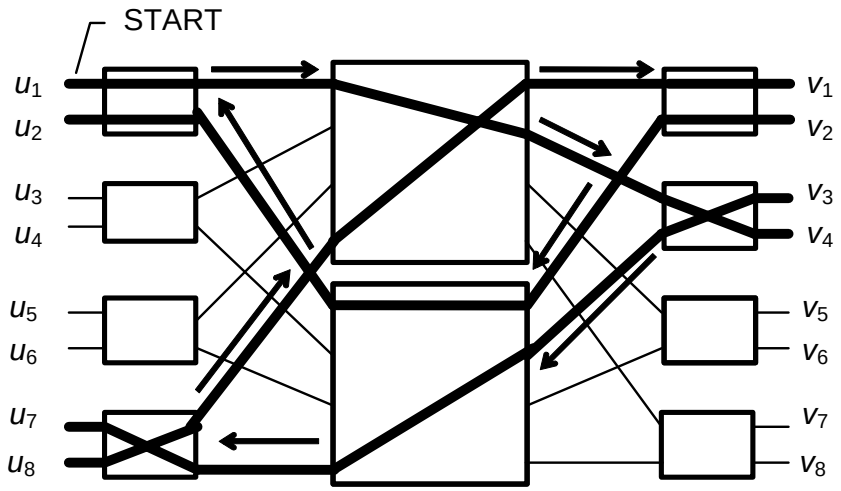

Figure 10: One loop in a Beneš network.

Calculating the switch settings and subnetwork subassignments in this way takes $O(N)$ time since N calls must be set up sequentially. By applying the algorithm repeatedly for each stage in the Beneš network recursion, routing of a complete network (including each subnetwork and their respective subnetworks etc.) can be carried out in $O(N\log N)$ time, meaning that the execution time for the algorithm is roughly proportional to $N\log N$. Hence one of the major disadvantages of a Beneš network compared to other types is the complex control required to accommodate even one new call. It is possible to ameliorate this situation to some extent by employing parallel processors [7], although the algorithm does not adapt to parallel implementation very efficiently.

2.7 Batcher sorting networks

In a sorting network, each input is tagged with a value, and the network sorts these inputs according to the tag value. A sorting network can thus be used as a switching network, although it is much more versatile. For maximal assignments, each input is simply labeled with the output it is destined for. In this section, the Batcher sorting network [8] is introduced. Since each switch can be controlled independently,

without a central controller, this network is said to be self-routing, and, like the Beneš network, it can be defined recursively.

It is based on the well-known sort-merge procedure. An $M \times M$ merge network takes two M-element lists which are already sorted into ascending order, and combines them into a $2M$-element sorted list. Based upon such a network, an $N \times N$ sorting network can be constructed as shown in Figure 11 where $M = N/2$. In other words, an $N \times N$ sorting network may be constructed from two smaller $N/2 \times N/2$ sorting networks and an $N/2 \times N/2$ merge network. Assuming a method of constructing merge networks exists, one can build an arbitrarily large sorting network by starting with a 2×2 sorting network. This is simply a 2×2 switch which is always set so that the element with the highest tag leaves on the lower output. This can be used to define a 4×4 sorting network which can in turn be used to define an 8×8, and so on.

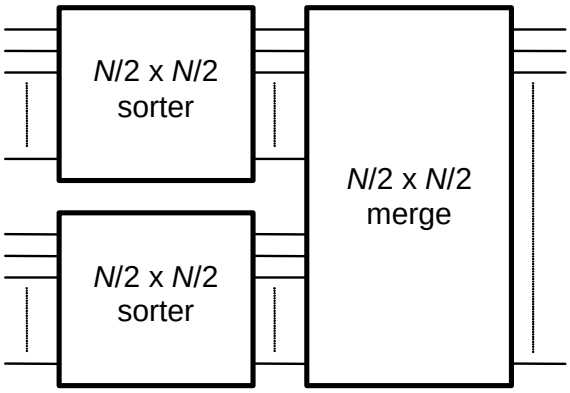

Figure 11: Recursive definition of an $N \times N$ sorter.

The merge networks may be implemented using Batcher's *bitonic sorter* [8], which is introduced below, but first, *bitonic sequences* must be defined. Suppose $a_0, a_1, \ldots, a_{M-1}$ is bitonic. Then either

1. There exists some $j \in \{0, \ldots, M-1\}$ such that $a_0 \le a_1 \le \cdots \le a_j \ge \cdots \ge a_{M-1}$.

2. The sequence can be modified to form a sequence of type 1 above by taking $k \in \{1, \ldots, M-1\}$ members from the right-hand side of the list and replacing them before the left-hand side, in the same order.

For example, 1, 3, 4, 5, 6, 4, 1, 1 is a bitonic sequence corresponding to 1 above. 4, 5, 6, 4, 1, 1, 1, 3 is also bitonic because it can be reduced to the first type of sequence by taking 1 and 3 from the right-hand side and moving them to the left-hand side.

Suppose that the sequence $a_0, a_1, \ldots, a_{2N-1}$ is bitonic. Define

$$d_i = \min(a_i, a_{N+i})$$
$$e_i = \max(a_i, a_{N+i})$$

for $i \in \{0, \ldots, N-1\}$. Then $d_0, \ldots, d_{N-1}$ and $e_0, \ldots, e_{N-1}$ are each bitonic, and:

$\max(d_0,\ldots,d_{N-1}) \le \min(e_0,\ldots,e_{N-1})$ [8].

A proof of this theorem, which provides the basis for the Batcher bitonic sorter, may be found in the Appendix (Section 8). A bitonic sorter takes a bitonic sequence and sorts it into ascending order. It can be used as a merge network simply by inverting the lower sequence on its inputs (Figure 12**)**. If both sets of *M* inputs are sorted, the input to the bitonic sorter is bitonic.

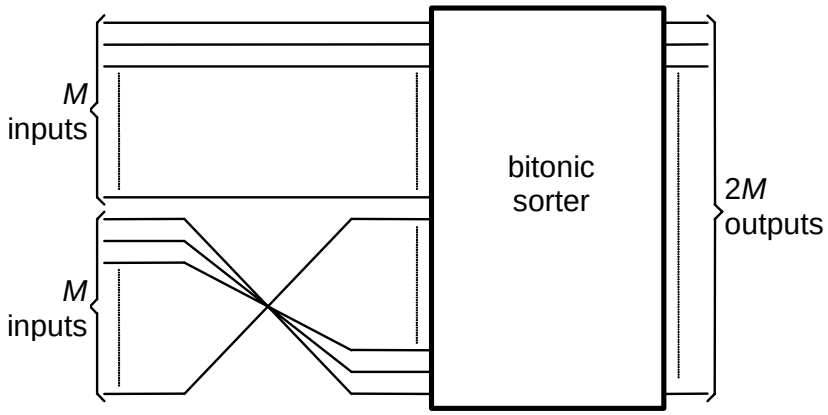

Figure 12: Using a bitonic sorter as a merge network.

Using the above theorem, a large bitonic sorter may be defined in terms of smaller bitonic sorters (Figure 13). The 2×2 sorters send $d_0,\ldots,d_{N-1}$ into the top subnetwork and $e_0,\ldots,e_{N-1}$ into the lower subnetwork. Since $\max(d_0,\ldots,d_{N-1}) \le \min(e_0,\ldots,e_{N-1})$, and the sequences are both bitonic, the subnetworks will sort them correctly.

An example of an 8 × 8 sorter made from bitonic sorters is shown in Figure 14. As shown in the diagram, it is made up of a stage of four 2 × 2 bitonic sorters, followed by two 4 × 4 bitonic sorters and finally a single 8 × 8 bitonic sorter, making up a complete Batcher sorting network. The 4 × 4 and 8 × 8 bitonic sorters are themselves made up from smaller bitonic sorters; this is also illustrated in the case of the 8 × 8. The connections which invert the lower sequence in Figure 12 have been eliminated by inverting the sorting direction of some 2 × 2 sorters; the arrows indicate the direction of sorting from lowest to highest. The interconnect between the 2 × 2 sorters of Figure 13 and the first stage of each subnetwork results in the concatenation of two interconnects. Figure 14 has been re-drawn to reflect this structure in Figure 15.

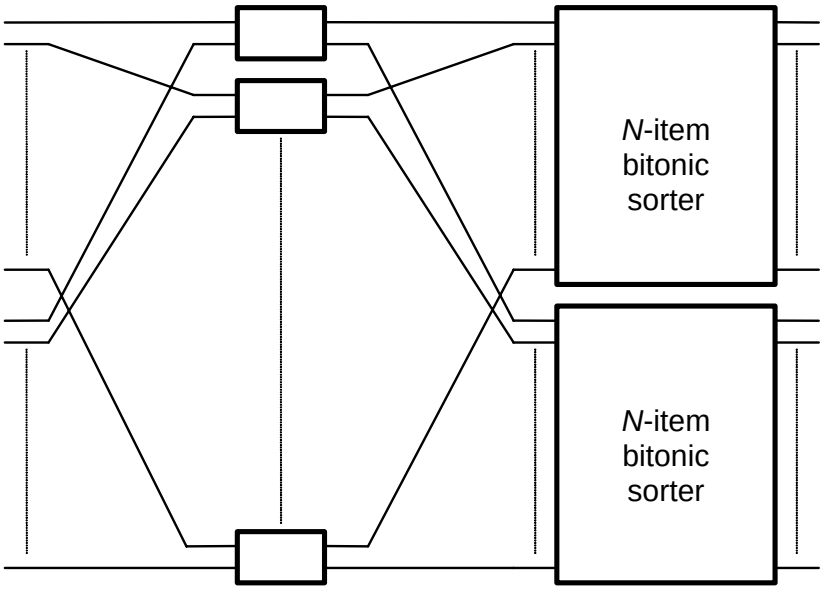

Figure 13: Recursive definition of a 2*N*-item bitonic sorting network. The 2 × 2 sorting elements send the lowest valued item of each pair to the upper network.

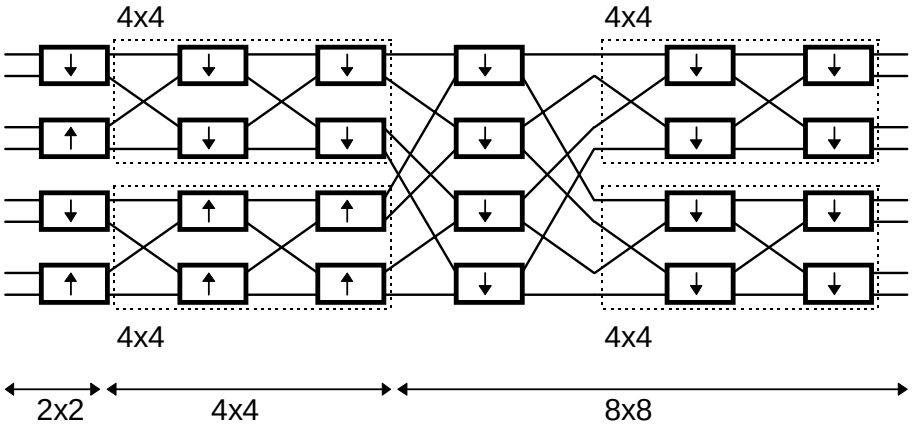

Figure 14: An 8 × 8 Batcher sorting network.

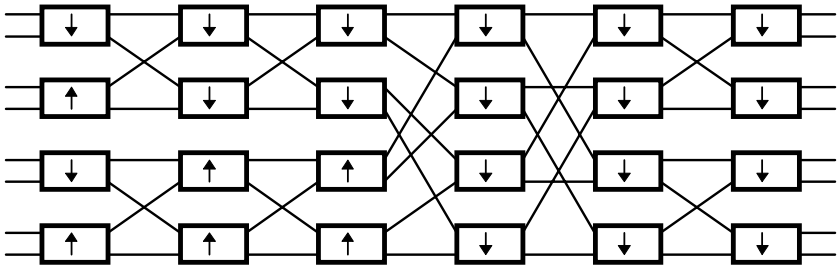

Figure 15: An 8 × 8 Batcher sorting network, with re-drawn interconnection pattern.

Each $N/2 \times N/2$ bitonic sorter has a total of N inputs and N outputs, and has $\log_2 N$ stages of $N/2$ switches, making $(N\log_2 N)/2$ switches in total. If the number of switches in such an N-input sorting network is $S(N)$, then:

$$S(N) = 2S\left(\frac{N}{2}\right) + \frac{N \log_2 N}{2}$$

Since $S(2) = 1$, the solution is:

$$S(N) = \frac{N}{4}\log_2 N\left(\log_2 N + 1\right)$$

There are hence $\log_2 N(\log_2 N + 1)/2$ stages.

2.8 Banyan networks

Unlike the switching networks considered earlier, *banyan networks* are blocking.

2.8.1 Binary permutations of connections

The interconnect field between adjacent stages in a switching network may be represented by a mapping of the outputs of the left-hand stage onto the inputs of the right-hand stage. For example, the interconnect in Figure 16 could be represented by the function $I(\ )$:

$$I(0) = 0$$
$$I(1) = 2$$
$$I(2) = 4$$
$$I(3) = 6$$
$$I(4) = 1$$
$$I(5) = 3$$
$$I(6) = 5$$
$$I(7) = 7$$

i.e. the pth output from the top on the left-hand stage is connected to the $I(p)$ th input on the right-hand stage. All connections are numbered starting from zero.

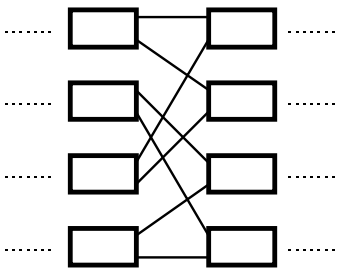

Figure 16: An interconnect between two stages.

It so happens that the interconnects in a banyan network (and many other networks besides) can be easily and compactly represented in terms of the binary representation of the connections. Suppose that p is the link number, as above. Let $p_2 p_1 p_0$ be its binary representation e.g. if $p = 3$ (i.e. 011 binary) then $p_2 = 0$ and $p_1 = p_0 = 1$. Then:

$$I(p_2 p_1 p_0) = p_1 p_0 p_2$$

It can easily be verified that this corresponds to the equations above. This is an instance of the so-called *perfect shuffle* permutation. On a k-bit binary number, this can be written in general as:

$$p_{k-1} p_{k-2} \cdots p_0 \rightarrow p_{k-2} \cdots p_0 p_{k-1}$$

This involves rotating the bits in p left, just like a "rotate left" instruction in a microprocessor.

2.8.2 Self-routing in banyan networks

Banyan networks have N inputs and outputs, and consist of $\log_2 N$ stages, each having $N/2$ 2×2 switches. Before the first stage, there is a perfect shuffle permutation, as described above, where $k = \log_2 N$. Between each adjacent pair of subsequent stages (stages i and $i + 1$, where $1 \le i \le k$), bit $k - i$ is exchanged with bit 0. An example, an 8×8 banyan network, is depicted in Figure 17.

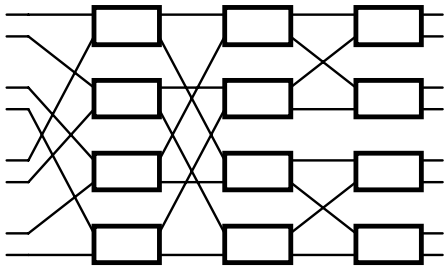

Figure 17: An 8 × 8 banyan network.

Suppose that a signal enters the network on some inlet number $s = s_{k-1} \cdots s_0$ and suppose also that all the switches in the network are in the bar-state (the top input is connected to the top output in each 2×2 switch, and the bottom input is connected to the bottom output.) After the first stage, the signal will be on link number $s_{k-2} \cdots s_0 s_{k-1}$ due to the preceding perfect shuffle. It will now be shown that after the ith stage, $i = 1,\ldots,k$, the signal is on link number $s_{k-1} \cdots s_{k-i+1} s_{k-i-1} \cdots s_0 s_{k-i}$; if $i = 1$, the digits $s_{k-1} \cdots s_{k-i+1}$ vanish.

The proof is by induction; it is certainly the case for $i = 1$. Assuming it is true for i, exchange bit $k - i$ with bit 0. The transformation is:

$$\begin{aligned} s_{k-1} \cdots s_{k-i+1} s_{k-i-1} \cdots s_0 s_{k-i} &\rightarrow s_{k-1} \cdots s_{k-i+1} s_{k-i} s_{k-i-2} \cdots s_0 s_{k-i-1} \\ &= s_{k-1} \cdots s_{k-(i+1)+1} s_{k-(i+1)-1} \cdots s_0 s_{k-(i+1)} \end{aligned}$$

Hence it is true for $i + 1$ also. When $i + 1 = k$, the digits in $s_{k-(i+1)-1}\cdots s_0$ vanish, so the output link on stage k is $s_{k-1}\cdots s_0$, i.e. the same numbered link it entered on.

A signal may be routed through the banyan network to the desired destination $d = d_{k-1}\ldots d_0$ as follows. When a switch in stage i is encountered ($i = 1,\ldots k$), the signal is diverted to the top output of the switch if $d_{k-i} = 0$ and sent to the bottom output if $d_{k-i} = 1$. For example, if the signal being routed through the switch were a packet, each switch could be individually set based upon the appropriate bit of the destination, which is contained in the packet header.

The ability to carry out routing in this distributed way is called *self-routing*, and results in the signal being routed to output d since at each stage i and the following interconnect, bit $k - i$ of the source s is replaced with bit $k - i$ of the destination. This is one of the most important properties of banyan networks, meaning that each switch can be locally controlled from the packet headers rather than requiring a centralized controller as in, say, a Beneš network.

2.8.3 Non-blocking conditions for banyan networks

Another important property of the banyan network is its ability to accept the output of a Batcher sorter, where the packets entering the banyan network are sorted in order of their destination and occupy contiguous input terminals starting from 0 (the top). To prove that the banyan can handle this situation, it is necessary and sufficient to consider two packets. Let s be the source and d be the destination of one packet, and let the source and destination of the other packet be s' and d' respectively. Without loss of generality, assume that $d' > d$ and $s' > s$.

Since both packets come from a Batcher sorter, $d' - d \geq s' - s$. It is necessary to prove that these two packets are always on different links, regardless of the value of i, i.e. regardless of which interconnect is being considered. From the above discussion the link number on stage i includes the following digits, although not in this order:

- $d_{k-1}\cdots d_{k-i}$ and $s_{k-i-1}\cdots s_0$ for one packet, or
- $d'_{k-1}\cdots d'_{k-i}$ and $s'_{k-i-1}\cdots s'_0$ for the other.

There are two cases to consider:

1. If one or more bits within $d_{k-1}\cdots d_{k-i}$ and $d'_{k-1}\cdots d'_{k-i}$ differ, then the two packets are on different links in stage i and the condition is satisfied.

2. Otherwise, $d' - d < 2^{k-i}$, meaning that the two destinations are less than 2^{k-i} links apart. Since $d' - d \geq s' - s$, $2^{k-i} > s' - s > 0$ i.e. their sources must also be less than 2^{k-i} links apart. Hence at least one bit in $s_{k-i-1}\cdots s_0$ must differ from those in $s'_{k-i-1}\cdots s'_0$, also implying the two packets are on different links.

Hence no two packets ever coincide in this scenario. This property (i.e. routing the output of a Batcher sorter) is very useful in designing ATM switches (Section 4).

3. Time switching

3.1 Time division multiplexing

Time division multiplexing (TDM) permits multiple voice or data channels to be carried along the same physical link. Until recently, it was the principle under which almost the entire telephone network operated, and an understanding of it is hence fundamental to appreciating the state of the network at present.

Information in the telecommunications network is almost universally transmitted in digital form; at regular intervals (8000 times a second) an analog voice signal is sampled and each sample is assigned a number corresponding to its amplitude at that point. These samples are transmitted as a stream of binary numbers. This procedure is known as Pulse Code Modulation (PCM), and although it was invented as long ago as 1938 by Reeves, it had to await the invention of microelectronics before it became widespread in the 1960s.

In TDM, samples of equal length are transmitted from each channel in turn, with the space occupied by each channel being known as a *timeslot*. After a sample has been transmitted from each channel, the sequence of timeslots starts again, consisting of the next sample from each channel (Figure 18). This sequence is called a *frame*, with the voice channel that a sample belongs to being denoted by its position within this frame.

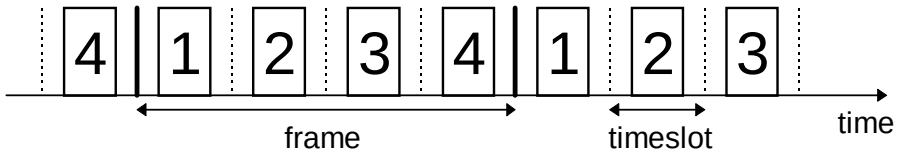

Figure 18: A time division multiplexed (TDM) frame; for clarity, gaps are shown between the information in consecutive timeslots, although this may not exist in practice.

In voice telephony, each circuit produces an 8-bit sample 8000 times a second, so the circuit bitrate is 64Kb/s. There are 8000 frames per second and each timeslot holds one 8-bit sample. In North America, a DS1 signal carries 24 such TDM circuits with a total bitrate of 1.544Mb/s, while in Europe, a CEPT1 signal carries 30 circuits, with a total capacity of 2.048Mb/s. In both cases, the frame structure is slightly more complex since there is also signaling and synchronization information included [9].

DS1 and CEPT1 signals can be transmitted over twisted pair cables, allowing multiple conversations to be transmitted between exchanges over one twisted pair line without the expense of installing more cables. They represent the lowest level of so-called "multiplexing hierarchies". For example, in North America, TDM signals exist from DS1 to DS4, each being created by multiplexing lower-level signals (Table 4). DS4 carries 4032 voice calls at an aggregate bitrate of 274Mb/s. These digital signals are transported by SONET/SDH paths, which is a more sophisticated form of TDM, forming a similar hierarchy; systems with line rates as high as 10Gb/s have been installed. There is also an earlier hierarchy, known as the Plesiochronous Digital

Hierarchy (PDH) which has largely been superseded by SONET/SDH in many parts of the world.

Digital Signal Number	Number of Voice Circuits	Bit Rate (Mb/s)
DS1	24	1.544
DS1C	48	3.152
DS2	96	6.312
DS3	672	44.736
DS4	4032	274.176

Table 4: North American and Japanese digital TDM signals.

3.2 Timeslot interchangers

A *timeslot interchanger* (TSI) accepts a TDM signal on its input, rearranges the order of the timeslots within each frame (under the control of a control unit) and outputs the new frames. Figure 19 shows frames of four timeslots undergoing timeslot interchange by having their timeslots put in a different order. Generally, TSIs work by writing each frame into a buffer memory and then reading the information out in a different order onto the output.

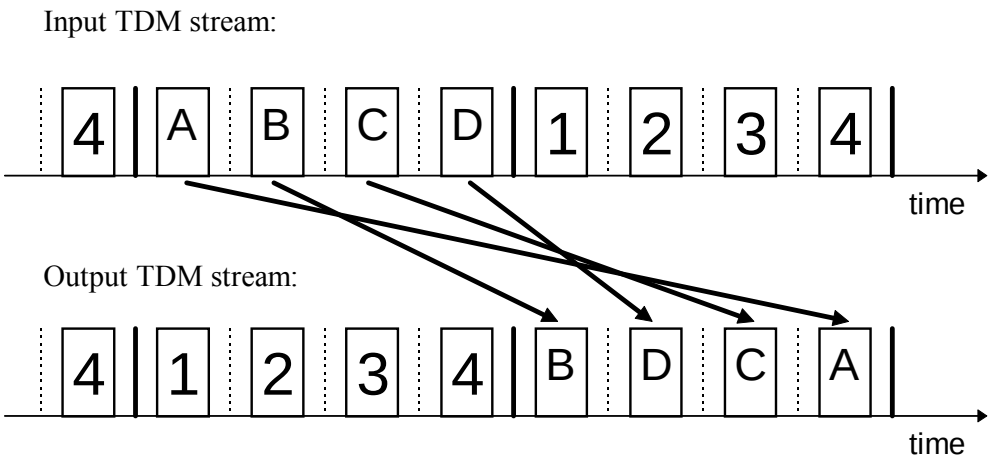

Figure 19: An example of timeslot interchange (TSI), shown for one frame in a sequence.

3.3 TST switches

A time-space-time (TST) switch (Figure 20) has r input lines and r output lines, each carrying n time-interleaved channels. There are hence a total of $N = rn$ channels, and any permutation of input channels onto output channels is possible. Each input stage TSI has m timeslots per frame on its output; the frame duration is the same throughout, so the bitrate is m/n times as great on the links to the space switch as it is on the inputs and outputs. Likewise, the inputs to the output stage TSIs have m timeslots per frame. The space switch reconfigures itself on every timeslot. Thus, one time-multiplexed space switch takes the place of the m space switches that would be required in a conventional Clos network, providing a reduction in the hardware requirements.

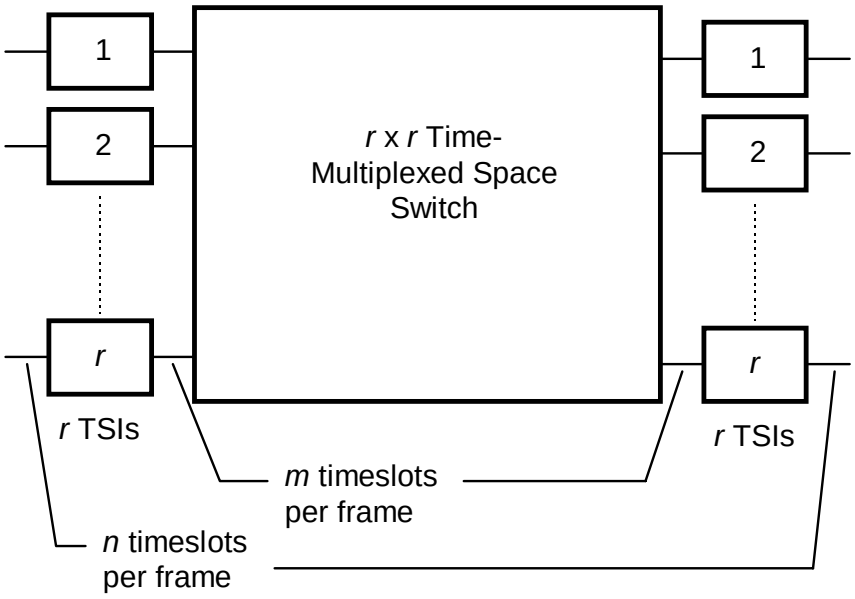

Figure 20: A time-space-time (TST) switch.

From a mathematical point of view, this is equivalent to the three-stage Clos network of Figure 5. Each TSI takes the place of an input or output stage switch, and on each new timeslot, the central space switch of Figure 20 mimics the operation of a different center stage switch in Figure 5. As before, the Lee and Jacobaeus methods (Section 2.3) may be applied to compute blocking probabilities. The Clos result applies: the TST switch is strict-sense nonblocking if $m \geq 2n - 1$, and it is rearrangeable if $m \geq n$.

Besides their use in circuit switches, TST architectures also carry out crossconnection of paths (represented by entities known as lower order virtual containers or LVCs) in SONET/SDH crossconnects [10].

4. ATM switches and crossconnects

Asynchronous transfer mode (ATM) is a key technique for deployment in the telecommunications network, due to its flexibility, efficient use of bandwidth and future-proof nature. It is essentially a form a packet switching where all traffic is divided into packets (or "cells") of 53 bytes, consisting of 5 bytes header and 48 bytes data (Figure 21). Time is partitioned into "timeslots", each timeslot consisting of the time taken to transmit one packet, usually at 155Mb/s or 622Mb/s.

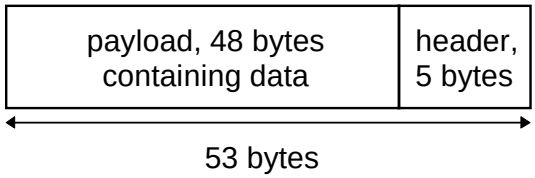

Figure 21: Format of an ATM cell, showing header and payload. The header contains the VCI and VPI fields.

One packet may be carried over any given link on each timeslot, but unlike TDM, there is no frame structure. A packet's channel is not identified by its position in a frame, but by its header. The header has many fields, but those of significance here are the *VCI* (virtual circuit indicator) and *VPI* (virtual path indicator), which indicate what channel a packet is on.

The VCI identifies a circuit, while the VPI identifies a path. In complete analogy with the discussion of the introduction, a path (VP) is a semi-permanent connection, carrying many user circuits (VCs). An ATM crossconnect carries out routing on paths using the VPI only while an ATM switch uses both VPI and VCI fields to switch circuits [11].

When a packet arrives at a crossconnect, the VPI in the header is examined and used to index a table, determining the crossconnect output for the packet and the new value of VPI which is to replace the old value in the header. This is known as *header translation*; the table is implemented with content-addressable memory. In a switch, both the VCI and the VPI are examined and replaced. In the remainder of this section, the term "switch" will be used loosely, referring to both switches and crossconnects. Besides header translation, other functions carried out by an ATM switch are:

- *switching*; this function simply ensures that each packet is directed to the correct output, and

- *buffering*; ATM is called *asynchronous* transfer mode because the arrival of packets at a switch's inputs is not coordinated; there is nothing to prevent two or more packets arriving at a switch's inputs on the same timeslot which need to go to the same output. To accommodate this, first in first out (FIFO) buffers in the switch hold up all but one of the contending packets. ATM switches are categorized below depending on where the buffers are placed relative to the switch itself.

A switch consists of many small *switch elements*, which might each fit on a circuit board, and are connected to form a *switch fabric*. This forms the switch *core*, surrounded by interface modules consisting of line cards, interline cards and feeder cards. The cost of the switch is dominated by software, and the remainder is dominated in cost by the interface modules. The switch core itself is a small but fundamental and highly critical part of the switch.

4.1 ATM switch element architectures

Throughout, when evaluating ATM switches, *uniform traffic* is assumed, where packets arriving at each input form a Bernoulli process i.e. there is a probability p of a packet arriving at a given input on any timeslot, and this is independent of what takes place on any other timeslot or input. A typical practical value is $p = 0.8$. Each packet has an equal probability of going to each output, again, this is independent of other inputs or timeslots.

While these assumptions have some validity when the ATM traffic consists of the summation of a large number of lower capacity streams, they are nonetheless idealistic and are usually regarded as merely a starting point leading to more detailed analysis and simulation. Other types of traffic include:

- *bursty* traffic where groups of packets tend to clump together in time, and there is a dependence between the probabilities of packets appearing on subsequent timeslots, and

- *non-uniform* traffic where the probabilities of a packet going to each output of a switch are not equal.

Furthermore, merely because the traffic going into a switch is a Bernoulli process does not imply that the traffic coming out will have the same statistics. Hence the analysis may not be valid in a real network where traffic must pass through several nodes. These topics are not considered further here but would be important in an advanced study of these switching architectures.

4.1.1 Input buffered switches

An input queueing switch consists of a space switch with a buffer on every input to prevent contention (Figure 22); they are controlled to prevent contention within the switch. The best known input buffered switch was proposed by Hui [12].

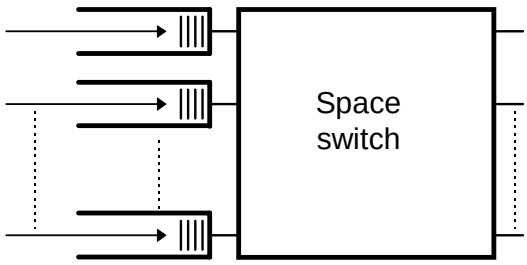

Figure 22: An input buffered switching element.

In this switch, a three-phase algorithm resolves contention. The space switch of Figure 22 consists of a Batcher sorting network (Section 2.7) followed by a banyan switch (Section 2.8); these sort the packets into order according to destination and then route them to the correct outputs. The phases of the algorithm are:

1. *Request phase.* Small packets consisting of each source-destination pair are sent through the sorting network. Adjacent requests with the same destination are then purged, leaving no more than one packet with each destination address.

2. *Acknowledgment phase.* An acknowledgment with destination is sent to each port which won the contention; this is routed through the entire sort-banyan network.

3. *Transmit phase.* Cells are transmitted by the successful ports.

The first two phases constitute an overhead, and the switch must be speeded up to compensate for this.

The throughput of this switch is limited to 58.6%, which can be demonstrated by some analysis; assume that the switch is saturated so that there is always at least one packet in each input buffer [13, 14]; every time a packet leaves an input buffer to go to an output, another packet takes its place at the head of the buffer. B_m^i is the number of packets destined for output i at the head of the input buffers during timeslot m but not sent because they were not selected; only one packet destined for a particular output can be sent in any timeslot. A_m^i is the number of packets destined for output i moving to the head of the input buffers in timeslot m. These quantities are related by:

$$B_m^i = \max\left(0, B_{m-1}^i + A_m^i - 1\right)$$

This is the same form as the equation for a single queue, where B_m^i is now the number of packets in queue i on timeslot m and A_m^i is now the number of packets arriving at the queue on timeslot m. In other words, each output i may be regarded as possessing a queue (called the "virtual queue"); the B_m^i packets at the heads of the input buffers destined for that output are regarded as waiting in this virtual queue and the A_m^i new packets destined for that output moving to the head of the input buffers may be regarded as the arrivals.

Now consider a particular virtual queue, number i. If F_{m-1} is the number of packets transferred through the switch on timeslot $m - 1$, then:

$$\Pr\left(A_m^i = k\right) = \binom{F_{m-1}}{k}\left(\frac{1}{N}\right)^k\left(1-\frac{1}{N}\right)^{F_{m-1}-k}$$

where $0 \le k \le F_{m-1}$ and

$$F_{m-1} = N - \sum_{i=1}^{N} B_{m-1}^i = \sum_{i=1}^{N} A_m^i$$

The transfer rate per output is $\rho_0 = \overline{F}/N$. As $N \to \infty$, A_m^i acquires a Poisson distribution with rate ρ_0; the proof of this is quite complex and can be found in an Appendix in Reference [13]. Hence the virtual queue for each output can be regarded as an M/D/1 system, with average waiting time given by:

$$\overline{B}^i = \frac{\rho_0^2}{2(1-\rho_0)}$$

Using the equation above,

$$\overline{B}^i = \frac{\sum_{i=1}^{N} \overline{B}^i}{N} = \frac{N - \overline{F}}{N} = 1 - \rho_0$$

hence, combining the last two equations, it can be shown that under conditions of saturation the throughput of the switch is $\rho_0 = 2 - \sqrt{2} = 58.6\%$. This is for very large *N*; for smaller sizes of switch the throughput is greater (Table 5) although it rapidly converges to 58.6% as the size of switch increases. Hence a major limitation of input-buffered switches is their low throughput, caused by a phenomenon known as head-of-line blocking. This occurs when the packet at the head of an input queue cannot reach its desired output because another packet from another input is being sent there. Meanwhile, there may be one or more packets behind it in the input queue which cannot reach their free outputs because this packet is in the way. To overcome this problem, it is possible to retrieve packets from behind the head of the input queues i.e. not operate in a FIFO (first in first out) manner, yielding higher throughput [15], as large as 0.9, but at the expense of greater complexity.

N, size of switch	saturation throughput
1	1.0000
2	0.7500
3	0.6825
4	0.6553
5	0.6399
6	0.6302
7	0.6234
8	0.6184
∞	0.5858

Table 5: Maximum throughput achievable using input queueing with FIFO buffers.

4.1.2 Output buffered switches

Output buffered architectures have buffers on their outputs (Figure 23). If the buffer size were infinite, they would have optimal delay/throughput performance as there is nothing (such as head-of-line blocking) to impede the flow of packets through the switch, except for the necessary delay incurred in waiting for the correct output to become available. In the diagram, multiple packets from different inputs may be placed in a buffer on the same timeslot. To achieve this, either the switch can be speeded up by a factor of *N*, or *N* lines can feed into each output buffer. There are many variations on this theme, but a typical example can be found in Reference [16].

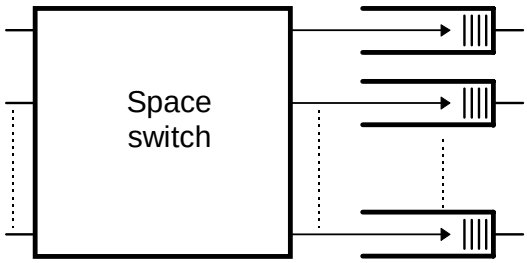

Figure 23: An output buffered switching element.

To model this architecture analytically [13,14], consider one particular output queue. A_m is the number of packets arriving in the queue on timeslot *m* and Q_m is the length of the queue. If *b* is the maximum capacity of the queue, these are related by:

$$Q_m = \min\left(\max\left(0, Q_{m-1} + A_m - 1\right), b\right)$$

If Q_{m-1} and A_m are both zero, no packet leaves the queue and $Q_{m-1} + A_m - 1 = -1$, hence the need for "max". The arrival process is described by a binomial distribution:

$$\Pr\left(A_m = k\right) = a_k = \binom{N}{k}\left(\frac{p}{N}\right)^k\left(1 - \frac{p}{N}\right)^{N-k}$$

Where $q_n = \Pr\left(Q_m = n\right)$, the probability that *n* packets are in the queue; this can be solved using a Markov chain [14] to yield:

$$q_1 = q_0 \frac{1 - a_0 - a_1}{a_0}$$
$$q_n = \frac{1 - a_1}{a_0} q_{n-1} - \sum_{k=2}^{n} \frac{a_k}{a_0} q_{n-k}$$

where $n \geq 2$. To generate a solution, q_0 is set to some arbitrary value (e.g. 1) and after computing $q_1, \ldots, q_b$, the results are normalized so that $\sum_{k=0}^{b} q_k = 1$. The throughput is $\rho_0 = 1 - q_0 a_0$ since a packet always leaves the queue except when the queue is empty and no packet arrives. The packet loss rate is $L = 1 - \rho_0 / p$.

With finite buffers, packets must be thrown away (or lost) when they arrive at a buffer which is full. For acceptable performance, the probability of this happening must be very small; the exact value depends on the application. Figure 24 shows the packet loss probability for load $p = 0.8$ as a function of the buffer size and the switch size. It can be seen that to obtain an acceptable packet loss probability (10^{-10} or less) for any but the smallest switch, a buffer depth of at least 50 is required. Bear in mind that this analysis is for uniform, non-bursty traffic; otherwise, larger buffers are required for an acceptable cell loss.

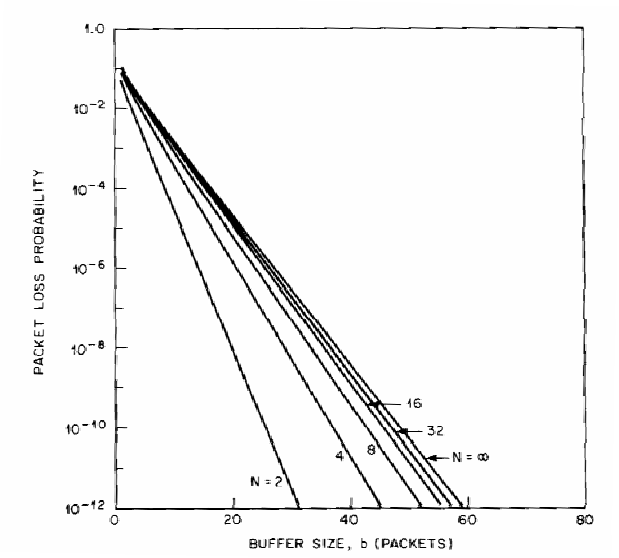

Figure 24: Packet loss probability for output buffering as a function of the buffer size *b* and the switch size *N*, for an offered load of *p* = 0.8. © 1988 IEEE, reproduced with permission [14].

As $N, b \to \infty$, the mean waiting time can be shown to be $p/[2(1-p)]$, the same as for an M/D/1 queue [13]. Figure 25 shows the loss as a function of buffer size for different loads; the load has a significant effect on the cell loss. In Figure 26, the mean waiting time is plotted; for small buffer depths more packets are lost but those that are successful suffer a smaller mean delay.

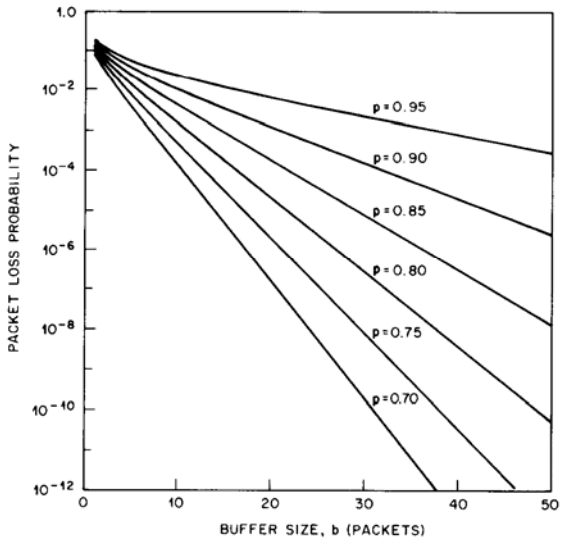

Figure 25: Packet loss probability for output buffering as a function of the buffer size *b* and offered loads varying from *p* = 0.70 to *p* = 0.95, for the limiting case of *N* = ∞. © 1988 IEEE, reproduced with permission [14].

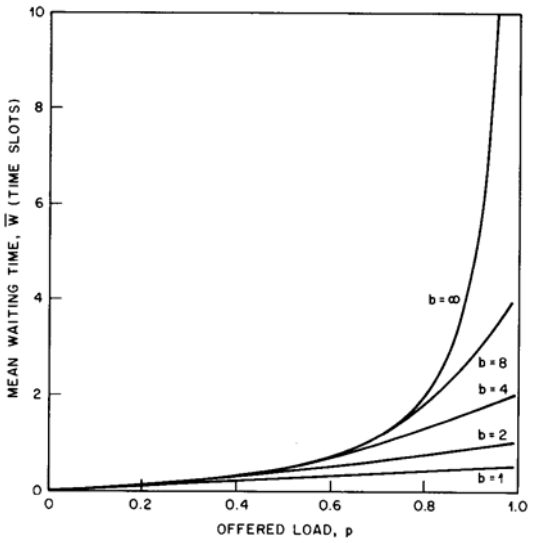

Figure 26: Mean waiting time for output buffering as a function of the offered load p, for $N = \infty$ and output buffer sizes varying from $b = 1$ to $b = \infty$. © 1988 IEEE, reproduced with permission [14].

4.1.3 Shared buffer switches

In a shared buffer memory switch, incoming packets are read into a central memory area, and read out when required. One queue is stored in the central memory for each output, and each of these queues may grow indefinitely, providing the sum of the queue sizes does not exceed the total memory size. This "sharing" mechanism implies the major advantage of shared buffer memory; for the same total amount of memory, the performance is superior to output buffering, particularly for bursty and nonuniform loads. The major disadvantage lies in the complexity of the necessary memory management.

With this central memory area, every input must be written and every output must be read at most once a timeslot, hence the requirements on the memory cycle T_m are:

$$T_m < \frac{P_B \times P_C}{R_B \times 2 \times N}$$

- P_B is the number of parallel bits entering or leaving each data highway,
- P_C is the number of memory chips in parallel,
- R_B is the bitrate (155Mb/s, 622Mb/s or higher), and
- N is the size of the switching element.

Hence the major drawback of this architecture is the increase in memory speed required as the size of the switch increases, meaning that it does not scale well to large sizes.

When modeling this architecture [14], the packet loss is approximated for $N \geq 16$ by the probability that the sum of the queue lengths exceeds the total buffer capacity Nb:

$$\Pr\left[\sum_{i=1}^{N} Q_i \geq Nb\right]$$

This is solved by considering a convolution of N M/D/1 queues (Figure 27).

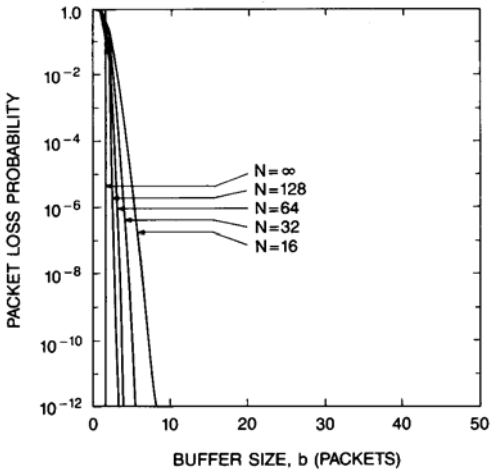

Figure 27: Packet loss probability for completely shared buffering as a function of the buffer size per output *b* and the switch size *N*, for offered load *p* = 0.8. © 1988 IEEE, reproduced with permission [14].

4.2 ATM switch fabrics

Recall that a switch fabric is made from many smaller switch elements (such as those discussed above); one way of implementing this involves using a banyan network configuration composed of buffered switching elements instead of space switches. This was extensively studied in the early days of ATM switch research, and has the advantage of being self-routing. However, this approach suffers from two major problems:

- *The performance degrades seriously under non-uniform traffic patterns* [17]. Consider the banyan network of Figure 17, and suppose that all the traffic generated by the even-numbered inputs (or the odd-numbered inputs) wishes to travel to the top four outputs. One can easily verify that all the traffic goes through one switch in the center stage, hence the throughput of all these inputs and outputs is restricted to the throughput of this single switch.

- *It is not tolerant to faults*. Referring to the same type of scenario, it is clear that large numbers of input-output pairs can be isolated by the failure of just one switch.

Furthermore, these problems become worse for larger networks, so for these reasons, the three-stage Clos architecture [18] is favored in most recent designs.

In Figure 5, there are clearly m different routes through the network between each input and output pair. Either each packet can be sent on a random route, or all packets in a virtual circuit can be routed the same way. The latter option is usually chosen, since it allows load balancing and preserves packet order.

By extending the Clos result (Section 2.2), it can be shown that a three-stage Clos network carrying ATM packets is strict-sense nonblocking to new calls if

$$m > 2 \max_{b \le \omega \le B} \left\{ \lim_{\varepsilon \to 0} \left\lfloor \frac{\beta n - \omega}{\max(b, 1 - \omega + \varepsilon)} \right\rfloor \right\}$$

where β is the maximum traffic level on any input/output link, b is the minimum traffic per circuit, and B is the maximum traffic per circuit [18]. The "traffic level" of a circuit is defined as the probability that the link will contain a packet from that circuit. Hence $0 \le b \le B \le \beta \le 1$. For the total traffic on a link, $\beta = 0.8$ is a possible practical value. For circuit switching, $b = B = \beta = 1$, and one can verify that the above result reduces to the Clos result, $m \ge 2n - 1$, for this case.

5. Optical WDM crossconnects

5.1 Introduction

Wavelength division multiplexing (WDM) involves multiplexing several SONET/SDH and/or PDH signals onto one fiber, each signal being at a different optical wavelength. This allows fiber ducts (typically containing 96 fibers) to transport more traffic even once they are fully utilized by single-wavelength systems. It also will allow future high-bandwidth services (such as high-speed data transfer, videoconferencing, videophone and high-speed Internet) to be provided at a reasonable cost. Point-to-point systems offering over 320 Gb/s capacity (32 wavelengths each at 10 Gb/s) are commercially available and laboratory experiments have been reported operating at over 3 Tb/s (3000 Gb/s) on a single fiber. Here, switching systems for WDM are discussed, consisting of *optical crossconnects* (**OXCs**) to be used in the top layer of the network.

The components that make up these crossconnects could be described in terms of their physical principles of operation [19], however it is appropriate here merely to state what their functions are. Of course, these devices have certain imperfections and shortcomings (such as noise, crosstalk and attenuation) which impact on the performance of the system with respect to optical loss and bit error rate. This is a subject in itself which will not be discussed here. Throughout, M is the number of wavelengths, and each device input or output is carried on a distinct optical fiber.

- A *WDM multiplexer* (mux) has M inputs and one output. The first input is always at wavelength 1, the second at wavelength 2, and so on. The output is an aggregate of all these signals.

- A *WDM demultiplexer* (demux) has one input and M outputs. It performs the converse operation to a multiplexer, taking a single input carrying multiple wavelengths and splitting it into its M components.

- An *optical combiner* combines multiple input signals into one output signal. Unlike the WDM multiplexer, each input may be on any wavelength; there is no stipulation that the input signal wavelengths must be in order. This component has the disadvantage when compared to the WDM multiplexer of having greater optical power loss.

- An *optical space switch* is an optical implementation of a switching network such as those in Section 2. Crossbar switches, Beneš networks and other architectures such as tree switches [20] have been proposed. Technologies include lithium niobate directional couplers and indium phosphide semiconductor laser amplifiers (SLAs).

- A *regenerator* converts an optical signal into electronic form then amplifies, re-shapes and re-times the binary pulses. It then converts the signal back to optical form.

- A *wavelength converter* is similar to a regenerator, but it re-transmits the signal at a different wavelength from that which it was received on.

Optical crossconnects will be a crucial part of future optical WDM mesh networks, forming the basis of the network nodes and allowing long-term connections to be provisioned. Each crossconnect node has an equal number of inputs and outputs, with two scenarios:

- In a network without wavelength conversion (called a WP or Wavelength Path network), a path is on the same wavelength throughout and there is no wavelength conversion within the crossconnects. This has the disadvantage of requiring rather more wavelengths in the network since the paths cannot necessarily be "packed" so efficiently into the available capacity. However, lack of wavelength conversion has the advantage of reducing the complexity and cost of each node. In such a network, a path entering a crossconnect on some input at some wavelength may leave on any output on the same wavelength, with the condition that multiple paths cannot leave a given link on the same wavelength.
- In a network with wavelength conversion (called a VWP or Virtual Wavelength Path network), each link on a given path may carry the path on a different wavelength, and to facilitate this, there must be wavelength converters within the crossconnects. This requires more hardware and increases cost, but makes the use of capacity more efficient. In such a network, a path entering a crossconnect on some input at some wavelength may leave on any output on any wavelength, with the condition that multiple paths cannot leave a given link on the same wavelength.

Naturally, a VWP OXC may be substituted for a WP OXC by disabling wavelength conversion although this is not necessarily efficient in terms of hardware usage and optical power loss. This section summarizes various optical crossconnect (OXC) nodes which can be adopted to both WP and VWP networks. These can be assessed based on several criteria:

- WP or VWP,
- cost,
- optical performance, especially loss,
- modularity – the ability to build the OXC from many identical units,

- upgradability – the ability to upgrade from a small OXC to a larger one gracefully and efficiently, and
- scalability – the ability to scale to large sizes while still retaining reasonably efficient use of hardware.

It is not anticipated that OXCs will actually be deployed until approximately 2002, although prototypes do exist in research laboratories. Throughout, M denotes the number of wavelengths handled by an OXC and N denotes the number of inputs and outputs.

5.2 OXCs without wavelength conversion (WP)

Figure 28 [21,22, 23] is a WP OXC – demultiplexers direct each wavelength to a different space switch, there being one for each wavelength (i.e. M space switches). Multiplexers merge the space switch outputs to produce the final OXC outputs. The OR/OS (optical receiver/optical sender) modules are optoelectronic regenerators which amplify the signal, and may also re-shape and re-time the pulses. Later, with VWP OXCs, they will be used as wavelength converters simply by having each transmitter at a different wavelength from the corresponding optical input signal. There is insufficient similarity between these architectures and VWP OXCs to allow upgrades to be made of WP OXCs, and they do not possess modularity. The cost of this architecture increases proportionally with the number of multiplexed wavelength channels because each space switch only switches one wavelength.

5.3 OXCs with wavelength conversion (VWP)

5.3.1 OXCs based on Clos networks

Figure 29 shows an OXC design based on a Clos network. The input signals are first demultiplexed before being fed into an $MN \times MN$ Clos network via wavelength converters which convert each path to the desired output wavelength. Each input fiber is paired with one $M \times (2M - 1)$ first stage switch. Each output stage switch is replaced by a combiner, since there would be a combiner at the output of each output switch in any case, making it redundant. Each path is directed by the Clos network to the optical combiner representing the output switch. Component loss in this VWP OXC is greater than in the WP OXC above, and its modularity is poor.

5.3.2 The delivery and coupling switch

An OXC based on "delivery and coupling" switches was proposed by NTT [24]. These are essentially SLA crossbar switches which can combine many inputs on different wavelengths and send them to one output. The feasibility of 8×16 "delivery and coupling" switch boards for providing 320 Gb/s total throughput has been confirmed [22]. This architecture is claimed to have superior modularity and upgradability characteristics as shown in Figure 30. Each input fiber is demultiplexed and wavelength converted, then the SLA crossbar switches send these signals to the coupler for the appropriate output. It is strictly non-blocking because any of the incoming optical signals can be connected to any of the outgoing ports without disturbing any established paths. The component loss associated with this architecture is also relatively low [25].

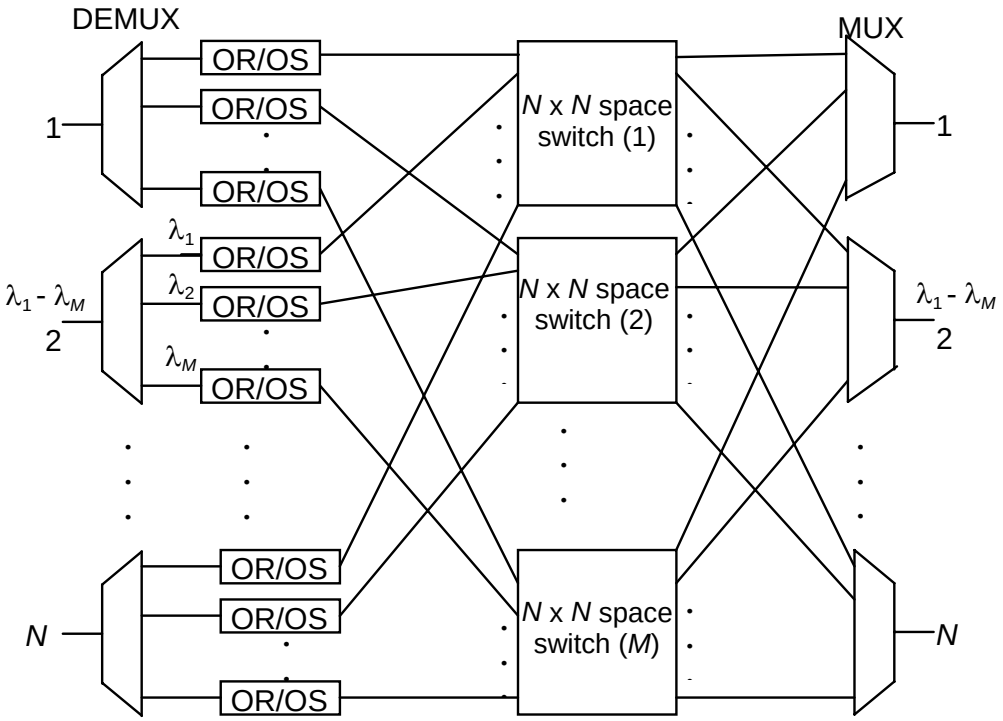

Figure 28: An optical crossconnect architecture incorporating optoelectronic regeneration with OR/OS modules.

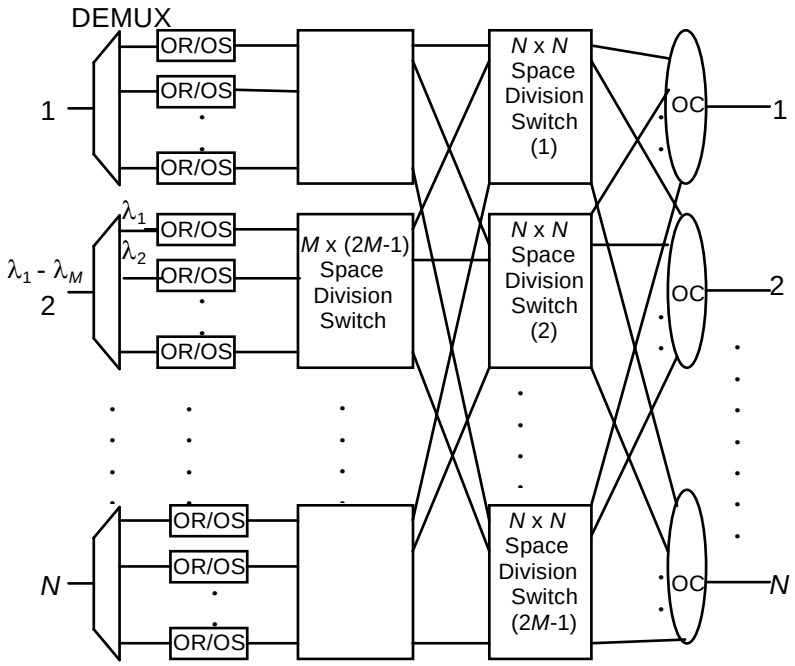

Figure 29: A VWP OXC based on a Clos network. OC = optical combiner.

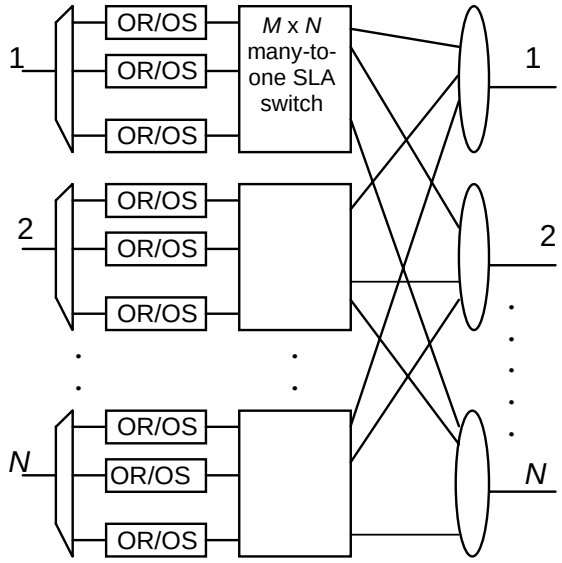

Figure 30: An OXC based on semiconductor laser amplifier (SLA) crossbar switches.

The originators of this OXC consider it most suitable for both VWP and WP transport network applications with respect to:

- non-blocking characteristics,
- optical loss,
- modularity,
- upgradability, and
- practicability.

6. Conclusions

Telecommunications is at present crucial to our society, and with the continuation of the information revolution, will become even more so. In turn, switching systems are a fundamental part of telecommunications networks, and their robustness and reliability is vital to our ability to depend on telecommunications. These switching systems have been studied for over 50 years and many of the papers that have been published are theoretical in nature, providing insights into the fundamental principles of their operation.

This article started by providing an overview of the most important of these theoretical results, covering Clos networks in various guises, and theorems which characterise the hardware performance for various levels of blocking and nonblocking operation. These Clos architectures form the basis of many switch architectures that were used both now and in the past. Batcher and banyan networks, which together can form an ATM switch, were also discussed and their key results were proved.

Time division multiplexing (TDM) and time switching are absolutely fundamental to developing an understanding of the telecommunications network as it is today. In explaining time switching, it was shown that, from a mathematical point of view, TST

networks are equivalent to Clos networks and hence the same theorems about blocking and nonblocking operation may be applied. TST networks have the advantage of reducing the amount of hardware required.

Two newer technologies were discussed in the remainder of the article. One of these, ATM switches and crossconnects, currently represents the state of the art, while optical crossconnects are likely to be introduced by around 2002. The principal types of ATM switches were introduced – input buffered, output buffered and shared buffered. Input buffered switches do not permit a high throughput while the other two exhibit optimal delay-throughput performance, with shared buffered switches giving superior performance for the same total amount of buffer memory. Finally, some of the principal types of optical crossconnect were introduced; these will permit future high-bandwidth services to be carried at a reasonable cost.

Further in the future it is possible that optical packet switches will be introduced, using high-speed optical switching and optical delay-line memories. These will overcome the problems associated with very high capacity electronic switch cores, such as crosstalk due to EMI and the difficulties introduced by having to interconnect many very high-speed connections. Optical packet switching is a topic in its own right, which is outside the scope of this article.

7. Acknowledgments

The author wishes to thank the EPSRC for support in the form of an Advanced Fellowship. He also wishes to thank Prof. Ivan Andonovic of the University of Strathclyde, and Dr. Tim H. Gilfedder of British Telecom for their careful and helpful reviews of drafts of this article.

8. Appendix: Proof of bitonic sorter iterative rule

Suppose that the sequence $a_0, a_1, \ldots, a_{2N-1}$ is bitonic. Define

$$d_i = \min\left(a_i, a_{N+i}\right)$$
$$e_i = \max\left(a_i, a_{N+i}\right)$$

for $i \in \{0, \ldots, N-1\}$. Then it is necessary to show that $d_0, \ldots, d_{N-1}$ and $e_0, \ldots, e_{N-1}$ are each bitonic, and:

$$\max\left(d_0, \ldots, d_{N-1}\right) \le \min\left(e_0, \ldots, e_{N-1}\right) \text{ [8].}$$

To prove this, suppose that $a_0, \ldots, a_{2N-1}$ is modified by taking sequence members from the right of the list and moving them so the left so that there is a j with $a_0 \le a_1 \le \cdots \le a_j \ge \cdots \ge a_{2N-1}$ (see Section 2.7). Then $d_0, \ldots, d_{N-1}$ and $e_0, \ldots, e_{N-1}$ will undergo a similar modification, which will not affect their bitonic property. Hence it is sufficient for the proof to consider only sequences where $a_0 \le a_1 \le \cdots \le a_j \ge \cdots \ge a_{2N-1}$. Furthermore, if the sequences are reversed, the bitonic property and their maxima and minima are not affected, so it is sufficient to consider $j \in \{N, \ldots, 2N-1\}$.

First, consider the case where $a_{N-1} \le a_{2N-1}$. It follows that for $i \in \{0,\ldots,N-1\}$, $a_i \le a_{N+i}$, so $d_i = a_i$ and $e_i = a_{N+i}$ so the above theorem holds. This is illustrated in Figure 31.

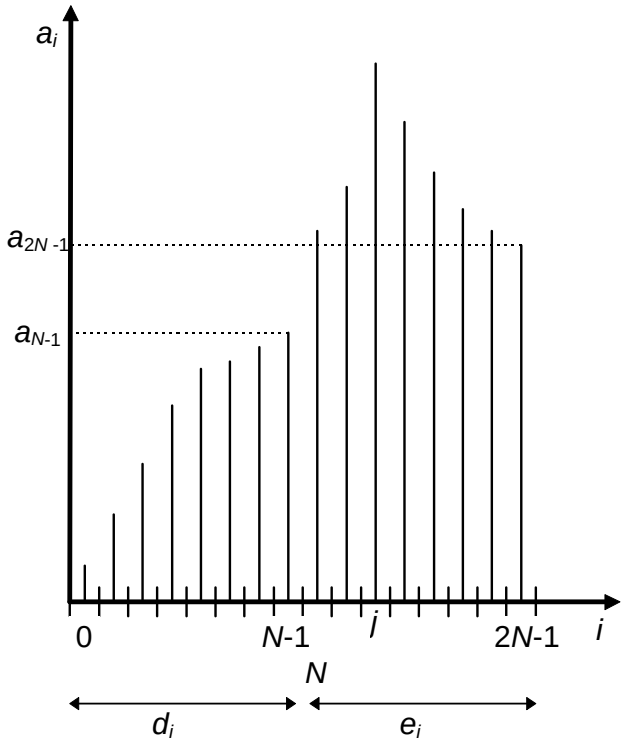

Figure 31: Illustration of a bitonic sequence for the case of $a_i \le a_{N+i}$.

Now consider $a_{N-1} > a_{2N-1}$ as shown in Figure 32; one can see that $a_j, a_{j+1}, \ldots, a_{2N-1}$ is decreasing while $a_{j-N}, a_{j+1-N}, \ldots, a_N$ is increasing. Hence a $k \in \{j,\ldots,2N-2\}$ can be found such that $a_{k-N} \le a_k$ and $a_{k-N+1} > a_{k+1}$. k is shown in Figure 32 for the particular case depicted. For $i \in \{0,\ldots,k-N\}$, $d_i = a_i$ and $e_i = a_{i+N}$. For $i \in \{k-N+1,\ldots,N-1\}$, $d_i = a_{i+N}$ and $e_i = a_i$. With the aid of Figure 32, one can verify that the following inequalities hold:

$$
\begin{aligned}
&d_i \le d_{i+1} \qquad i \in \{0,\ldots,k-N-1\} \\
&d_i \ge d_{i+1} \quad i \in \{k-N+1,\ldots,N-2\}
\end{aligned}
$$

$$
\begin{aligned}
&e_i \le e_{i+1} \quad i \in \{k-N+1,\ldots,N-2\} \\
&e_{N-1} \le e_0 \\
&e_i \le e_{i+1} \qquad i \in \{0,\ldots,j-N-1\} \\
&e_i \ge e_{i+1} \qquad i \in j-N,\ldots,k-N-1
\end{aligned}
$$

These prove that $d_0,\ldots,d_{N-1}$ and $e_0,\ldots e_{N-1}$ are bitonic and that

$$\max\left(d_0,\ldots,d_{N-1}\right)=\max\left(a_{k-N},a_{k+1}\right)\le\min\left(a_{k-N+1},a_k\right)=\min\left(e_0,\ldots,e_{N-1}\right)$$

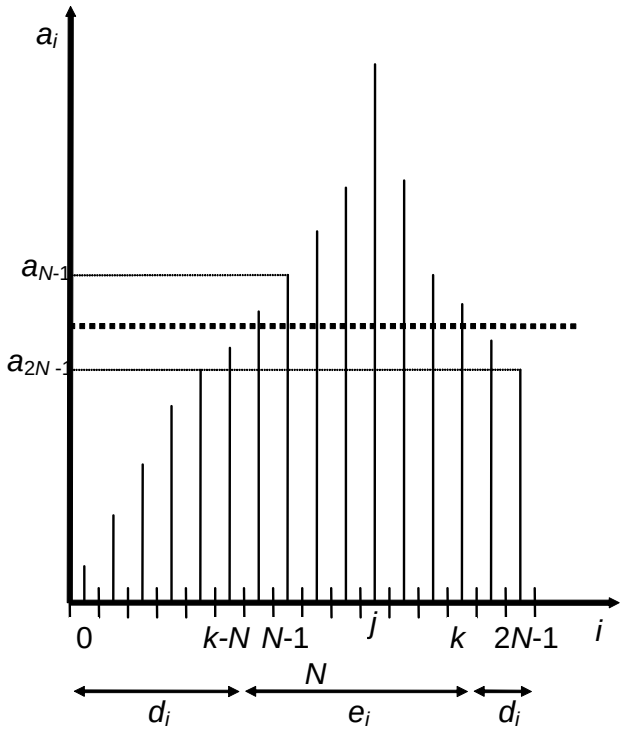

Figure 32: Diagram of a_i for the case of $a_{N-1}>a_{2N-1}$; the thick dotted line represents the division between d_i values and e_i values.

9. References

1. C. Clos: "A Study of Non-Blocking Switching Networks", *Bell System Technical Journal*, vol. 32, 1953, pp406-424.

2. J. Y. Hui: *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Publishers, 1990.

3. V. E. Beneš: *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, 1965.

4. P. Hall: "On Representatives of Subsets", *Journal of the London Mathematical Society*, vol. 10, 1935, pp26-30.

5. A. Waksman: "A Permutation Network", *Journal of the Association for Computing Machinery*, vol. 15, no. 1, January 1968, pp159-163.

6. D. C. Opferman, N. T. Tsao-Wu: "On a Class of Rearrangeable Switching Networks. Part 1: Control Algorithm", *Bell System Technical Journal*, vol. 50, no. 5, May-June 1971, pp1579-1600.

7. G. F. Lev, N. Pippenger, L. G. Valiant: "A Fast Parallel Algorithm for Routing in Permutation Networks", *IEEE Transactions on Computers*, vol. 30, no. 2, February 1981, pp93-100.

8. K. E. Batcher: "Sorting Networks and their Applications", *Proceedings of AFIPS 1968 Spring Joint Computer Conference*, 1968, pp307-314.

9. J. Bellamy: *Digital Telephony*, Second Edition, Wiley Interscience, 1991.

10. M. Sexton, A. Reid: *Broadband Networking*, Artech House, Boston, 1997.

11. R. O. Onvural: *Asynchronous Transfer Mode Networks: Performance Issues*, Artech House, Boston, 1994.

12. J. Hui, E. Arthurs: "A Broadband Packet Switch for Integrated Transport", *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 8, October 1987, pp1264-1273.

13. M. J. Karol, M. G. Hluchyj: "Input Versus Output Queueing on a Space-Division Packet Switch", *IEEE Transactions on Communications*, vol. 35, no. 12, December 1987, pp1347-1356.

14. M. G. Hluchyj, M. J. Karol: "Queueing in High-Performance Packet Switching", *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, 1988, pp1587-1597.

15. N. Arakawa, A. Noiri, H. Inoue: "ATM Switch for Multi-Media Switching System", *13th International Switching Symposium*, Stockholm, Sweden, May 27-June 1, 1990, paper A7#2, vol. V, pp9-14.

16. P. Barri, J. A. O. Goubert: "Implementation of a 16 to 16 Switching Element for ATM Exchanges", *GLOBECOM '90*, San Diego, California, paper 805.6, pp1615-1627.

17. H. S. Kim, A. Leon-Garcia: "Performance of Buffered Banyan Networks Under Nonuniform Traffic Patterns", *INFOCOM '88*, New Orleans, 27-31 March, paper 4A.4, pp344-353.

18. P. Coppo, M. D'Ambrosio, R. Melen: "Optimal Cost/Performance Design of ATM Switches", *IEEE/ACM Transactions on Networking*, vol. 1, no. 5, October 1993, pp566-575.

19. D. J. G. Mestdagh: *Fundamentals of Multiaccess Optical Fiber Networks*, Artech House, Boston, 1995.

20. D. K. Hunter, I. Andonovic: "Guided Wave Optical Switch Architectures. Part I. Space Switching", *International Journal of Optoelectronics*, vol. 9, no. 6, 1994, pp477-487.

21. A. Jourdan, F. Masetti, M. Garnot, G. Soulage, M. Sotom: "Design and Implementation of A Fully Reconfigurable All-optical Crossconnect For High

Capacity Multiwavelength Transport Network", *IEEE Journal of Lightwave Technology, June 1996*, vol.14, no. 6, pp1198-1206.

22. A. Watanabe, S. Okamoto, M. Koga, K.-I. Sato, M. Okuno: "Packaging of 8×16 Delivery and Coupling Switch For A 320Gb/s Throughput Optical Path Cross Connect System", *ECOC'96*, Oslo, pp111-114.

23. S. Okamoto, A. Watanabe, K.-I. Sato: "Optical Crossconnect Node Architecture For Photonic Transport Network", *IEEE Journal of Lightwave Technology*, June 1996, vol. 14, no. 6, pp1410-1422.

24. S. Okamoto, A. Watanabe, K. Sato: "A New Optical Path Crossconnect System Architecture Using Delivery and Coupling Matrix Switch", *Trans. IEICE Japan*, Oct. 1994, vol. E77-B, no. 10, pp1272-1274.

25. M. Koga, Y. Hamazumi, A. Watanabe, S. Okamoto, H. Obara, K. I. Sato, M. Okuno, S. Suzuki: "Design and Performance of an Optical Path Crossconnect System Based On Wavelength Path Concept", *IEEE Journal of Lightwave Technology*, June 1996, vol. 14, no. 6, pp1106-1119.