Report Number: WUCS-91-47

1991-10-09

A Quantitative Comparison of Architectures for ATM Switching Systems

Ellen E. White

There have been a number of ATM switching system architectures proposed, but little in the way of comparison to indicate which architectures are preferable from a cost standpoint given specific performance requirements. This paper considers a range of performance requirements and compares various architectures base don the number of pin-limited chips needed to realize a system which can meet the requirements. Our results indicate that certain architectures, for example the Knockout network, are not competitive within the range we considered. Other architectures perform reasonably well in some cases, but less well in others. The buffered Beness network with shared... Read complete abstract on page 2.

A Quantitative Comparison of Architectures for ATM Switching Systems

Ellen E. White

Complete Abstract:

There have been a number of ATM switching system architectures proposed, but little in the way of comparison to indicate which architectures are preferable from a cost standpoint given specific performance requirements. This paper considers a range of performance requirements and compares various architectures base don the number of pin-limited chips needed to realize a system which can meet the requirements. Our results indicate that certain architectures, for example the Knockout network, are not competitive within the range we considered. Other architectures perform reasonably well in some cases, but less well in others. The buffered Beness network with shared buffering at each switch element consistently has the lowest chip count over a range of network sizes and performance requirements.

A Quantitative Comparison of Architectures for ATM Switching Systems

Ellen E. Witte

WUCS-91-47

October 9, 1991

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

Abstract

There have been a number of ATM switching system architectures proposed, but little in the way of comparison to indicate which architectures are preferable from a cost standpoint given specific performance requirements. This paper considers a range of performance requirements and compares various architectures based on the number of pin-limited chips needed to realize a system which can meet the requirements. Our results indicate that certain architectures, for example the Knockout network, are not competitive within the range we considered. Other architectures perform reasonably well in some cases, but less well in others. The buffered Beneš network with shared buffering at each switch element consistently has the lowest chip count over a range of network sizes and performance requirements.

Revised on January 9, 1992.

A Quantitative Comparison of Architectures for ATM Switching Systems

Ellen E. Witte

1. Introduction

There have been a number of architectures for ATM switching systems proposed in the literature [5, 6, 7, 8, 9, 12, 13, 14, 15] with extensive performance data, but little in the way of comparison to indicate which architectures are preferable from a cost standpoint given specific performance requirements. Any reasonable architecture can be configured to provide a given level of performance. Our goal is to fix requirements on performance, configure architectures to meet these requirements, then compare the cost of the architectures. To compare costs we use a count of the number of pin-limited chips needed to realize the architecture. Transistor count plays a part in this comparison, but the additional issue of packaging the transistors onto pin-limited chips is considered. We have chosen chip count because it is a dominant component in the cost of a switching system once prototyping has been completed and large scale production is underway.

In this discussion we use the term *switching system* to refer to the functional unit that interconnects the external data links. The switching system is responsible for receiving packets from external links, routing them as appropriate and transmitting the packets on external links. Within the switching system there is a *network* or *switching fabric* that performs the actual routing function. Many of the networks we consider are constructed by interconnecting multiple copies of some smaller building block. We use the term *switch element* to refer to the smaller building block. Switch elements will be organized in *stages*, with interconnection restricted to adjacent stages. In a k stage network, the inputs are connected to switch elements in stage 0, while the outputs are connected to switch elements in stage $k-1$. In packaging, the chips will sometimes be organized in *ranks*, with interconnection restricted to adjacent ranks.

There are a number of parameters which apply to all of the networks described. Their definitions are given below.

[0]This work was supported by the National Science Foundation, Bell Communications Research, BNR, DEC, Italtel SIT, NEC and NTT. The author was partially supported by an Olin Graduate Fellowship from Washington University.

n number of inputs to the network
l external link data rate
f network clock speed
p chip dimension ($2p$ = number of pins per chip)

We are interested in differences between switching systems based on architectural choices as opposed to details of implementation. Thus, in Sections 2-5 we consider several broad categories of systems based on high level architecture choices. Within each category we consider one or more alternatives and develop an equation for the chip count for each alternative. These equations are used to make plots of chip count for each network over a range of parametric values. In Section 6 we compare the chip counts of the various networks. Clearly the chip count depends on the strategy used to assign components of the system to chips. Included in this paper is the packaging strategy for each architecture. Every attempt is made to optimize the packaging so that a minimum number of chips is required. In considering the chip count, we focus on the switching network of each switching system and ignore the input and output circuits which interconnect the external data links. This is done on the grounds that the input and output circuit complexity is comparable for each of the networks. Section 7 contains conclusions.

2. Crossbar Networks

The first category of networks we consider are crossbar networks. These are fairly simple architectures based on a crossbar interconnection between inputs and outputs. The example we consider here is the Knockout network [15].

2.1. Knockout Network

The structure of the Knockout network is shown in Figure 1. The network consists of n^2 disjoint paths, connecting each input to each output. At each output there are n filters and a knockout concentrator responsible for directing the packets which are destined for the output into the output buffer. In order to keep up with the external link data rate, we configure the network with $m = \lceil l/f \rceil$ parallel planes, each a bit serial Knockout network, with all bits for a particular packet confined to the same plane. This same approach will be used for many of the other architectures considered in this paper. This means that a multiplexor and buffer would be needed at the outputs of each plane to combine the m separate packet streams. In addition, some logic would be needed at the inputs to separate the packet stream into m planes. In considering network chip count, we have not included the components which separate and combine packets into multiple planes. Motivated by pin constraints, the packaging strategy for the Knockout network is to package each plane separately, and within a plane to package each knockout concentrator separately.

Each knockout concentrator must consider the n incoming packets and select at most L to reach the output buffer, where L denotes the number of outputs of the knockout concentrator. Clearly, the choice of L affects the packet loss. Figure 2 shows the construction

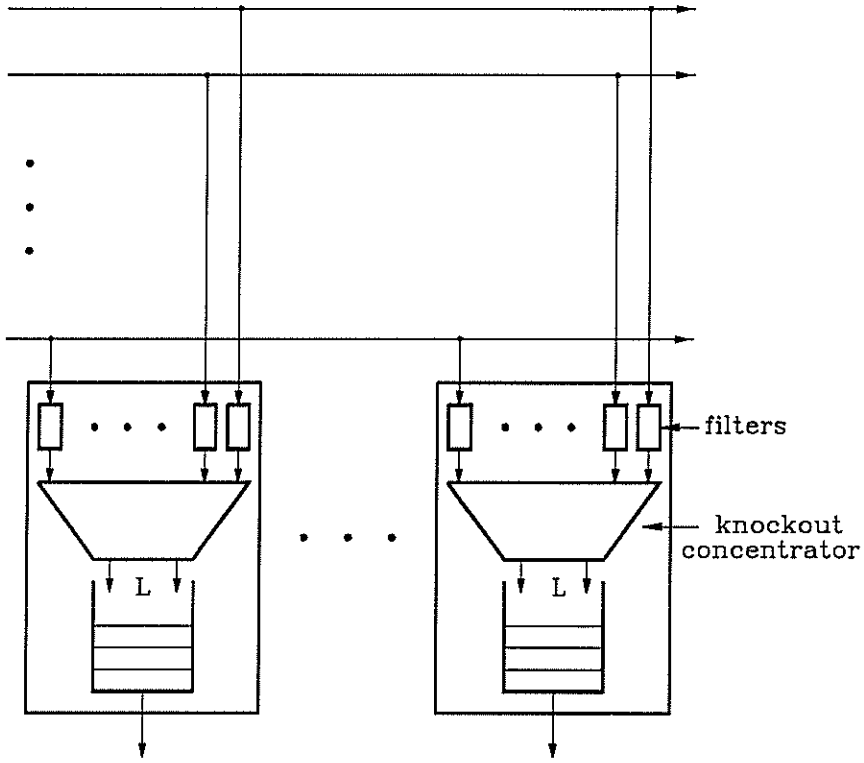

Figure 1: Knockout Network

of an $n : L$ knockout concentrator from a tree of $p' : L$ concentrators. In order to package one concentrator per chip, we require p' and L to satisfy $p' + L \leq 2p$. At each level of the tree the number of outputs decreases by a factor of L/p'. The number of levels (and thus, ranks of chips) is the smallest integer i satisfying $n(L/p')^i \leq L$. In rank j, $1 \leq j \leq i$, there are $n/L(L/p')^j$ chips. In practice L will often be small enough that for the purposes of estimating the chip count we will allow $p' = 2p$. Also, we assume that the output buffer can be packaged with the last $p' : L$ concentrator.

Assuming $p' = 2p$, the chip count for a one bit knockout concentrator, C_{kc}, is

$$C_{kc} = \frac{n}{L} \sum_{j=1}^{i} \left(\frac{L}{2p} \right)^j .$$

The summation can be reduced to a closed form; that form is omitted here. There are n knockout concentrators and $\lceil l/f \rceil$ planes, thus the total number of chips for the network is $C = n\lceil l/f \rceil C_{kc}$.

2.2. Knockout Chip Count

With the equations developed in the previous section, we are in a position to examine chip count as we vary network requirements. Specifically, we consider network size

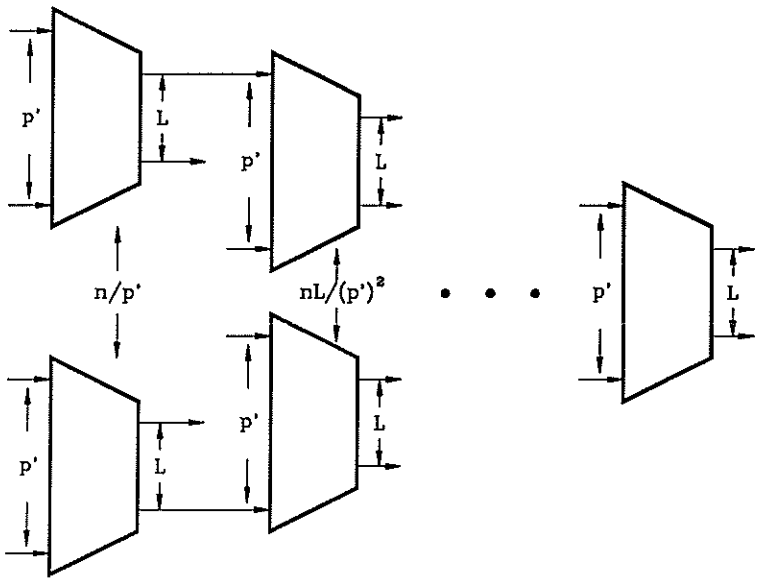

Figure 2: Knockout Concentrator

$n \in \{16, 64, 256, 1024, 4096\}$, external link speed $l \in \{100, 200, \ldots, 1000\}$ Mb/s and chip dimension $p \in \{32, 64\}$. We fix the network clock speed f to be 100 MHz. These choices will apply to all architectures considered here. Specific to the Knockout network is the choice of L. In reference [15] it is shown that $L = 8$ is sufficient for a packet loss rate of 10^{-6} as $n \to \infty$. We have chosen to use $L = 8$.

Figure 3 shows the chip count for the Knockout network. The top two plots contain all values of n; the bottom two plots display the curves for $n = 16, 64, 256$ on a different scale. As is indicated by the chip count equation, there is a linear dependence on the link speed. Increasing n significantly increases the number of chips per port, as the knockout concentrators become more complex. The packaging is pin constrained, thus increasing p from 32 to 64 cuts in half the chip count.

3. Sorter Based Networks

Another category of networks is distinguished by the use of a sorting circuit followed by a banyan network. These networks are commonly referred to as Batcher-banyan networks. (Batcher refers to the designer of the most common sorting network [1].) These networks exploit the fact that a banyan network is nonblocking if the active inputs are consecutive and the packets at these inputs are destined for outputs in increasing order. The use of a sorting network followed by a banyan network yields a nonblocking network, provided no two inputs have packets destined for the same output. The challenge of sorter-banyan network design is to resolve the problem of *contention* between inputs with packets destined for the same output.

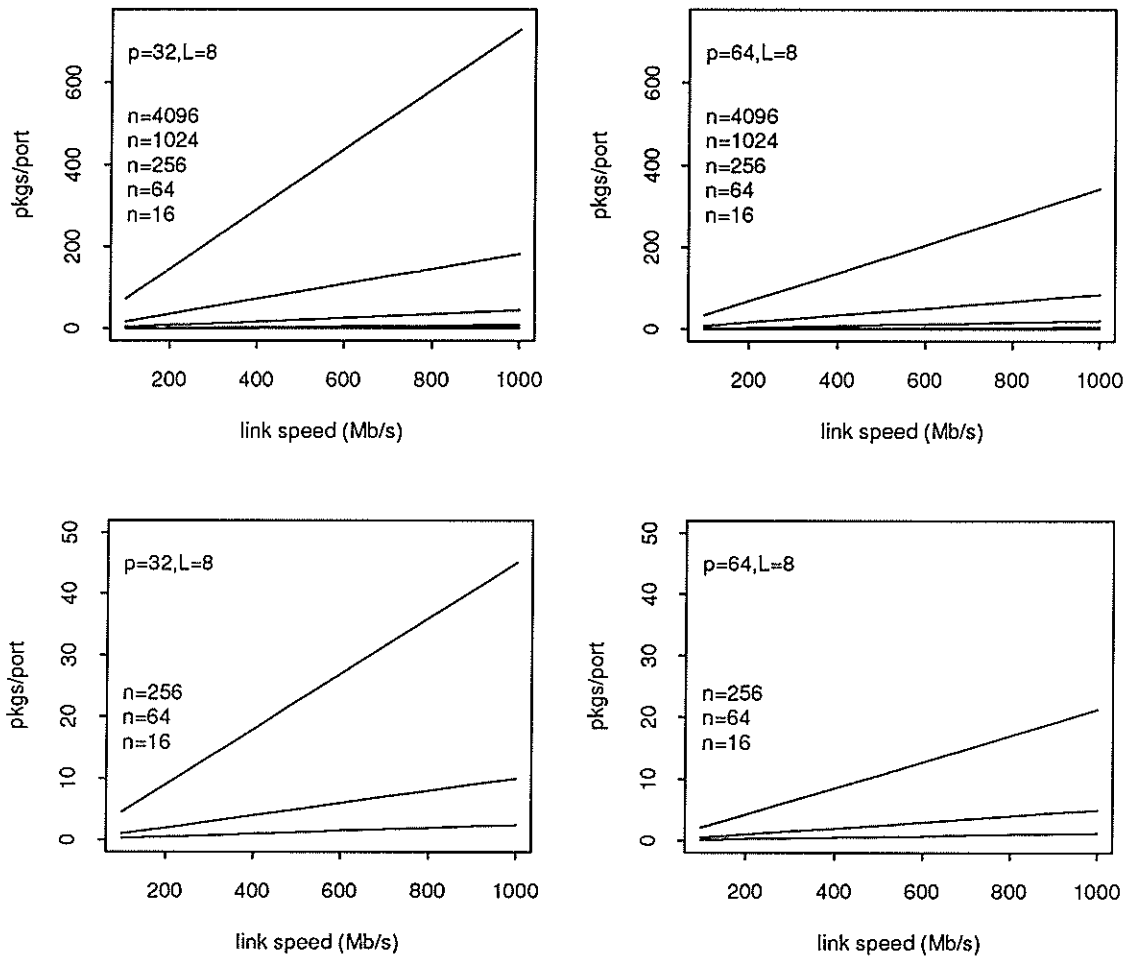

Figure 3: Knockout Chip Count

3.1. Sunshine Network

The Sunshine network [8] shown in Figure 4 is an example of a Batcher-banyan network. A sorting circuit is used to sort the packets by destination address. A trap considers the sorted packets and marks Y per destination to be winners, where Y is the number of banyan networks. A second sorting network separates the winners from the losers and sorts the losers by priority. A selector routes the highest priority losers to the recirculation ports and passes the winners to the banyan networks. Due to the connectivity of the Sunshine network, the packaging is limited by pin constraints. As in the Knockout network, we divide the network into $m = \lceil l/f \rceil$ parallel planes. Each plane is a bit-serial Sunshine network. Within a plane we separately package each sorter, the trap, the selector, each banyan and the delay circuits. The chip count for each of these components is described below.

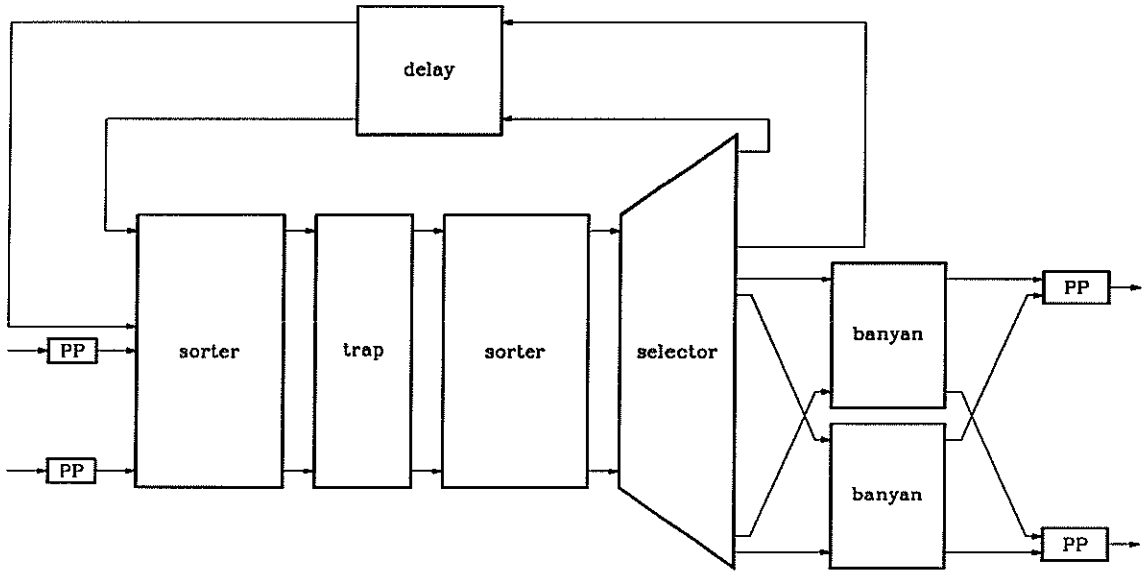

Figure 4: Sunshine Network

3.1.1. Banyan Packaging. A banyan network with n inputs composed of $d \times d$ crossbars, denoted $Y_{n,d}$, is shown in Figure 5. An isomorphic representation is shown in Figure 6, where $D_{n,d}$ denotes a delta network constructed of $d \times d$ crossbars. We would like to package this network using chips with dimension p. It is clear from this picture that $\log_d p$ stages can fit on one rank of n/p chips. There are $\log_d n$ stages in the network, thus packaging an n input banyan network requires $\lceil \log_d n / \log_d p \rceil = \lceil \log_p n \rceil$ ranks of chips with n/p chips per rank.

3.1.2. Sorter Packaging. A Batcher sorting network is constructed of 2×2 comparison elements as shown on the left of Figure 7. This element compares the address fields of incoming packets and routes the packet with the smaller address field to the output at the head of the arrow. The larger is routed to the output at the tail of the arrow. An n input sorting network S_n is defined recursively as shown on the right of Figure 7. The subnetwork M_n is responsible for merging the sorted outputs from each of the subnetworks $S_{n/2}$. The merge network is a binary banyan network.

Figure 8 shows the sorting network with the subnetworks $S_{n/2}$ expanded. We could continue to expand the sorting subnetworks into smaller sorters followed by merge networks. Consider expanding the network until the sorters have size p. The sorters are followed by M_{2p}, then M_{4p}, and so on until M_n. This is sketched in Figure 9.

We can package the network in Figure 9 as follows. Each chip in the first rank contains a sorter S_p. There are n/p of these chips. Next consider the merge networks M_{2p}. There are $n/(2p)$ such networks. Each network has $\lg(2p) = 1 + \lg p$ stages. Based on an isomorphic construction of the binary banyan network, we can fit $\lg p$ stages on a single chip before the interconnection prohibits this. Thus each network M_{2p} can be packaged in two ranks with 2 chips per rank. The merge networks M_{4p} each have $\lg(4p) = 2 + \lg p$ stages and

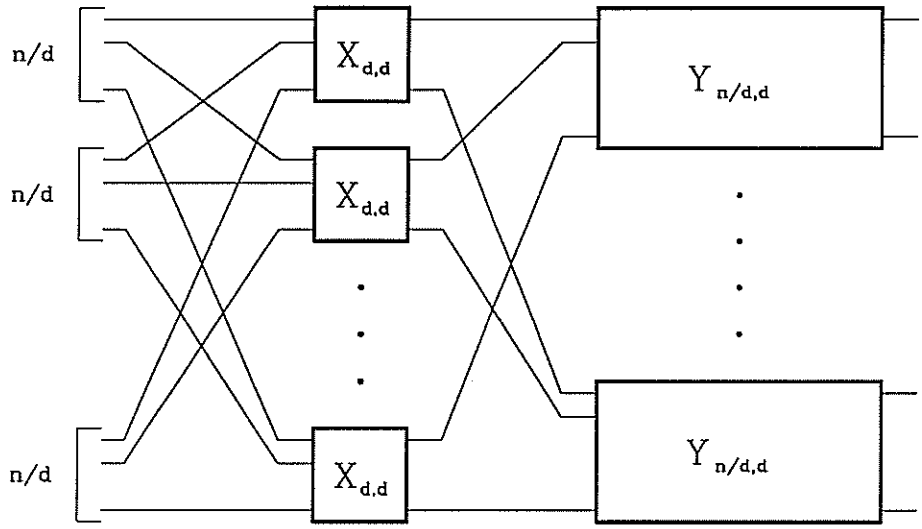

Figure 5: Banyan Network $Y_{n,d}$

can be packaged in two ranks with 4 chips per rank. There are $n/(4p)$ such networks. In general, each network $M_{2^i p}$ can be packaged in $1 + \lceil i/\lg p \rceil$ ranks with 2^i chips per rank, where $1 \leq i \leq \lg(n/p)$. There are $n/(2^i p)$ merge networks $M_{2^i p}$.

Combining these results, the number of ranks in the sorter is

$$\text{sorter ranks} = 1 + \lg \frac{n}{p} + \sum_{i=1}^{\lg \frac{n}{p}} \left\lceil \frac{i}{\lg p} \right\rceil ,$$

with n/p chips per rank. More generally, if n/p is not a power of two, then we round up to the nearest power of two, essentially building a slightly larger sorter. The number of ranks in the sorter is

$$\text{sorter ranks} = 1 + \lceil \lg \frac{n}{p} \rceil + \sum_{i=1}^{\lceil \lg \frac{n}{p} \rceil} \left\lceil \frac{i}{\lg p} \right\rceil ,$$

with $2^{\lceil \lg(n/p) \rceil}$ chips per rank.

3.1.3. Selector Packaging. The selector has $n + r$ inputs and $n + 2r$ outputs, r that recirculate and $n + r$ that connect to the banyan networks. Considering the inputs from the "top" down, there is a contiguous block of zero or more loser packets sorted by priority, followed by a contiguous block of one or more winner packets sorted by output. The selector can be implemented with an array of $n + 2r$ selector (or s-) elements, each of which considers a single input to the selector and connects to one output of the selector. The top r s-elements consider the top r inputs to the selector. If the input is a loser, the s-element passes the packet to a recirculation port. It the input is a winner, the s-element ignores the packet. The next r s-elements also consider the top r inputs to the selector. If the input is a winner, the s-element passes the packet to a banyan network. The last n

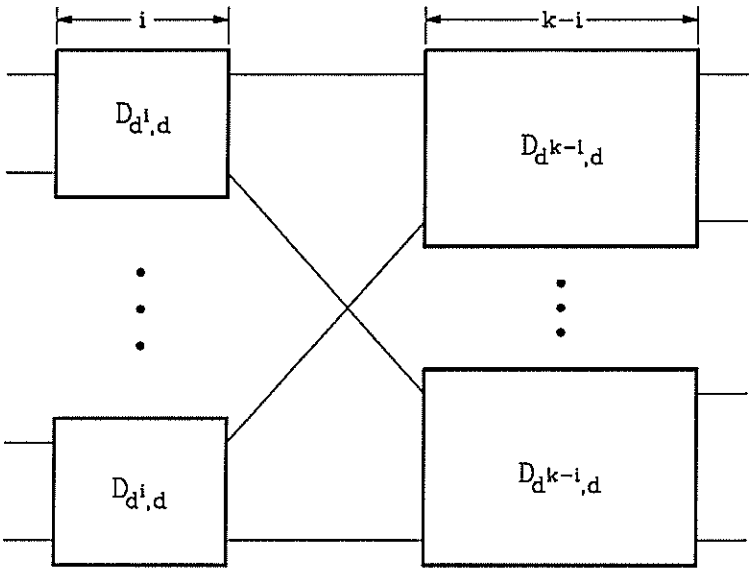

Figure 6: Isomorphic Representation of $Y_{n,d}$

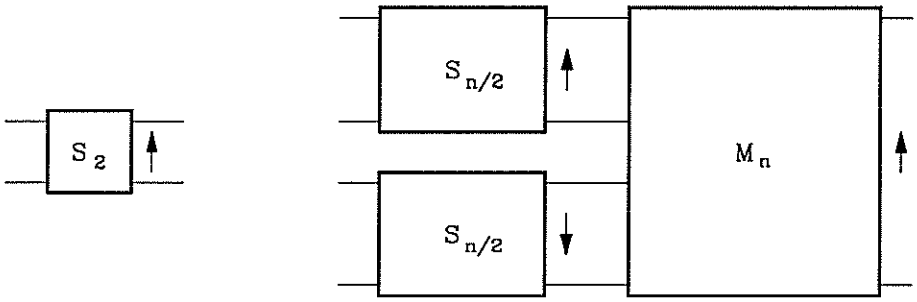

Figure 7: Batcher Sorting Network, Top Level

s-elements consider the last n inputs to the selector. If the input is a winner it is passed to a banyan network. It is sufficient to use $(n + 2r)/p$ chips for the selector.

3.1.4. Sunshine Packaging. We can now use the chip counts developed in the previous three sections to determine the total chip count for the Sunshine switch. We let r denote the number of recirculation ports. Each sorter has $n + r$ inputs and outputs. The number of ranks of chips for each sorter is given by

$$\text{sorter ranks} = 1 + \left\lceil \lg \frac{n+r}{p} \right\rceil + \sum_{i=1}^{\lceil \lg \frac{n+r}{p} \rceil} \left\lceil \frac{i}{\lg p} \right\rceil .$$

There are $2^{\lceil \lg((n+r)/p) \rceil}$ chips per rank. For example, if $n = 256$, $r = 128$ and $p = 64$, then each sorter requires seven ranks of eight chips each.

The trap requires one rank of $(n+r)/p$ chips. The selector requires one rank of $(n+2r)/p$ chips. As explained above, each banyan can be packaged in $\lceil \log_p n \rceil$ ranks with n/p chips

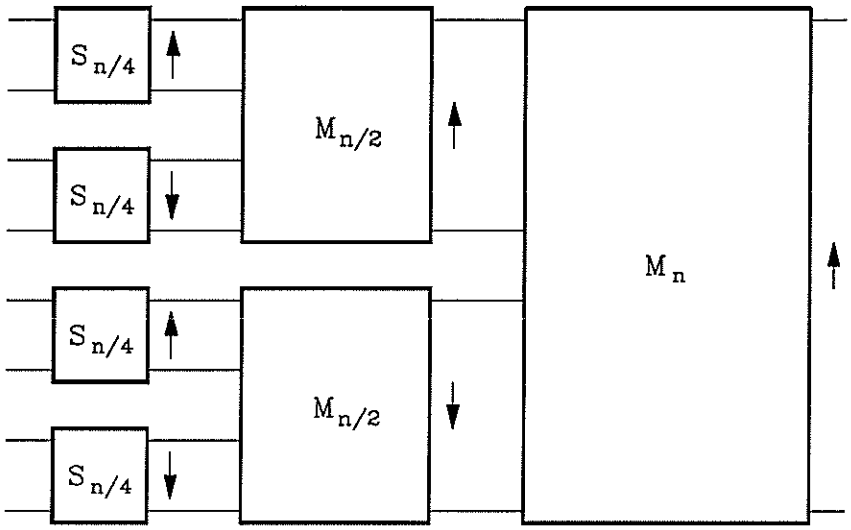

Figure 8: Batcher Sorting Network with $S_{n/2}$ Expanded

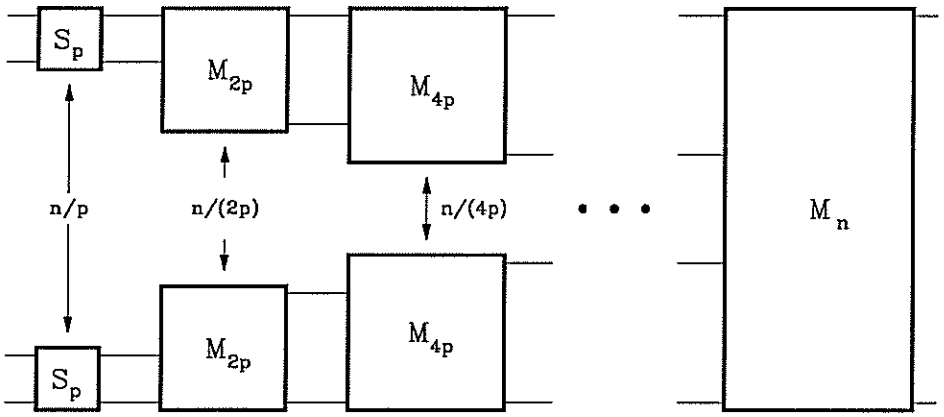

Figure 9: Batcher Sorting Network Expanded to S_p

per rank. We let Y denote the number of banyan networks. The delay circuits can be packaged in one rank of r/p chips.

At each output there are Y inputs arriving from each of the m planes. It is necessary to concentrate the packets from these inputs to the output. We assume that it is sufficient to concentrate the inputs to m output lines. Because packets may arrive at a faster rate than the output data rate, some storage is needed in the concentrator. We assume each concentrator is implemented by a set of shift registers with a crossbar for writing to the registers and another crossbar for reading from the registers. On the write side, the inputs to the crossbar are the mY input lines. On the read side, the outputs of the crossbar are the m output lines. Within each group of Y inputs we expect on average one packet arrival per cycle, thus we can get away with a modest number of shift registers. The transistor count for the concentrator is determined by the number of shift registers and the size of the crossbars. Depending on the various parameters, packaging of the concentrators may be pin

limited or transistor limited. To the extent it is possible, we package multiple concentrators on a single chip. We let C_{out} denote the chip count due to all of the output concentrators.

The chip count per plane for all components except the output concentrators, C_p, is

$$C_p = 2^{\lceil \lg(n+r)/p \rceil + 1} \left(1 + \left\lceil \lg \frac{n+r}{p} \right\rceil + \sum_{i=1}^{\lceil \lg \frac{n+r}{p} \rceil} \left\lceil \frac{i}{\lg p} \right\rceil \right) + \frac{1}{p}(2n + 4r + Y n \lceil \log_p n \rceil).$$

The number of planes is $\lceil l/f \rceil$, thus the number of chips in the network is $C = \lceil l/f \rceil C_p + C_{out}$.

3.2. Sunshine Chip Count

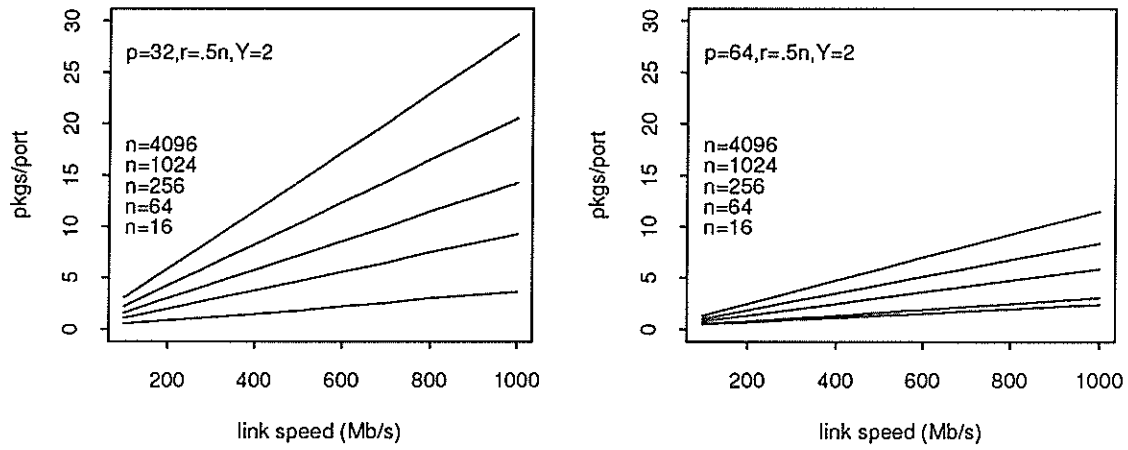

Figure 10: Sunshine Chip Count

Figure 10 gives the chip count for the Sunshine network. In reference [8] it is suggested that $r = 0.4n$ and $Y = 2$ are sufficient to achieve packet loss rates of 10^{-9}. We have chosen to use $r = 0.5n$ and $Y = 2$. As in the Knockout network, the chip count increases linearly with the link data rate. Increasing p from 32 to 64 more than halves the chip count.

3.3. Lee's Network

Lee's network [10] is a unification of the Batcher-banyan network and the Knockout network. The design is modular; it is constructed from k independently operated switch modules that are connected by concentrators at the outputs of the network, as shown in Figure 11. Each switch module has $s = n/k$ inputs and n outputs. Figure 12 shows a switch module in more detail. It consists of an $s \times s$ sorting network, followed by s binary trees (one per output of the sorting network) with k leaves each, followed by k banyan networks with s inputs and s outputs. Each switch module has one output line to each network output. There is a concentrator at each network output to collect the lines from the k switch modules. In the extreme when $k = 1$ (thus $s = n$) the network is a Batcher-banyan network. In the extreme when $k = n$ (thus $s = 1$) the network is the Knockout network.

It can be shown that if no two inputs to a switch module request the same output, then the module is nonblocking. (We already know this is the case for a Batcher network followed by one banyan network. Lee's paper shows this to hold when the Batcher network is followed by binary trees and multiple banyan networks.) The problem of contention between inputs to a switch module requesting the same output is solved in two ways. First, each switch module is designed with nL outputs, L per network output. A packet can be routed through the switch module to any of the L outputs associated with the desired network output. This decreases the probability that a packet will be blocked. Second, a

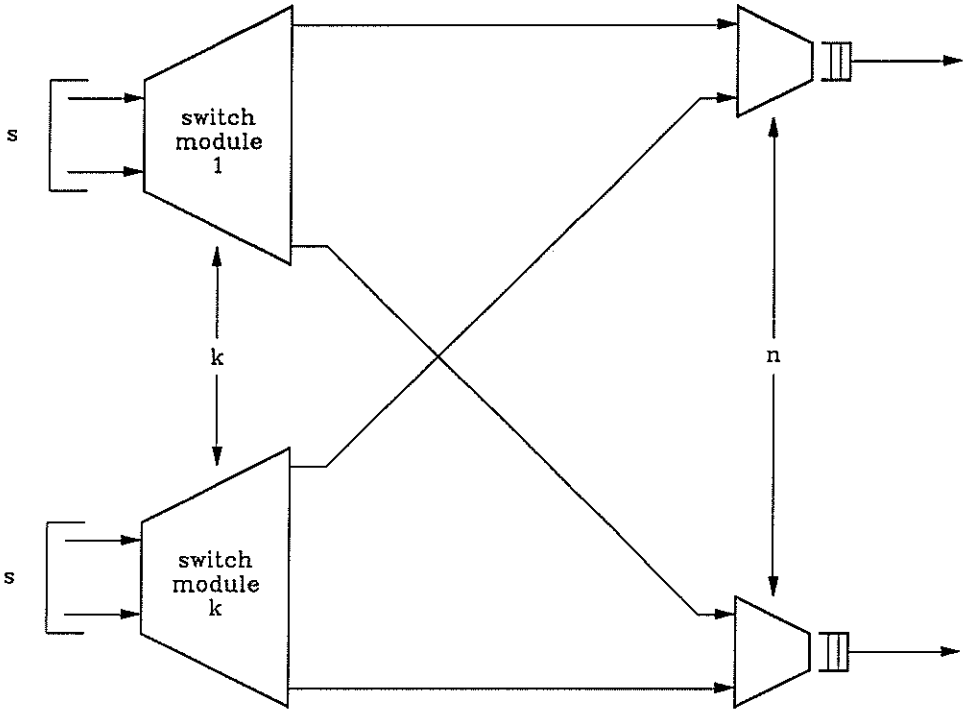

Figure 11: Lee's Network

ring reservation algorithm is used to allow packets to reserve access to a particular switch module output. This algorithm has the disadvantage of being sequential within a switch module.

To package the network, we divide the network into $m = \lceil l/f \rceil$ parallel planes. Next we consider each switch module separately. Let k be the number of switch modules, s be the number of inputs per switch module and L be the number of switch module outputs per network output. Each module contains one $s \times s$ sorting network, s binary trees with kL leaves each and kL $s \times s$ banyan networks. Thus, the number of chips per switch module in a single plane, C_{sm}, is

$$C_{sm} = 2^{\lceil \lg(s/p) \rceil} \left(1 + \left\lceil \lg \frac{s}{p} \right\rceil + \sum_{i=1}^{\lceil \lg \frac{s}{p} \rceil} \left\lceil \frac{i}{\lg p} \right\rceil \right) + \frac{s}{\lfloor \frac{2p}{kL} \rfloor} + kL \lceil \log_p s \rceil \frac{s}{p}.$$

At each concentrator there are kL inputs from each of the m planes. As in the Sunshine network, the concentrator must direct the packets from these inputs to the output. We use the same implementation approach as described for the Sunshine network and let C_{out} denote the chip count due to all of the output concentrators.

There are $\lceil l/f \rceil$ planes and k modules, thus the number of chips in the entire network is $C = \lceil l/f \rceil (n + kC_{sm}) + C_{out}$.

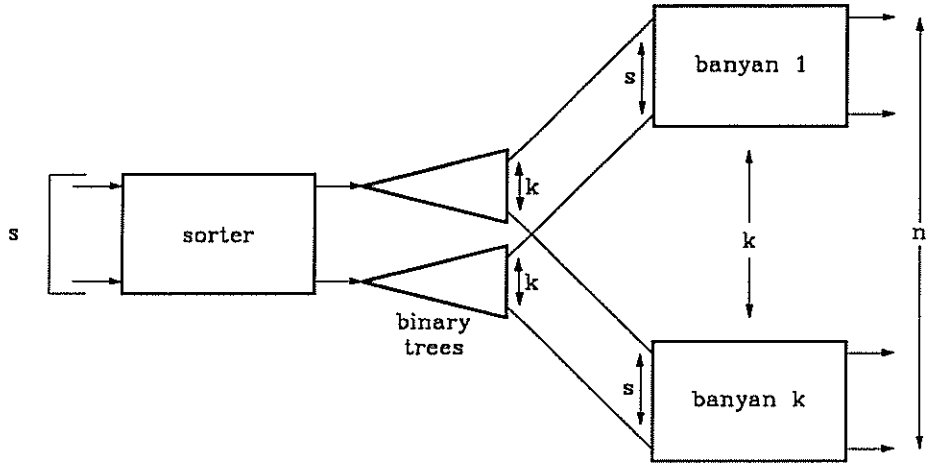

Figure 12: Switch Module of Lee's Network

3.4. Lee's Chip Count

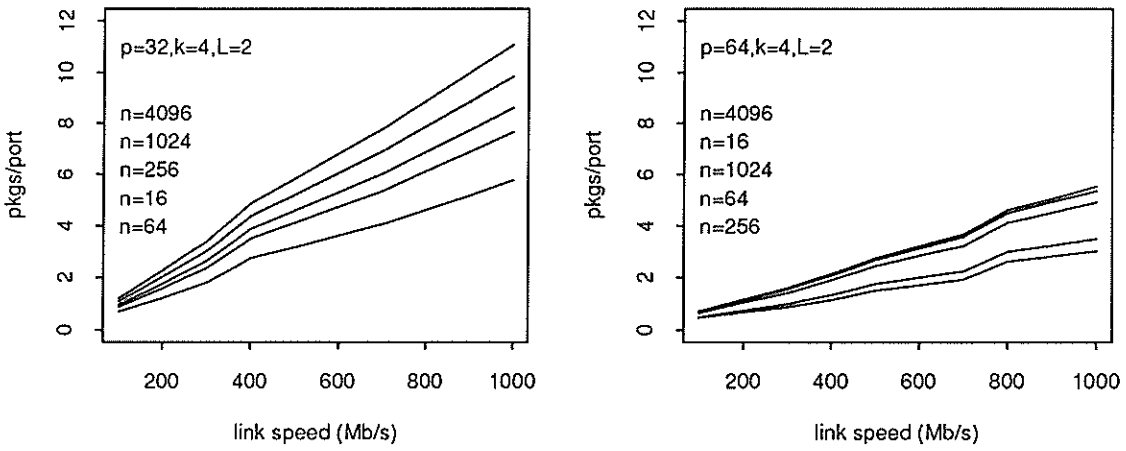

Figure 13: Lee's Chip Count

Figure 13 gives the chip count for Lee's network. In reference [10] it is suggested that $k = 4$ and $L = 2$ result in high throughput with a reasonable window for look-ahead contention resolution. We have used these values for k and L over all values of n. This leads to some rather surprising results as n is varied. At low values of n, the switch modules are very small compared to the chip dimension. This causes the chip count per port to *decrease* as n increases until the switch module size is comparable to the chip dimension.

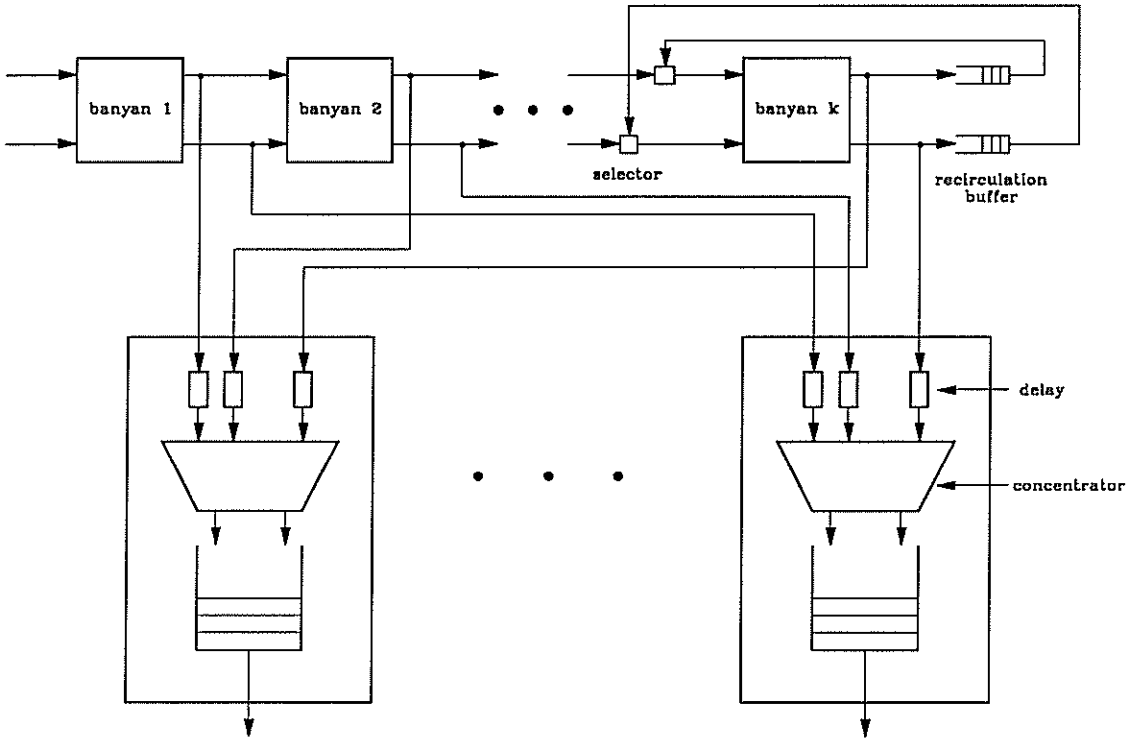

Figure 14: Tandem Banyan Network

4. Unbuffered Networks with Deflection Routing

As indicated by their name, unbuffered networks have no buffering within the switch fabric. We consider two examples of such networks, both made up of multiple stages of switch elements. Some scheme must be devised to handle the situation in which more than one packet is destined for the same output of a switch element. Deflection routing is one such scheme. In deflection routing, one packet is sent to the the desired output while the other packet is deflected to a different output. Those packets which are deflected are given other chances to reach their destination. The two networks we consider are the Tandem Banyan network and the Shuffleout network.

4.1. Tandem Banyan Network with Recirculation

The Tandem Banyan network consists of k banyan networks arranged one after the other, as shown in Figure 14 [13]. For banyans 1 through $k-1$, output i is connected both to input i of the next banyan and to output circuit i. The outputs of banyan k are connected both to recirculation buffers and to the appropriate output circuit. The recirculation buffers feed back into banyan k through a selector which mediates between packets from banyan $k-1$ and recirculation packets.

Each packet is routed through the banyan networks towards the desired output. Because the banyan network can block, conflicts may occur between packets seeking the same output

of a particular switch. If a conflict occurs between two packets, one is allowed to continue on the correct path, while the other is marked as misrouted and forced off the correct path. (Once a packet is marked as misrouted, it cannot force an unmarked packet off the correct path.) When a packet reaches the output of a banyan network, it is examined to see if it was correctly routed. If so, it is sent to the output circuit. If not, it is unmarked and sent to the next banyan for another try.

The delay elements in each output circuit ensure that packets arriving at the switch at the same time reach the output circuits at the same time, provided no recirculation occurs. This helps to keep the packets in the correct order. However, the recirculation can cause some packets to be received out of order.

Once again, we package the network in multiple planes. We let k denote the number of banyan networks. Each banyan network can be packaged in $\lceil \log_p n \rceil$ ranks of n/p chips per rank. The total chip count for the banyans in a single plane, C_b, is

$$C_b = k\frac{n}{p}\lceil \log_p n \rceil.$$

The recirculation buffers and selectors can be packaged together. We let B denote the number of packet slots in each recirculation buffer. The transistor count required for the recirculation buffer in one plane is xLB, where L is the packet length in bits and x is the transistor count per memory cell. The transistor count for the selector is negligible in comparison. Based on transistor considerations, we can fit $\lfloor T/(xLB) \rfloor$ buffers and selectors per chip, where T is the maximum transistor count per chip. Based on pin limitations we can fit $\lfloor 2p/3 \rfloor$ buffers and selectors per chip. Thus, the total chip count for the recirculation buffers and selectors in a single plane, C_r, is

$$C_r = \frac{n}{\min\{\lfloor T/(xLB) \rfloor, \lfloor 2p/3 \rfloor\}}.$$

At each output circuit, there are k inputs arriving from each of the m planes. As we saw earlier, these inputs must be concentrated to m output lines. We use the implementation approach described previously, with C_{out} denoting the chip count due to all of the output circuits.

There are $\lceil l/f \rceil$ planes, thus the chip count for the entire network, C, is

$$\begin{aligned} C &= \left\lceil \frac{l}{f} \right\rceil (C_b + C_r) + C_{out} \\ &= \left\lceil \frac{l}{f} \right\rceil n \left(\frac{k}{p} \lceil \log_p n \rceil + \frac{1}{\min\{\lfloor T/(xLB) \rfloor, \lfloor 2p/3 \rfloor\}} \right) + C_{out}. \end{aligned}$$

4.2. Tandem Banyan Chip Count

Figure 15 gives the chip count for the Tandem Banyan network. The number of banyans used for each value of n was determined from reference [13] to achieve low packet loss rates with reasonable size recirculation buffers. We have chosen $k = 3$ when $n = 16$, $k = 4$ when

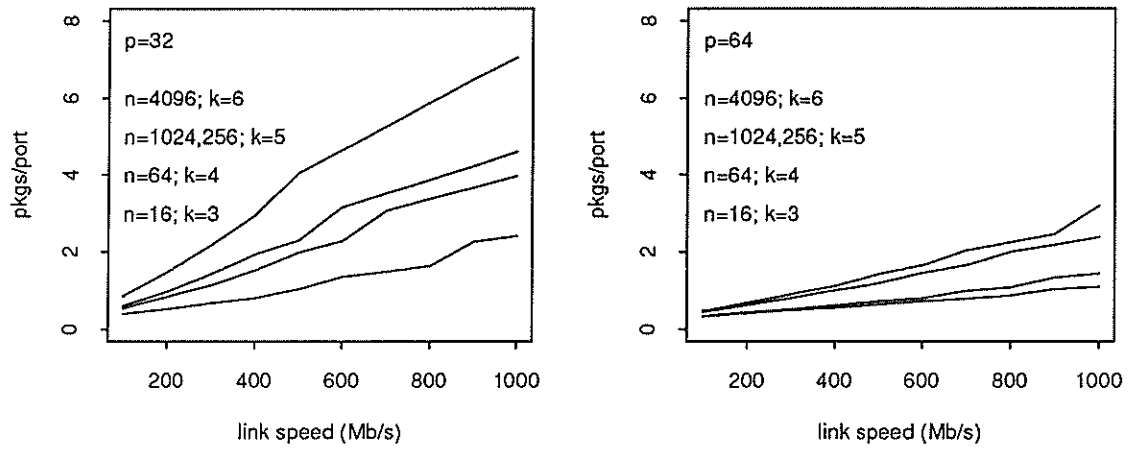

Figure 15: Tandem Banyan Chip Count

$n = 64$, $k = 5$ when $n = 256, 1024$ and $k = 6$ when $n = 4096$. We used $B_r = 4$ for all values of n. The chip count per port is the same for $n = 256$ as for $n = 1024$ because the term $\lceil \log_p n \rceil$ has the same value for $n = 256$ as $n = 1024$ when $p = 32, 64$. These are the lowest chip counts we have seen thus far.

4.3. Shuffleout Network

The Shuffleout network is similar to the Tandem Banyan network in that it routes packets through a series of "units" with the outputs of each unit connected both to the next unit and to the output circuits [7]. In the Tandem Banyan, the unit is a banyan network. In the Shuffleout network, the unit is one stage of a shuffle interconnection network. The Shuffleout network consists of 2×4 switch elements, with each switch element connected in the shuffle pattern to two switch elements in the next stage and connected to two output circuits. There are also some 2×2 switch elements that connect to two switch elements in the next stage, with no connection to the output circuits. Figure 16 shows the Shuffleout network with five inputs, three recirculation ports and three stages of shuffle interconnect.

As in the Tandem Banyan, packets which leave the last stage of the Shuffleout network without success are recirculated into the network. In the Tandem Banyan they recirculate into the last banyan network. In the Shuffleout, the output lines from the last stage of the network are concentrated onto r lines which pass through a delay then recirculate to the inputs of the network. This means that r of the network inputs are reserved for use by recirculating packets and cannot be used for packets coming directly from the external links. The value of r must be chosen so that packet loss at the concentrator is tolerable. We let n denote the number of network inputs that are connected to external links.

Again, we package the network in multiple one-bit planes. Each one bit plane of the

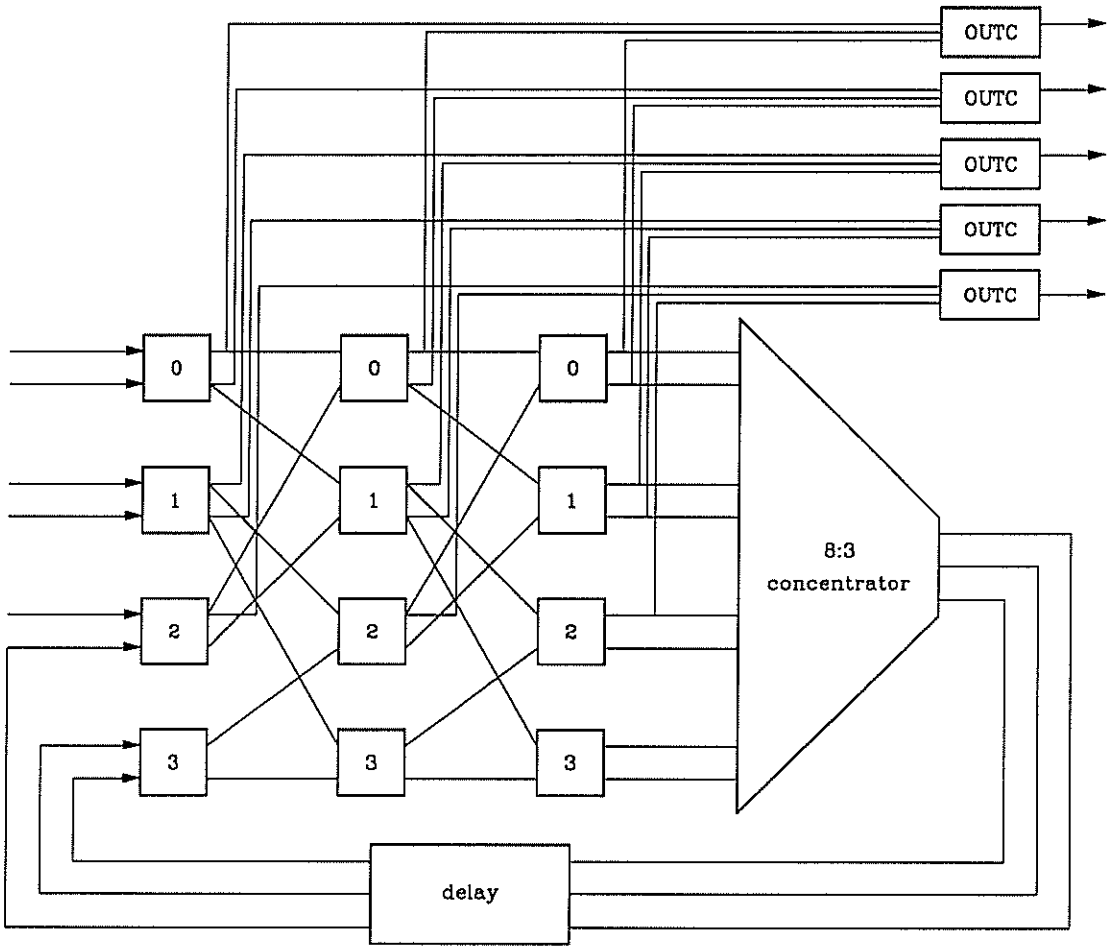

Figure 16: Shuffleout Network

delay takes r/p chips. The total chip count for the delay in a single plane, C_d, is

$$C_d = \frac{r}{p}.$$

The packaging of the switch elements is complicated by the fact that $n/2$ of the switch elements in each stage have four outputs, two that go to the next stage and two that go to the output circuits. The rest of the switch elements have two outputs that go to the next stage. Using an isomorphic construction of the shuffle network, and assuming the switch elements with two outputs are evenly spaced within a stage, we can fit i stages of one plane on one chip, where i is the largest integer satisfying $2^i + (ni2^{i-1})/(n+r) \leq p$. There are $\lceil k/i \rceil$ ranks of chips with $(n+r)/2^i$ chips per rank. The total chip count for the switch elements in a single plane, C_s, is

$$C_s = \left\lceil \frac{k}{i} \right\rceil \frac{n+r}{2^i}.$$

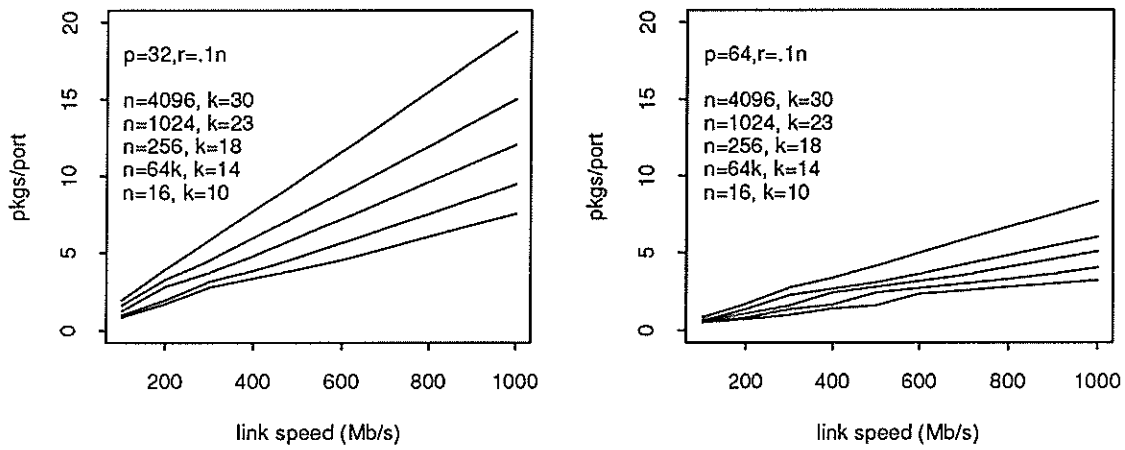

Figure 17: Shuffleout Chip Count

After the last stage of the shuffle interconnect there is an $n + r : r$ concentrator. The implementation of the concentrator depends upon its size. If it is small, it can be implemented as a knockout concentrator. As it becomes larger, it is not possible to use a knockout concentrator. Rather, multiple knockout concentrators must be used, each connected to a fraction of the recirculation ports. This can result in additional packet loss at the concentrator. It is difficult to give a closed form expression for the concentrator chip count, thus in the expression we leave this term as C_c. In our experiments, we determined C_c for each value of n and r of interest.

At each output circuit, there are k inputs arriving from each of the m planes. We use the same implementation as was described earlier and let C_{out} denote the chip count due to all of the output circuits.

There are $\lceil l/f \rceil$ planes, thus the chip count for the entire network, C, is

$$\begin{aligned} C &= \left\lceil \frac{l}{f} \right\rceil (C_s + C_d + C_c) + C_{out} \\ &= \left\lceil \frac{l}{f} \right\rceil \left(\left\lceil \frac{k}{i} \right\rceil \frac{n+r}{2^i} + \frac{r}{p} + C_c \right) + C_{out}. \end{aligned}$$

4.4. Shuffleout Chip Count

Figure 17 shows the chip count for the Shuffleout network. We have chosen to use $r = 0.1n$, as suggested by the authors in reference [7]. Based on limited data available in the paper we chose values for k, the number of shuffle stages. We have used $k = 10$ when $n = 16$, $k = 14$ when $n = 64$, $k = 18$ when $n = 256$, $k = 23$ when $n = 1024$ and $k = 30$ when $n = 4096$. These are almost certainly optimistic estimates of the number of stages needed to achieve low packet loss rates with high offered load.

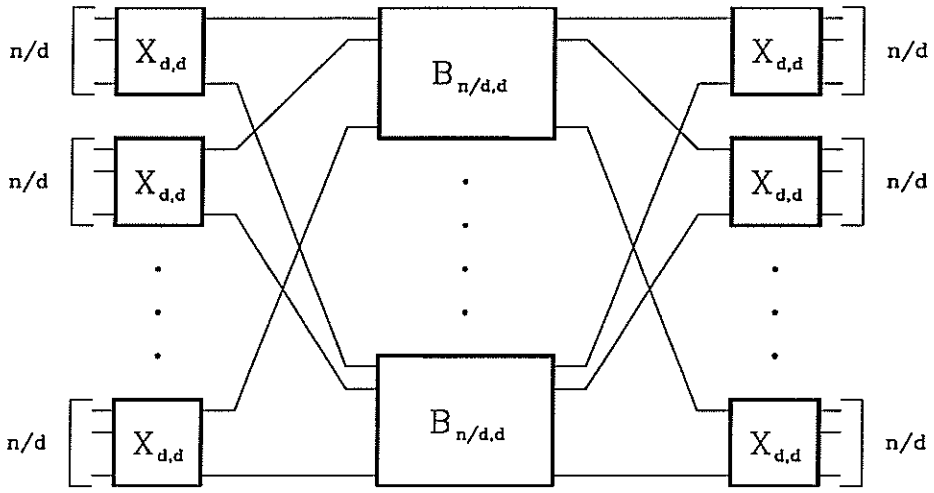

Figure 18: Beneš Networks $B_{n,d}$

5. Buffered Beneš Networks

The last category of networks we consider is buffered Beneš networks. These networks are constructed from $d \times d$ switch elements connected in a Beneš topology [2, 3] with buffering at each switch element. An n input Beneš network constructed from $d \times d$ switch elements has $2\log_d n - 1$ stages with n/d switch elements per stage. The structure of such a network is shown in Figure 18. $X_{d,d}$ is a $d \times d$ crossbar and $B_{d,d} = X_{d,d}$. If n is not a power of d, then a generalized Beneš network can be constructed with $2\lceil \log_d n \rceil - 1$ stages, n/d switch elements per stage and multiple links between pairs of switch elements.

5.1. Switch Element Design

Many of the networks within this category use a *bit-sliced* scheme for organizing the data and control portions of the switch element. If m denotes the width of the data path, then a bit-sliced design divides the data portion of the switch element into m planes, one per data path bit. A single control block sends common control signals to all m planes. Figure 19 shows a bit-sliced switch element with m data slices.

The value of m depends on a number of factors. For example, the external links will often transmit data at a higher rate than the clock rate of the switch element. A wide data path allows the switch element to keep up with the external link data rate. If l is the external link data rate and f is the maximum clock rate, then m must be at least $\lceil l/f \rceil$ in order to keep up. An additional bit may be added to the data path for control information.

5.1.1. Data Slice. The packaging strategy for bit-sliced architectures will be to put as many slices as possible on a single chip, subject to pin and transistor constraints. One bit slice of a $d \times d$ switch element requires $2d$ pins. The number of transistors per data

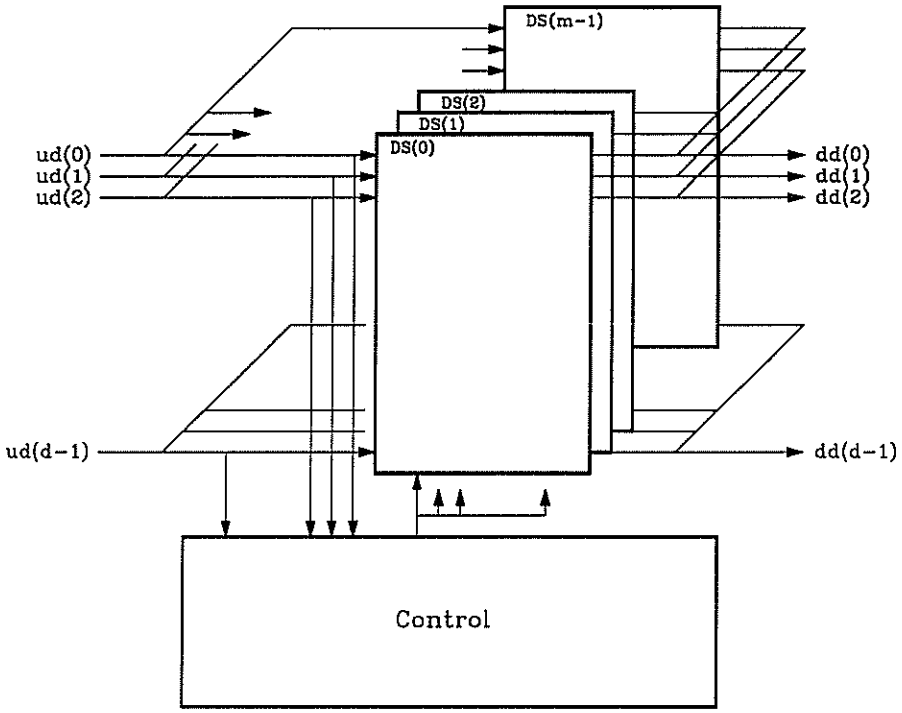

Figure 19: Bit-Sliced Switch Element Design

slice depends on design decisions regarding the amount of buffering and the mechanism for accessing the buffer memory.

The amount of buffering is affected by the location of the buffers and the desired network throughput. We consider three options for the location of the buffers. In *input buffering* the buffers are placed at the inputs to the switch element. A packet arriving on input i is placed into a buffer corresponding to input i. When possible the switch element takes the packet and routes it to the desired output. In *output buffering* the buffers are placed at the outputs of the switch elements. A packet arriving for output i is placed in the buffer corresponding to output i. When possible the packet is sent out on output i. In *shared buffering* the buffers are placed between the inputs and outputs of the switch element. A packet arriving on input i destined for output j is placed in an available buffer. When possible the packet is removed from the buffer and send out on output j.

Through simulation it is possible to determine the number of buffer slots needed to achieve the desired throughput. We let B denote the ratio of the total number of buffer slots in the switch element to d, L denote the packet size in bits and x_1 denote the transistor count per bit of memory. Then the transistor count per data slice due to buffer memory, T_{buf}, is

$$T_{buf} = LdBx_1/m.$$

We consider two mechanisms for accessing the buffer memory, bus access and crossbar access. We describe the use of each mechanism for writing to the buffer; reading is similar. With a bus access scheme, the inputs take turns writing to the buffer memory. Each input

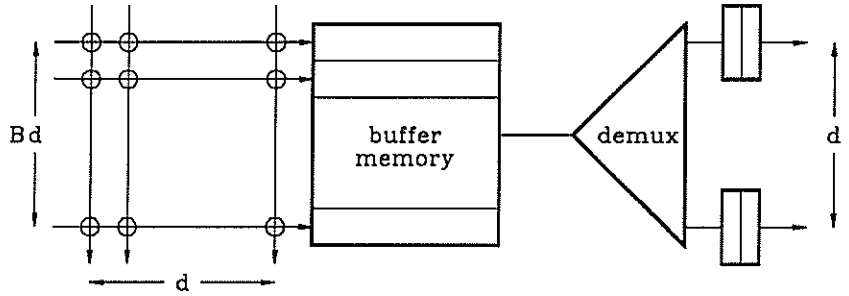

Figure 20: Crossbar and Bus Memory Access

must be capable of storing a packet while waiting its turn. In addition, each input must be able to store a second packet that is arriving while the first is being written to memory. The storage for the packets can be implemented with shift registers. Figure 20 shows a shared memory with bus read access. If x_2 denotes the transistor count per shift register bit, then the transistor count per data slice due for write access to the bus, T_{mwa}, is

$$T_{mwa} = 2Ldx_2/m.$$

With a crossbar access scheme, a crossbar is placed between the inputs and a set of buffer slots, where each buffer slot can store one bit slice of one packet. Each input can access each buffer location through a particular crosspoint. The crossbar is controlled so that no two inputs attempt to write to the same buffer location at the same time. If B is the number of buffer slots per input then a $d \times dB$ crossbar is needed. Figure 20 shows a shared memory with crossbar write access. We let x_3 denote the transistor count per crosspoint. Then the transistor count per data slice due to crossbar write access, T_{cwa}, is

$$T_{cwa} = Bd^2x_3.$$

The need for complex read and write access mechanisms as described above is determined by the type of buffering used. With input buffering, the input side does not require a bus or a crossbar. It is only necessary to use one of the above mechanisms for read access. Likewise, with output buffering, the output side does not require a bus or a crossbar. It is only necessary to use one of the above mechanisms for write access. With shared buffering a mechanism is needed for both write and read access.

Given a buffer size and access mechanisms for writing to and reading from the buffer, it is possible to package the data portion of the switch element. The transistor count per data slice, T_{ds}, is the sum of the transistor count for the buffer memory and the access mechanisms. Each data slice has roughly $2d$ pins. (The actual pin count is slightly higher due to control.) We let T denote the maximum transistor count per chip and p denote the chip dimension. The number of data slices that fit on one chip, b, is given by

$$b = \min\{\lfloor p/d \rfloor, \lfloor T/T_{ds} \rfloor\}.$$

The chip count per switch element, C_{se}, is then given by

$$C_{se} = m/b + C_{control},$$

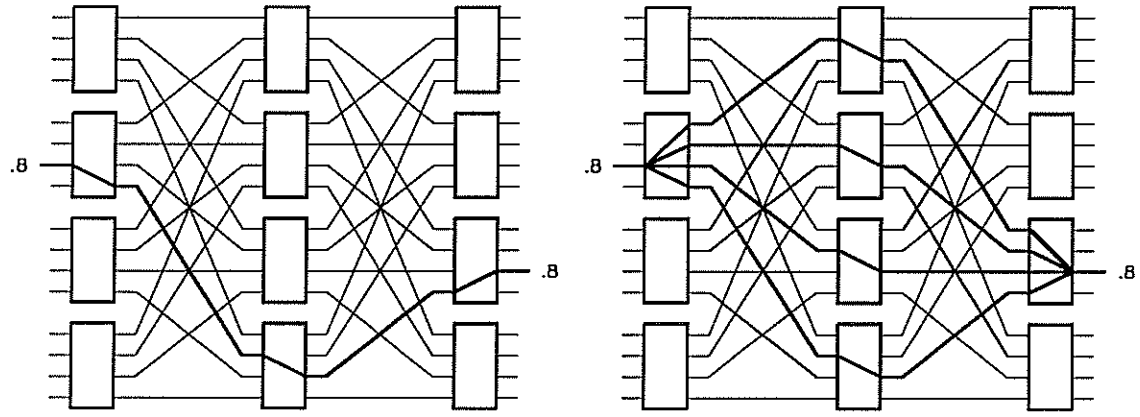

Figure 21: Fixed Path and Per Cell Routing

where $C_{control}$ is the chip count for the control portion of the switch element. Determining $C_{control}$ is discussed next.

5.1.2. Control. It is difficult to say much about the transistor count for the control portion of the switch element because it varies significantly depending upon the specific design. For the purposes of estimating chip count for the control portion we assume $C_{control} = 1$ for all architectures. It is unlikely that any architecture would use more than one chip for control.

5.2. Satisfying Traffic Requirements

In order to compare various architectures, we will impose certain requirements on the performance of each network and configure the network to meet the requirements. Specifically, we will fix the external link data rate and the acceptable packet loss rate. If l is the external link data rate, then the collection of virtual circuits to be routed by the network will be such that the sum of the bandwidth requirements for circuits on any one input or output is at most l. We must determine the internal link rate necessary to support any such collection of virtual circuits.

A virtual circuit *induces* a load on each link of the network equal to the bandwidth required on that link to route packets for the circuit. The induced load on a particular link depends upon the type of routing performed. In a network with *fixed path routing* each virtual circuit is assigned a path through the network and every packet belonging to that connection follows that path. The induced load of a virtual circuit is confined to the links on the path and is equal to the bandwidth of the virtual circuit. The left side of Figure 21 shows a fixed path for one virtual circuit requiring .8 of the bandwidth on an external link. Every link in the path carries a load of .8.

In contrast, a network with *per cell routing* considers each packet separately. The induced load is evenly divided among all possible paths between the source and the destination. The right side of Figure 21 shows all the paths between the source and destination

for the virtual circuit on the left side. There are four disjoint paths between the input and output. The links in each path have an induced load of .2. Intuitively it should be clear that per cell routing results in a more even distribution of the load across the internal links. In fact, in a Beneš network with per cell routing, the induced load on any internal link due to the collection of virtual circuits is $\leq l$. In a Beneš network with fixed path routing, the induced load on any internal link can be as large as $\approx (2\log_d n - 1)l$. The internal links must be capable of operating a rate of at least the maximum induced load.

Even if a network is configured so that the internal link data rate is sufficient to carry the maximum induced load, there may still be some packet loss at the buffers of the switch elements due to short term contention. This packet loss can be reduced by operating the internal links at a slightly faster rate than is required by considering only long term induced load.

One way to increase the internal link data rate is to increase the data path width. For example, if the external links operate at 600 Mb/s and the internal clock rate is 100 MHz, a data path width of 6 bits results in an internal link data rate of 600 Mb/s. Increasing the data path width to 8 bits provides an internal link data rate of 800 Mb/s. There is some practical limit on the width of the data path; as the data path widens, each packet must be processed more quickly. If the maximum data path width is not sufficient to meet the requirements on the internal link data rate, then the entire Beneš network can be replicated. A network consisting of r parallel Beneš networks is called a Cantor network with multiplicity r [4]. We use the term *speed advantage* to refer to the ratio of the link speed within the network to the external link speed. In reference [11] it is shown that if each Beneš network has a speed advantage of $1/\beta$, a Cantor network consisting of r such Beneš networks can handle any mix of traffic, where r satisfies

$$r \geq \frac{2\beta}{d(1-\beta)}(1 + (d-1)(\lceil \log_d n \rceil - 1)).$$

As mentioned above, the speed advantage can be increased by increasing m, the data path width. For a given value of m, the speed advantage and minimum value for r can be computed. The entire network consists of r parallel Beneš networks, each containing $2\lceil \log_d n \rceil - 1$ stages of n/d switch elements per stage. The chip count for the network, C, is then given by

$$C = rn/d(2\lceil \log_d n \rceil - 1)(m/b + 1).$$

Clearly there is a tradeoff between the values of m and r; as m increases, the minimal value for r decreases. For a particular set of network parameters it is necessary to find the value of m which minimizes C. It is practical to do this by an exhaustive search of the space of possible values for m; technology constraints will impose a reasonable upper limit for the search space.

5.3. Examples

In this section we consider several examples of buffered Beneš networks obtained by choosing from the design alternatives presented in the previous section. Each network will use a bit-sliced organization for the switch element. For each example we assume that q denotes the

speed advantage needed to reduce queueing at the switch element buffers. We assume that one bit of the data path is used for control information. Thus the minimum data path width is $m = \lceil lq/f + 1 \rceil$. In each case B will denote the number of buffer slots per input. The networks will differ in the expression for the transistor count per data slice, T_{ds}. Given a value for T_{ds}, the chip count for the network can be computed as described in the previous section.

In the first example we choose fixed path routing of packets through the network and output buffering at each switch element with bus write access. These choices are similar to those made in the Atom switch being built by NEC [12]. The transistor count per data slice, T_{ds}, is

$$T_{ds} = LdBx_1/m + 2Ldx_2/m.$$

Because fixed path routing is used, multiple Beneš networks may be needed. The optimal value of m (and consequently r) must be computed by examining the effect on the total chip count C as m is varied from the minimum value of $\lceil lq/f + 1 \rceil$ to the maximum value based on technology constraints.

Figure 22 gives the chip count for this example, where we have chosen $B = 50$ slots per output and $q = 1.25$. We have chosen $d \in \{8, 16, 32\}$ and limited the transistor count per chip to be 500,000. For a particular value of d, the plots are the same for those values of n where $\lceil \log_d n \rceil$ is the same. From the figure we see that as d increases, the number of packages per port decreases. This change is most noticeable going from $d = 8$ to $d = 16$. The packaging of the data slices is transistor limited, so increasing p from 32 to 64 has no effect on the chip count. As the link rate increases, the number of parallel Beneš networks grows, leading to a significant increase in packages per port.

In the second example we choose per-cell routing of packets and a shared buffer at each switch element with crossbar write and read access. These choices are similar to those made in the second generation of the Broadcast Packet Switch [14]. The transistor count per data slice, T_{ds}, is

$$T_{ds} = LdBx_1/m + 2Bd^2x_3.$$

A single Beneš network with per-cell routing is sufficient to meet the traffic requirements in the range we will consider.

Figure 23 gives the chip count for this second example. Here we have chosen $B = 3$ buffer slots per input and $q = 1.25$. As for Example 1, an increase in d causes the number of packages per port to decrease. Unlike Example 1, the packaging of the data slices is affected by pin count. Increasing p from 32 to 64 just about cuts in half the number of packages per port.

To examine differences between crossbar and bus access, we consider a third example which is identical to the second except that the access to the shared buffer is through a bus rather than a crossbar. Again we choose per-cell routing of packets through the network and a shared buffer at each switch element. The transistor count per data slice, T_{ds}, is

$$T_{ds} = LdBx_1/m + 4Ldx_2/m.$$

Figure 24 gives the chip count for the third example. As for Example 2, we have chosen $B = 3$ buffer slots per input. These curves have almost exactly the same shape as those for Example 2, with the exception that Example 3 has a slightly higher package count for low values of link speed. In this range, the packaging of the data slices for Example 3 is transistor constrained, leading to slightly higher chip counts. As link speed increases, pin constraints dictate the package count.

6. Network Comparison

In this section we compare the chip counts of the various networks on the same graph allowing us to reach conclusions about the relative chip complexity of the networks. Figure 25 compares the number of chip packages per input as a function of the external link data rate for all of the networks described. The number of packages is plotted on a log scale. It is important to keep in mind the log scale; it may appear that the networks have more comparable chip counts than is actually the case. The curves for Examples 1 and 2 are marked with the numbers 1 and 2. (Example 3 is not included, as the chip count is identical to Example 2 except at very low l.) The curves for the remaining networks are marked by the first letter of the corresponding network, except that the Shuffleout network is marked with an "h". So, "1" marks the Example 1 network, "2" marks the Example 2 network, "s" marks the Sunshine network, "l" marks Lee's network, "k" marks the Knockout network, "t" marks the Tandem Banyan network, "h" marks the Shuffleout network.

We have chosen three values for the number of inputs. The top row is $n = 16$, the middle row is $n = 256$ and the bottom row is $n = 4096$. These choices reflect a range of network sizes, from small to fairly large. The left column is $p = 32$ and the right column is $p = 64$. For the Benes based networks, we have chosen the value of d which minimizes the chip count for each particular value of n. In all cases the choice of $d = 16$ gave the best chip count.

We can make several observations from this figure that apply to all of the networks considered. First, as the chip dimension increases, the number of packages per port stays the same or decreases. With the exception of Example 1, all of the networks benefit greatly from an increase in chip dimension. Second, as the number of inputs increases, the number of packages per port increases, except for Lee's network. This is expected from the equations for chip count; each network has an n^2 or an $n \log n$ term in the chip count.

Considering the relative performance of the networks, we observe that the Knockout network has by far the highest chip count for $n = 256$ and $n = 4096$. Lee's network and the Shuffleout network have the highest chip counts for $n = 16$, but compare more favorably for larger n. The Sunshine network also tends to have a relatively high chip count for all three values of n. Turning to the more competitive networks, we observe that Example 2 performs well in all cases. In addition, Example 1 and the Tandem Banyan networks perform well for low n and high n respectively.

7. Conclusions

In comparing the multitude of architectures proposed for ATM switching, it is important to extend the comparison beyond the qualitative level and consider quantitative differences between architectures. Any architecture worth considering can be configured to meet performance requirements within a realistic range. This paper has attempted to get at the issue of network cost as a basis for comparison. To this end, we have compared the chip count for a variety of architectures. Not surprisingly, we found significant disparity across architectures and across various network sizes and performance requirements. While certain networks consistently had either relatively low or relatively high chip counts, in general the relative chip count depended on the network size and performance requirements.

As mentioned early in the paper, the use of multiple planes to allow the networks to keep up with the external link speed has implications for additional hardware at the inputs (to separate the packet stream into multiple planes) and at the outputs (to combine the separate planes). We have not considered the additions to chip count due to these components. An extension of this work would be to more carefully consider implementations of such components and to examine the impact on chip count when these components are included.

Another natural extension of this work would be to consider architectures for multipoint ATM switching. The multipoint environment requires that a switch be capable of copying a packet to one or more switch outputs. Storing and accessing multipoint information adds considerably to the chip complexity of a switch.

Acknowledgements

The author gratefully acknowledges many useful discussions with Jon Turner.

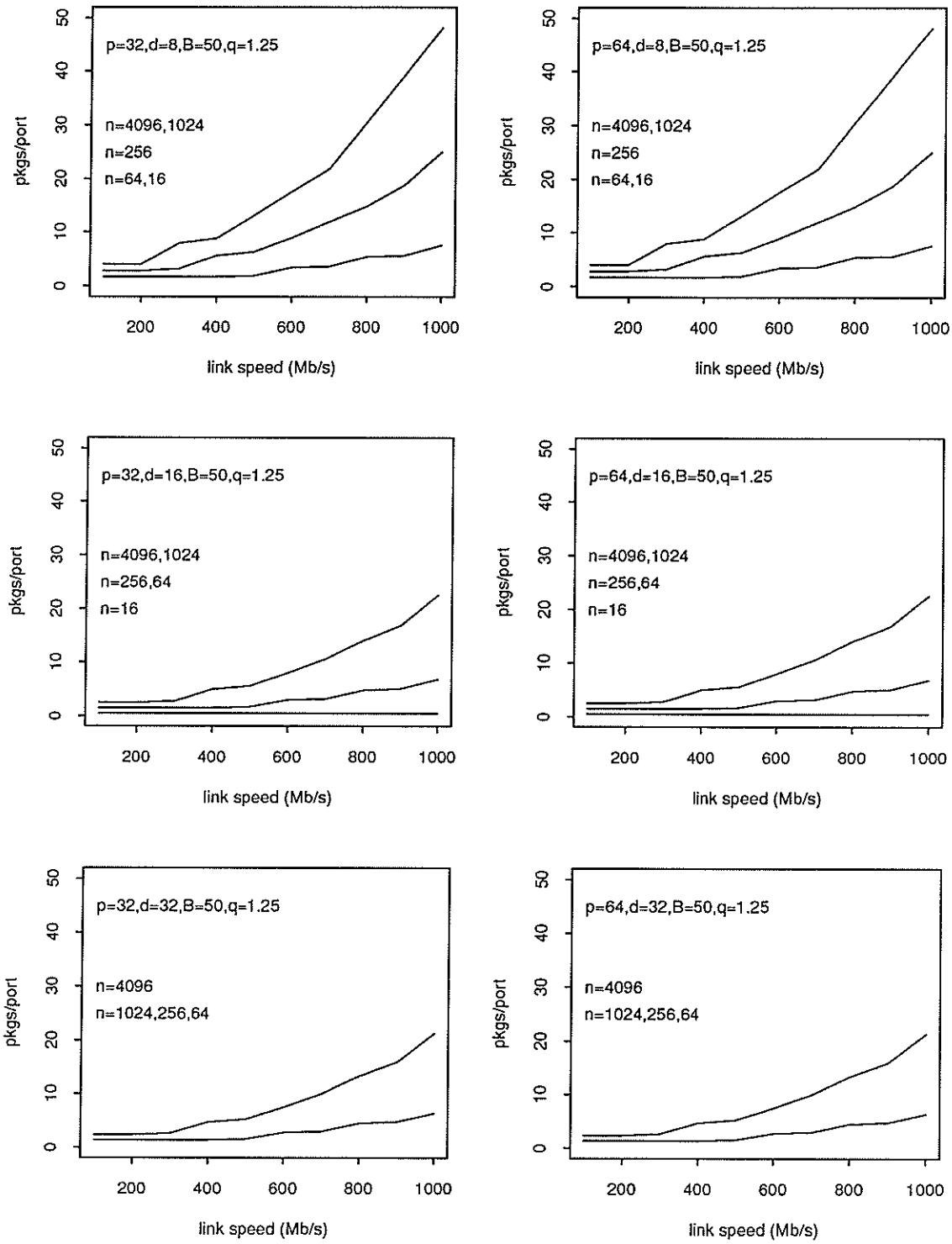

Figure 22: Example 1 Chip Count

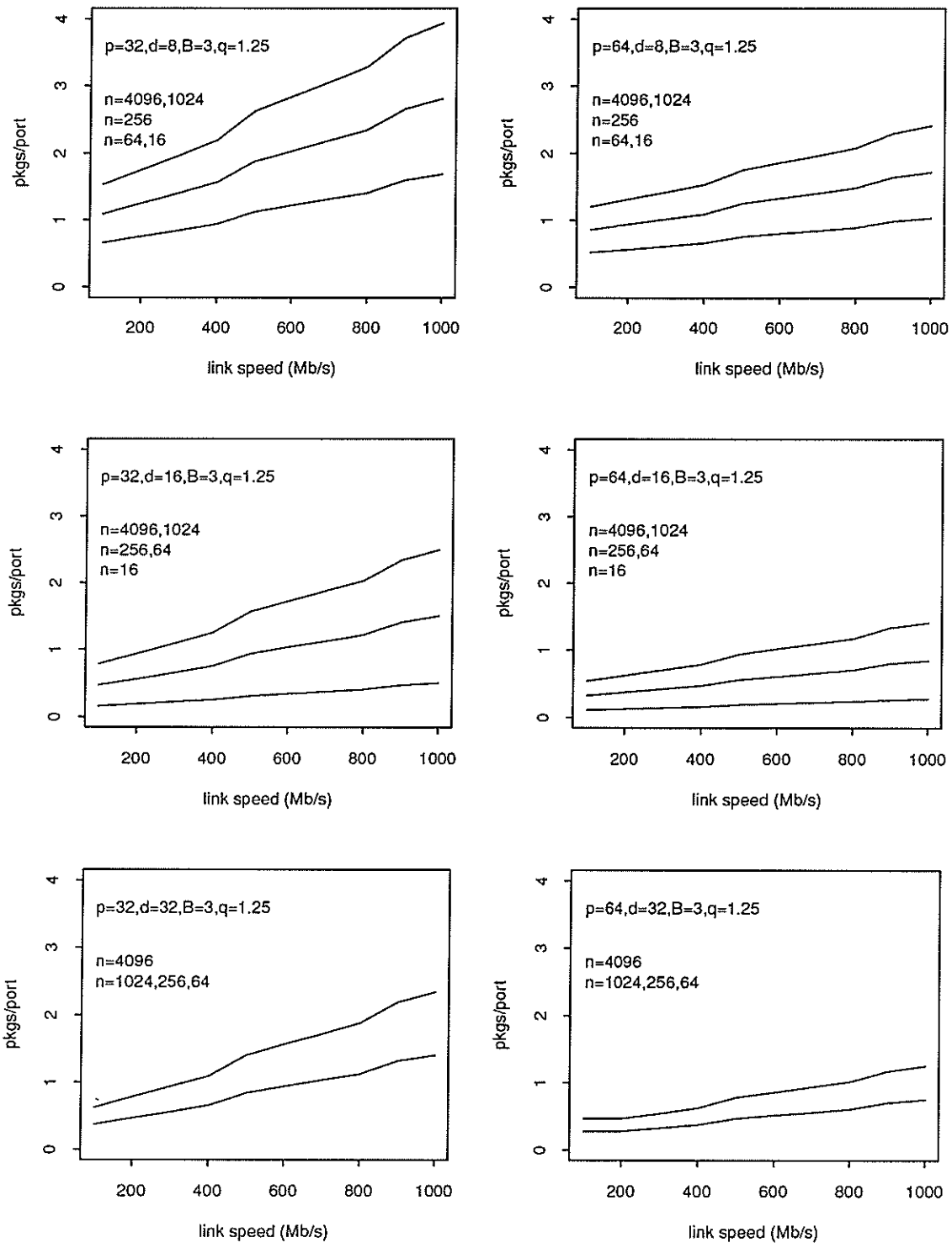

Figure 23: Example 2 Chip Count

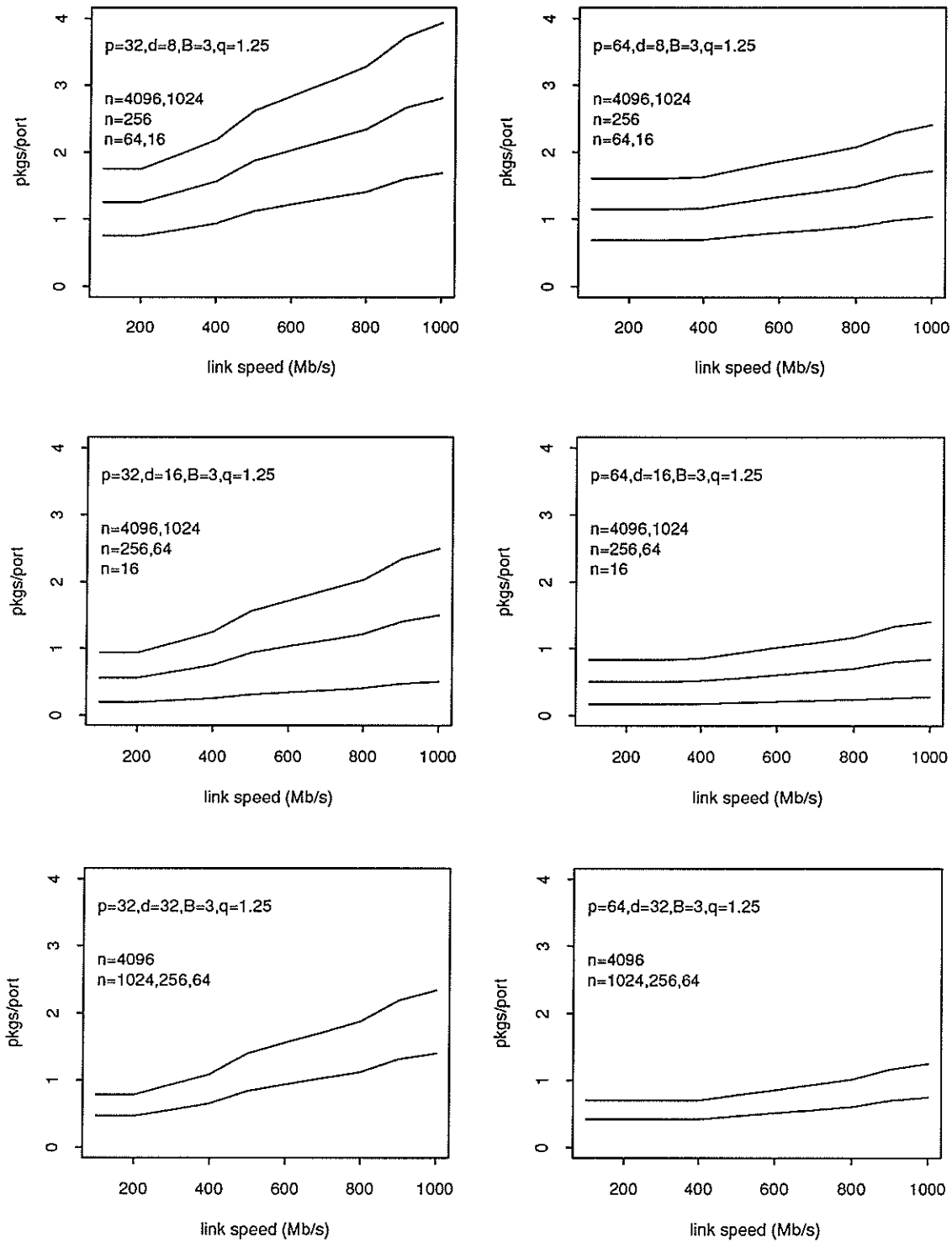

Figure 24: Example 3 Chip Count

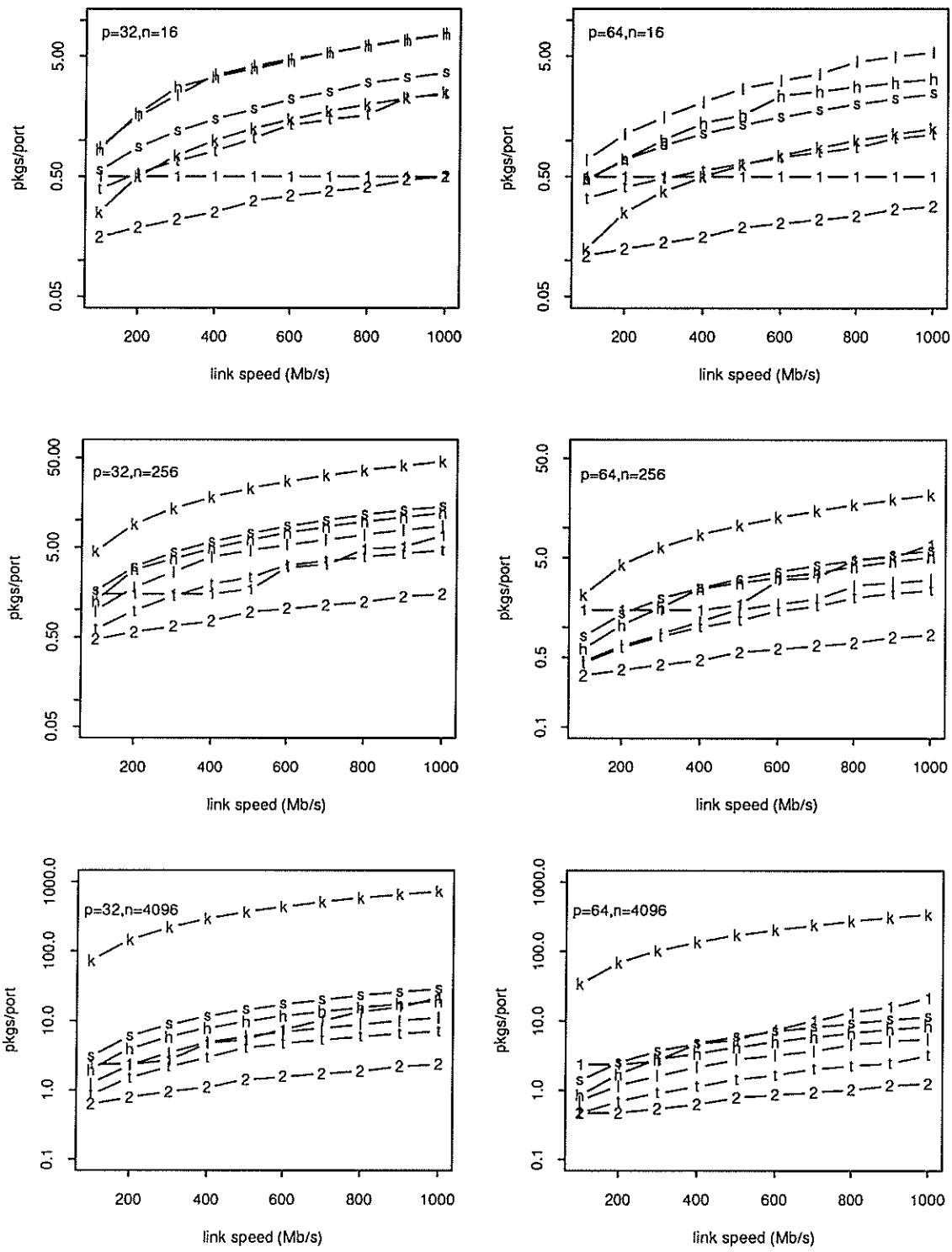

Figure 25: Network Comparison

References

[1] K. E. Batcher. Sorting networks and their applications. In *Proc. of the Spring Joint Computer Conference*, 1968.

[2] V. E. Beneš. *Mathematical Theory of Connecting Networks and Telephone Traffic.* Academic Press, 1965.

[3] V. E. Beneš. Blocking states in connecting networks made of square switches arranged in stages. *Bell Systems Tech. Journal*, 1981.

[4] D. B. Cantor. On non-blocking switching networks. *Networks*, 1971.

[5] J. P. Coudreuse and M. Servel. Prelude: An asynchronous time-division switched network. In *Proc. of the International Communications Conference*, 1987.

[6] M. De Prycker and J. Bauwens. A switching exchange for an asynchronous time division based network. In *Proc. of International Communications Conference*, 1987.

[7] M. Dècina, P. Giacomazzi, and A. Pattavina. Shuffle interconnection networks with deflection routing for ATM switching: The closed-loop shuffleout. In *Proc. of Infocom 91*, 1991.

[8] J. N. Giacopelli, W. D. Sincoskie, and M. Littlewood. Sunshine: A high performance self routing broadband packet switch architecture. In *Proc. of the International Switching Symposium*, 1990.

[9] Tony T. Lee. Non-blocking copy networks for multicast packet switching. *Bell Communications Research*, 1987.

[10] Tony T. Lee. A modular architecture for very large packet switches. *IEEE Transactions on Communications*, 1990.

[11] Riccardo Melen and Jonathan S. Turner. Nonblocking multirate networks. *SIAM Journal on Computing*, 1989.

[12] Hiroshi Suzuki, Hiroshi Nagano, Toshio Suzuki, Takao Takeuchi, and Susumu Iwasaki. Output-buffer switch architecture for asynchronous transfer mode. In *Proc. of the International Switching Symposium*, 1989.

[13] Fouad A. Tobagi and Timothy C. Kwok. The Tandem Banyan switching fabric: A simple high-performance fast packet switch. In *Proc. of Infocom 91*, 1991.

[14] Jonathan S. Turner. Design of a broadcast packet network. *IEEE Transactions on Communications*, 1988.

[15] Y. S. Yeh, M. G. Hluchyj, and A. S. Acampora. The Knockout switch: A simple modular architectur for high performance packet switching. In *Proc. of the International Switching Symposium*, 1987.