ATM SWITCH BASED INTERCONNECTION NETWORKS

Wei Wei

Department of Computer Science

M.S. Degree, October 1995

ABSTRACT

Networks of workstations are becoming a popular platform for parallel computing. Asynchronous Transfer Mode (ATM) network connections can improve the communication characteristics of these parallel clusters. Interconnection networks that are made from a large number of small ATM switches are cost effective and can be scaled up to include a large number of processors. This thesis introduces a new class of topologies for constructing multicomputers called Small ATM Switch based Interconnection (SASI) networks. They are hierarchical, symmetrical, have low diameters, scalable bisection width, and are easy to implement. An addressing schema that takes advantage of the hierarchical properties of these networks is presented. Comparisons are made between the SASIs and other topologies. A simulation shows that SASI networks can be implemented using available switch technologies. The results give these networks a performance advantage over traditional workstation clusters.

COMMITTEE APPROVAL: Mark J. Clement, Committee Chairman

J. Kelly Flanagan, Committee Member

Scott N. Woodfield, Committee Member

Tony R. Martinez, Graduate Coordinator

ATM SWITCH BASED INTERCONNECTION NETWORKS

A Thesis

Submitted to the

Department of Computer Science

Brigham Young University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Wei Wei

October 1995

This thesis by Wei Wei is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the thesis requirement for the degree of Master of Science.

Mark J. Clement, Committee Chairman

J. Kelly Flanagan, Committee Member

Scott N. Woodfield, Committee Member

Date

Tony R. Martinez, Graduate Coordinator

ACKNOWLEDGMENTS

I would like to thank Professor Mark J. Clement, for his support, guidance, encouragement, and most of all, for his patience and understanding. He has not only given invaluable technical suggestions, but also editorial comments on both this thesis and other papers. He has done all he could to make life easier for me as a international student. Without his help, I would not have completed my education and thesis in such a short time. It would have been impossible to complete this thesis. I am indebted to Professor J. Kelly Flanagan and Professor Scott N. Woodfield, who have read my thesis proposal and this thesis, for their suggestions and criticisms that motivated this research.

I thank the Department of Computer Science at Brigham Young University for the availability and use of their excellent computer facilities that made this work possible.

I also want to express my appreciation to Lamar and Darlene Berrett who gave me both spiritual and financial support.

Most importantly, I thank my lovely wife Mimi for her patience, constant moral support, and encouragement throughout my graduate career. It is to her and my son I dedicate this thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Many interconnection topologies have been proposed for building Massively Parallel Processing (MPP) networks [1, 2, 3, 4, 5, 6]. Although many of these networks have been implemented and have performed adequately, their excessive cost has made them impractical for many parallel applications. As a result, many applications are run on clusters of workstations that have traditionally communicated through a shared media Local Area Network (LAN) such as Ethernet. However, because of the fixed bandwidth available in these configurations, shared media communication networks are not practical for connecting large numbers of workstations. This research will investigate network technologies that enable cluster solutions to provide acceptable performance at a fraction of the cost of MPP machines.

In the past decade, high-performance communication networks have been developed that overcome many of the problems inherent in Ethernet based LANs. ATM networks have been proposed as a flexible interconnection media for cluster computing platforms [7, 8]. ATM switching technology is currently being used for telephony, and the hardware promises to become less expensive as telephony deployment continues [9, 10, 11, 12, 13]. Using ATM technology, Gigabit communication links are available for each node in a multicomputer, overcoming the bandwidth limitations of Ethernet-style LAN connections [14]. The cost of ATM switches, however, increases exponentially as more inputs and outputs are added to the switch. For this reason Small ATM Switch-based Interconnection (SASI) networks [15] may be more feasible for cost-sensitive cluster computing solutions. Networks that use a large number of small ATM switches (from four to sixteen ports) can be implemented so that the cost of a high-performance network increases linearly with the number of processors.

Traditional MPP networks use electrical media and generally operate at a fixed bandwidth [16]. Since optical fiber can be used as a medium for ATM networks, much higher bandwidth connections are available. The ATM protocol specifies a fix-sized, cell-based, connection-oriented protocol, which results in switching fabrics that are simpler than many MPP machines. Standard ATM data rates range from 51.84 Mbps to 2,488.32 Mbps [14]. Although the use of ATM by the telephony industry decreases the cost of network connections, the latency induced by adherence to standards will generally be greater than that found in custom MPP networks. The improvement in bandwidth and the low cost of ATM networks makes them attractive for many parallel applications that are not highly sensitive to latency.

SASI networks are very flexible. They can be used to construct Generalized Fat Trees (GFT) [17] Meshes, Pyramids, and many other topologies [16, 18].

One example a SASI network is the Round Table (RT) network [1] [15] which maximizes the usability of switch ports, but does not provide scalable bisection width. In this thesis, we introduce the Chordal Spoke (CS) network [2] , another example of SASI network. This network provides scalable bisection width and offers higher levels of fault-tolerance than the RT network.

An ATM *switch* can be viewed as a connection unit with w pairs of input and output ports. We will denote the *width* of a switch as the number of input and output port pairs. Each pair of ports has a unique *address* labeled from $0, ...,$to $w-1$. A switch with *width* w can also be thought of as a w-node completely connected graph. If processing elements are connected to each port of the switch, there is a direct, contention-free connection between each pair of nodes [7, 9]. But if several source nodes send a message to a common destination node at the same time congestion can occur. Congestion can cause increased delays or lost data in an ATM switch. In order to connect large numbers of workstations through smaller ATM switches, some of the ports will be connected to processing elements and some ports will be connected to other switches. For simplification, we assume that the input and output ports are connected to the same endpoint. Since the ATM protocol is connection-oriented, the connection must be established between two ports on a switch before data can be transmitted. A connection of two switches through their ports is a bidirectional *edge*. The chordal spoke network

[1] The round table network was developed as part of this thesis research.
[2] The chordal spoke network was developed as part of this thesis research.

is a collection of ATM switches with *internal connections* between switches and *leaf ports* connected to workstations or *leaf nodes*. A *path* is a sequence of *edges* from one *leaf node* to another *leaf node*. The *distance* of a path is the number of switches that must be traversed in the path.

1.1 Thesis Outline

The remainder of this thesis is organized into four chapters and one appendix. Chapter 2 describes the round table ATM interconnection networks that achieve the maximum of port usability. Chapter 3 presents chordal spoke ATM interconnection networks that achieve scalable bisection width and are suitable for variaous communication applications. Chapter 4 presents the simulations for both round table and chordal spoke interconnection networks. Implementation of common communication models such as point to point, broadcast, and reduction have been built including addressing and signaling. Chapter 5 contains conclusions and possible future research opportunities. Appendix A appends the main programs and their function names of simulation.

CHAPTER 2

THE ROUND TABLE ATM INTERCONNECTION NETWORK

2.1 Definitions and Notations

A *switch* is a connection unit with w pairs of input and output ports. Each input port and output port pair has a unique *address* from $0, 1, ..., w-1$. A switch with *width* w can also be thought of as a w-node completely connected graph. If processing elements were connected to each port of the switch, there would be a direct, contention free connection between each pair of nodes. But if several source nodes send a message to a common destination node at the same time congestion can occur. Congestion can cause increased delay or lost data in an ATM switch.

In order to connect large numbers of workstations through smaller ATM switches, some of the ports will be connected to processing elements and some ports will connect switches. For simplification, we say a switch has w ports instead of w pairs of input and output ports. Since the ATM protocol is connection oriented, a connection must be established between two ports on a switch before data can be transmitted. A connection of two switches through their ports is called an *edge*.

It is the shortest or unit *distance* between two switches.

A SASI network is a collection of ATM switches with *internal connections* between switches and *leaf ports* connected to workstations or *leaf nodes.* A *path* is a sequence of *edges* from one *leaf node* to another *leaf node.* The *distance* of a path is the number of switches that must be traversed in the path.

2.2 Round Table Networks

A *Round Table* network RT(w, r) is a hierarchical network that has many appealing properties. It can be characterized in the following way:

- A RT$(w, 1)$ is an ATM switch. It is the *atomic center* of its w ports. The *addresses* of leaf ports in RT$(w, 1)$ are 0, 1, ..., w-1.

- A RT(w, r) with $r > 1$ is a network which uses an RT$(w, r-1)$ as its *center* with a *perimeter* switch connected at each leaf port of RT$(w, r-1)$. The *address* of each leaf port of RT(w, r) is the *perimeter* switch's port address followed by the leaf port address in the RT$(w, r-1)$ where the *perimeter* switch is connected.

  For instance, a RT(4, 1) has leaf ports 00, 01, 10 and 11 and a RT(4, 2) can be built by adding four *perimeter* switches to RT(4, 1)'s four ports. The RT(4,2) has 12 leaf ports shown in Figure 2.1.

The *radius* r of RT(w, r) is defined as the number of switches between a leaf port and the RT(w, r) network's *atomic center.* For RT$(w, 1)$, the radius is

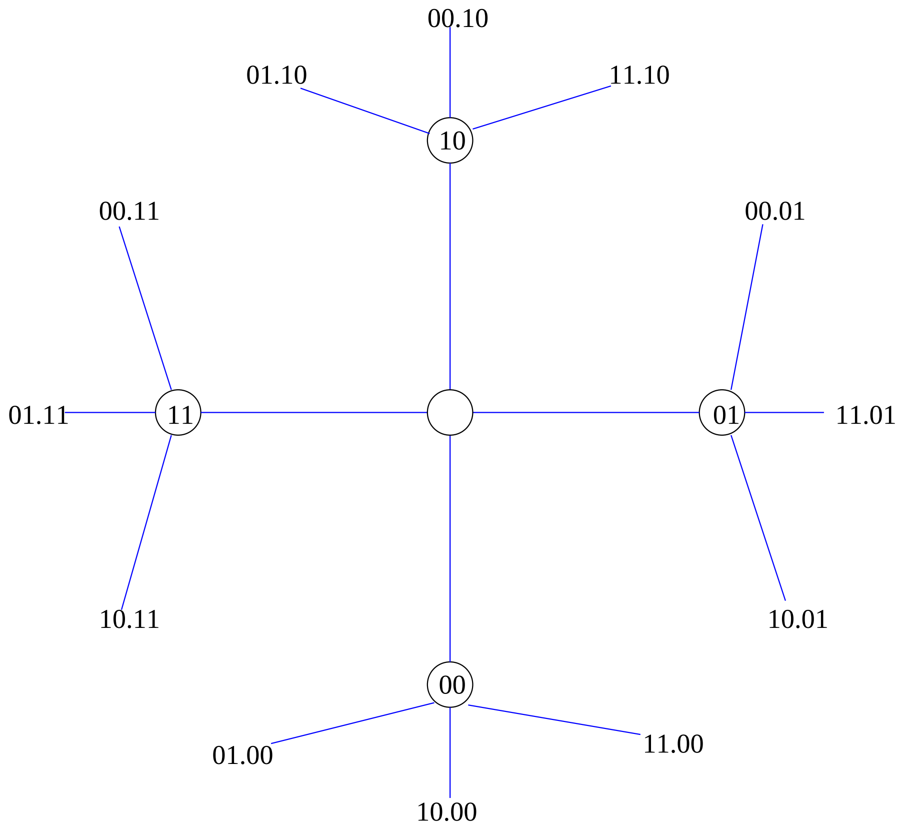


Figure 2.1: An example of RT (4, 2)

1. The number of leaf ports in RT(w, r) is denoted by nlp(w, r) and the number of switches is denoted by ns(w, r).

2.3 Topological Properties

The Round Table Network RT(w, r) is symmetrical about the atomic center. This symmetry simplifies routing and node addressing and makes the RT easy to scale up to large numbers of processors. The distance from each leaf node to the atomic center is the same, resulting in predictable communication latency between nodes. There exists only one path between any two leaf nodes in a RT.

Because of this property, and the fact that there is a path between every pair of leaf nodes, RT inherits the desirable properties of Banyan networks[2]. The *diameter* – the longest distance between two leaf nodes – of RT(w, r) is $2r - 1$ since the distance from a leaf node to the atomic center is r and the distance from the atomic center to another leaf node is r so there exist $2r - 1$ switches in the path.

The RT(w, r)'s atomic center has w leaf ports which are connected to w perimeter switches. These w perimeter switches each have $w - 1$ leaf ports which are connected to $w(w-1)$ perimeter switches. The number of leaf ports for RT(w, r) is

Theorem 2.3.1

$$nlp(w, r) = (w - 1)nlp(w, r - 1) = w(w - 1)^{r-1}$$

Proof:

- The formula obviously holds for $r = 1$:

  $nlp(w, 1) = w$

- We make the induction hypothesis that $nlp(w, r) = w(w - 1)^{r-1}$ for some positive integer $r > 1$.

  This implies

$$nlp(w, r + 1) \quad = \quad (w - 1)nlp(w, r)$$

$$
\begin{aligned}
&= (w-1)w(w-1)^{r-1} \\
&= w(w-1)^r
\end{aligned}
$$

□

A RT(w,1) consists of a single ATM switch connected to leaf nodes (ns(w, 1) = 1). As the network grows, a perimeter switch will be added to each leaf port. The generalized number of switches in a RT(w, r) is

Theorem 2.3.2

$$
\begin{aligned}
ns(w,r) &= ns(w,r-1) + nlp(w,r-1) \\
&= \begin{cases} 1 & when\ r = 1 \\ 1 + \sum_{i=0}^{r-2} w(w-1)^i & when\ r > 1 \end{cases} \\
&= 1 + w\left(\frac{(w-1)^{r-1}-1}{w-2}\right)
\end{aligned}
$$

Proof:

- The formula obviously holds for $r = 1$:

  $ns(w,1) = 1$

- We make the induction hypothesis that

  $ns(w,r) = 1 + \sum_{i=0}^{r-2} w(w-1)^i$

  for some positive integer $r > 1$.

  This implies

$$
ns(w,r+1) = ns(w,r) + nlp(w,r)
$$

$$
\begin{aligned}
&= 1 + \sum_{i=0}^{r-2} w(w-1)^i + w(w-1)^{r-1} \\
&= 1 + \sum_{i=0}^{r-1} w(w-1)^i \\
&= 1 + w\left(\frac{(w-1)^r - 1}{w-2}\right)
\end{aligned}
$$

$\square$

The average distance between nodes in a RT(w, r) is shorter than the diameter since some paths do not go through the atomic center. One of the advantages of an RT is that one percentage of w of the paths are shorter than the diameter.

Theorem 2.3.3

$$al(w,r) = \frac{(2r-1)(w-1)^r + (w-2)\sum_{i=0}^{r-2}(2i+1)(w-1)^i}{w(w-1)^{r-1} - 1}$$

for $r \geq 2$ *and* $w \geq 3$

Proof:

- The formula holds for $r = 2$:

$$al(w,2) = \frac{1(w-2) + 3(w-1)(w-1)}{w-2+(w-1)(w-1)} = \frac{3(w-1)^2 + w - 2}{w(w-1) - 1}$$

- We make the induction hypothesis that

$$al(w,r) = \frac{(2r-1)(w-1)^r + (w-2)\sum_{i=0}^{r-2}(2i+1)(w-1)^i}{w(w-1)^{r-1} - 1}$$

for some positive integer $r > 2$.

This implies

$$
\begin{aligned}
al(w, r+1) &= \frac{1}{nlp(w, r+1) - 1} \cdot \{(2r - 1 + 2)(w-1)^r (w-1) \\
&\quad + (w-2) \sum_{i=0}^{r-2} (2i+1)(w-1)^i \\
&\quad + (w-2)(2r-1)(w-1)^{r-1}\} \\
&= \frac{1}{w(w-1)^{(r+1)-1} - 1} \cdot \{(2(r+1) - 1)(w-1)^{r+1} \\
&\quad + (w-2) \sum_{i=0}^{(r+1)-2} (2i+1)(w-1)^i\}
\end{aligned}
$$

□

For a given network, the *bisection width* is the minimum number of connections removed in order to divide the network into two equal halves. Since the atomic center of a RT must be cut in order to divide the network into two equal halves, the bisection width of the RT(w, r) is equal to the bisection width of the atomic center. For an ATM switch with w ports, the bisection width is

Theorem 2.3.4

$$bw(w, r) = (\frac{w}{2})^2$$

Proof: An ATM switch can logically be viewed as a w-node fully connected graph. The number of edges for a w-node fully connected graph is $w(w-1)/2$. The bisection of a w-node fully connected graph will result in two $(w/2)$-node fully connected subgraphs. The number of edges that must be removed is equal to the difference between the number of edges in a w-node fully connected graph and the number of edges in the two $(w/2)$-node fully connected graphs

$$\begin{aligned} bw(w,r) &= \frac{w(w-1)}{2} - 2\frac{\frac{w}{2}(\frac{w}{2}-1)}{2} = \frac{w(w-1)}{2} - \frac{w}{2}(\frac{w}{2}-1) \\ &= \frac{w}{2}(w-1-\frac{w}{2}+1) = \frac{w}{2}(\frac{w}{2}) = (\frac{w}{2})^2 \end{aligned}$$

□

Higher bandwidth ATM connections at the atomic center will result in larger bisectional bandwidth for the RT. Large radius RT may require higher bandwidth ATM switches for applications which are sensitive to bisectional bandwidth.

The *leaf port efficiency* in a RT can be determined by dividing the number of leaf nodes by the total number of ports available on switches in the network. RT achieves maximum efficiency because if any connection between switches is removed, the graph is disconnected.

$$effp(w,r) = \frac{nlp(w,r)}{w \cdot ns(w,r)} = \frac{(w-1)^{r-1}}{1 + w(\frac{(w-1)^{r-1}-1}{w-2})}$$

The *bandwidth efficiency* in a RT can be determined by dividing bisection width by the number of leaf ports in the network. It shows the worst case bandwidth sharing.

$$effb(w,r) = \frac{bw(w,r)}{nlp(w,r)} = \frac{w}{4(w-1)^{r-1}}$$

Table 2.1: Topological properties for RT($4,r$)

radius	diameter	bw	ns	nlp	effp	effb	al
1	1	4	1	4	1.0	1.0	1.0
2	3	4	5	12	0.6	0.333333	2.545455
3	5	4	17	36	0.529412	0.111111	4.142857
4	7	4	53	108	0.509434	0.037037	5.813084
5	9	4	161	324	0.503106	0.012346	7.526316
6	11	4	485	972	0.501031	0.004115	9.260556

Table 2.2: Topological Properties for RT(16,r)

radius	diameter	bw	ns	nlp	effp	effb	al
1	1	64	1	16	1.0	4.0	1.0
2	3	64	17	240	0.882353	0.266667	2.828452
3	5	64	257	3600	0.875486	0.017778	4.701584
4	7	64	3857	54000	0.875032	0.001185	6.584307

Table 2.1 and Table 2.2 show the topological properties of two RTs. The efficiency for RT($4,r$) is lower than RT($16,r$), but the network scales up more gradually. With 16 port ATM switches, a radius of 3 is big enough to connect thousands of processors.

The leaf port efficiency is affected little by radius but strongly by switch width. The wider the switch width is, the higher the efficiency is. The results are shown in the left part of Figure 2.2.

The bandwidth efficiency is affected by switch width and strongly by radius. The bigger the radius is, the lower the efficiency is. The results are shown in the right part of Figure 2.2.

The low bandwidth efficiency means that in the leaf level of RT, the $w-1$ leaf ports will share the bandwidth of one port of the leaf level switch.

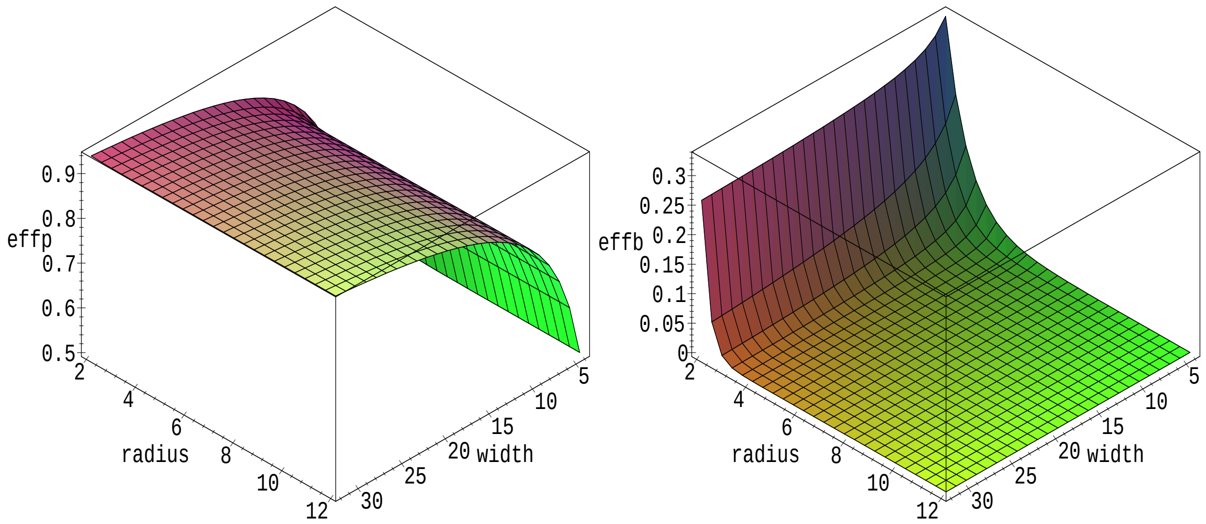


Figure 2.2: Leaf Port Efficiency and Bandwidth efficiency

Correspondingly, the leaf level switches will share the bandwidth of one port on the center switch. So the leaf or near leaf level switches will not effectively utilize their bandwidth. One way to overcome this problem is to use high bandwidth switches (OC-12 or higher) in centers and use low bandwidth switches (OC-3 or lower) in leaf or near leaf regions. Figure 2.3 is an example of how to balance the bandwidth efficiency.

Table 2.3 compares metrics of interest for Hypercubes, Generalized Fat Trees (GFT), Extended Generalized Fat Trees (XGFT) and Round Table Networks. These topologies were chosen for the following reasons. The Hypercube has been used extensively in previous research and many algorithms can be mapped to this topology. Although Generalized Fat Trees have not been implemented, in

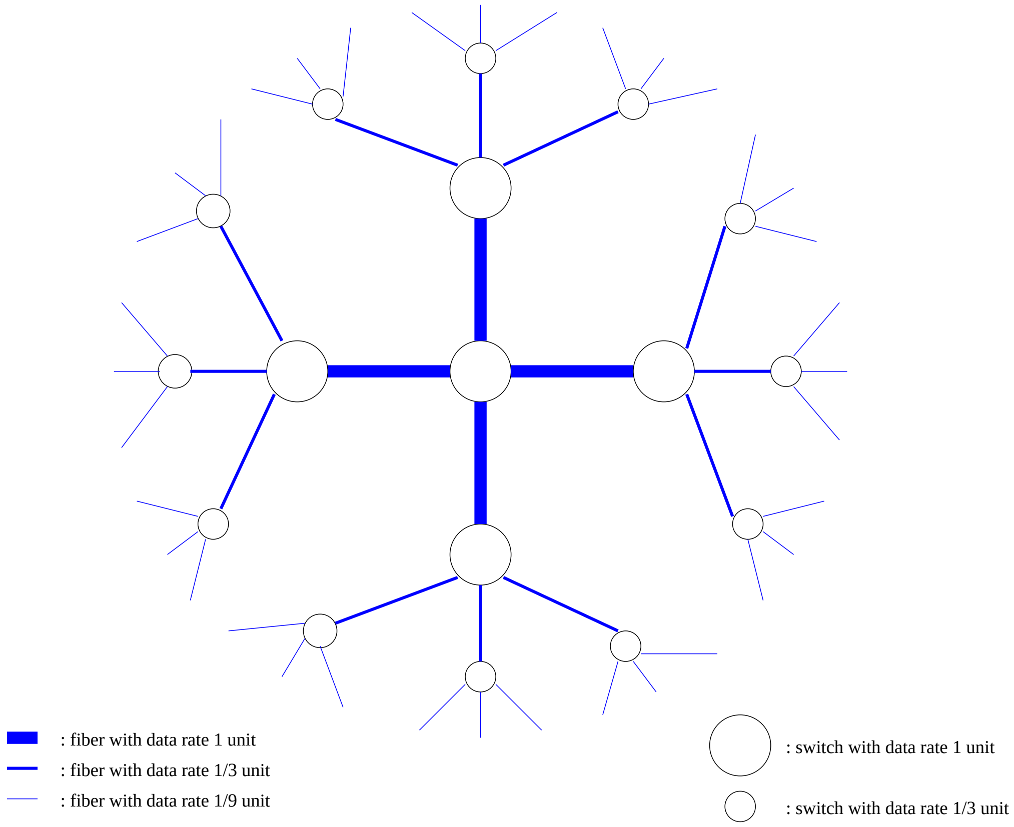


Figure 2.3: Switches with different bandwidth to achieve bandwidth efficiency

theory, they compare favorably with many popular interconnection networks. The topology used in the Connection Machine CM-5 is a special case of the Extended Generalized Fat Tree[19, 17].

Table 2.3 shows that RT(w, r) has less leaf nodes than GFT(r,w,p), but far more than that found in a Hypercube with the same diameter. The RT has degree of one while Hypercube has degree 2r and GFT has degree p (number of parent edges). This allows for economical connection of processing elements on RT, but limits the fault tolerance and bisection width of RT. The *cost*, which is the product of the diameter and degree [17] is lower for RT than Hypercubes or

Table 2.3: RT(w,r) compared with other topologies

Comparison of topological properties					
Network	nlp	dmt	d	bw	Cost
Hypercube Q(2r)	4^r	$2r-1$	$2r$	2^{2r-1}	$2r(2r-1)$
Fat Trees GFT(w,r,p)	w^r	$2r-1$	p	$\frac{w \cdot p^r}{2}$	$(2r-1)p$
XGFT(r;$w_1, ..., w_r; p_1, ..., p_r$)	$\prod_{i=1}^{r} w_i$	$2r-1$	p_1	$\frac{w_r \cdot \prod_{i=1}^{r} p_i}{2}$	$(2r-1)p_1$
Round Table RT(w,r)	$w(w-1)^{r-1}$	$2r-1$	1	$(\frac{w}{2})^2$	$2r-1$

Table 2.4: RT(16,4) compared with other topologies

Comparison of topological properties					
Network	nlp	diameter	Degree	bw	Cost
Hypercube $Q(2 \cdot 4)$	256	7	8	128	56
Fat Trees $GFT(16, 4, p)$	65536	7	p	$8 \cdot p^4$	$7 \cdot p$
Round Table $RT(16, 4)$	54000	7	1	64	7

GFTs. Both the Hypercube and GFT have higher bisection width and leaf nodes than RT, but the RT(w,r) is equivalent or better than GFT(r,w,1) in diameter, degree and cost and has a higher bisection width ($(\frac{w}{2})^2$ instead of $\frac{w}{2}$). The RT(16,4) compares favorably with other topologies. See Table 2.4.

2.4 Communications

The routing scheme for RT follows from the Identical Address Connection Method. Port number 00 in the atomic center is connected to port 00 in one of the perimeter switches. Similarly, additional ports on interior switches are connected to identical port numbers in next level switches. Node addresses are a concatenation of the leaf port address followed by the intermediate switch port numbers between the leaf and the atomic center. Using this connection method and addressing scheme for nodes, routing from one leaf node to another is simple. Figure 2.4 is an example of an RT(4,3) network.

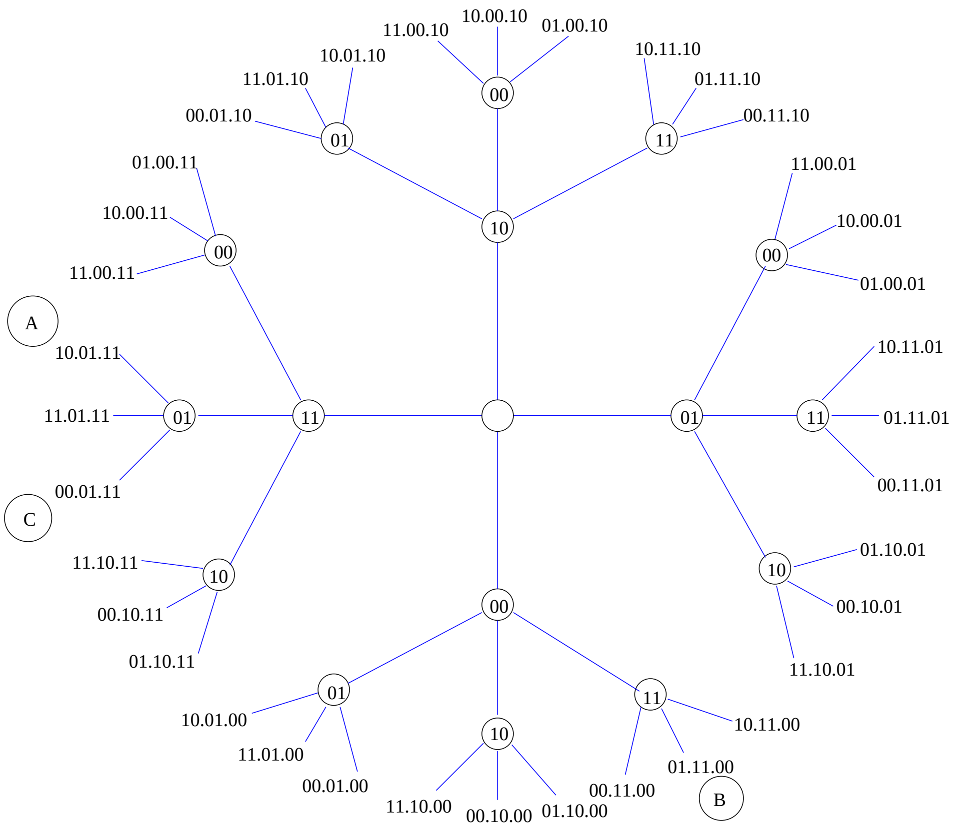


Figure 2.4: An example of RT (4, 3)

A path from leaf node A (which has address $10 \cdot 01 \cdot 11$) to leaf node B (which has address $01 \cdot 11 \cdot 00$) can be obtained by combining the source node's address with the reverse of the destination node's address. The path would be $10 \cdot 01 \cdot 11 \cdot 00 \cdot 11 \cdot 01$. The length of the path is 5. This path can be used by the signaling protocol to set up a connection through the ATM network.

The path from leaf node A (which has address $10 \cdot 01 \cdot 11$) to leaf node C (which has address $00 \cdot 01 \cdot 11$) can be shortened to $10 \cdot 00$ since the two leaf node have the same RT(4, 2) *center* address $01 \cdot 11$. The length of the path is 1.

Broadcasting can be performed easily in RT using the multicast facilities provided by many ATM switch vendors. Since a RT(w,r) is a spanning tree, a multicast message sent through the atomic center will propagate to all leaf nodes without inducing congestion.

2.5 Conclusion

round table SASI networks maximize switch utilization efficiency and have many other desirable properties. RT is hierarchical, symmetrical, has a low diameter and is easy to implement.

Compared with generalized fat trees, RTs have the same diameter, average distance and lower cost metric. Due to the high bandwidth and multicasting ability of ATM switches, reasonable bisection bandwidth and broadcasting capabilities can be achieved using RTs. The structure of RTs also provides for simple and extensible routing schemes.

CHAPTER 3

THE CHORDAL SPOKE ATM INTERCONNECTION NETWORK

3.1 Chordal Spoke Networks

A *Chordal Spoke* network CS(w, r, i) is a hierarchical network with switch *width* w, *radius* r, and *inside width* i. It can be characterized in the following way:

- A CS($w, 1, i$) is a network that consists of $w + 1$ ATM switches with *width* w: One switch (*atomic center*) connects to w perimeter switches in a star topology [20]. The perimeter switches (*leaf switches*) have chordal ring connections [21]. The *radius* is defined as the number of edges from the atomic center to the leaf switches. The *inside* width is defined as the number of ports on each leaf switch; these ports are used to connect to other leaf switches and the center. Inside width values are restricted to odd values since one port will be connected to the atomic center and pairs of ports are used to create the chordal connections ($i = 3, 5, 7, ..., w - 1$). Larger values for the inside width are used to create networks with higher bisection width. We define the *outside width* as the number of ports for each leaf switch that can be connected to workstations or to outer layers of the hierarchical switch. It

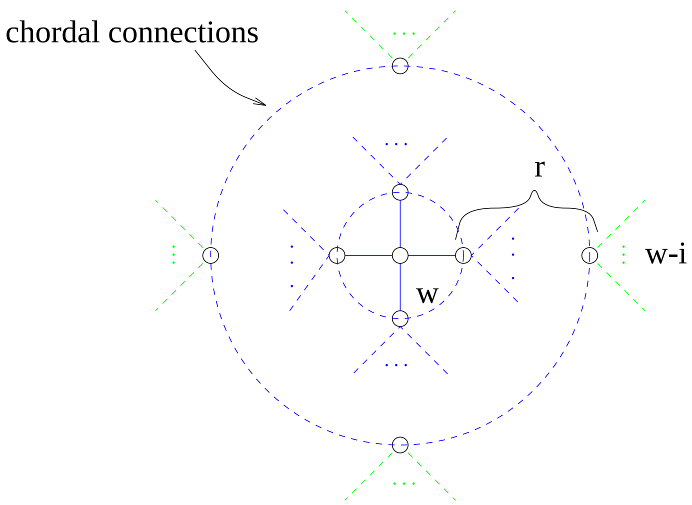


Figure 3.1: An example of CS(w, r, i).

is the difference between switch width and inside width. The *addresses* of leaf ports in a CS$(w, 1, i)$ network are the leaf switch's outside port number appended to the corresponding center's port number.

- A CS(w, r, i) with $r > 1$ is a network that uses a CS$(w, r-1, i)$ as its *center*, with a *perimeter* switch connected at each leaf port of CS$(w, r-1, i)$. These perimeter switches are connected as a chordal ring. Figure 3.1 shows the general chordal spoke networks.

When the chordal ring connections of a CS(w, r, i) are removed, the network becomes an acyclic graph. We call this the *natural spanning tree* of the chordal spoke network. Figure 3.2 (a) is an example of a CS(8,2,5) network, and figure 3.2 (b) is its spanning tree.

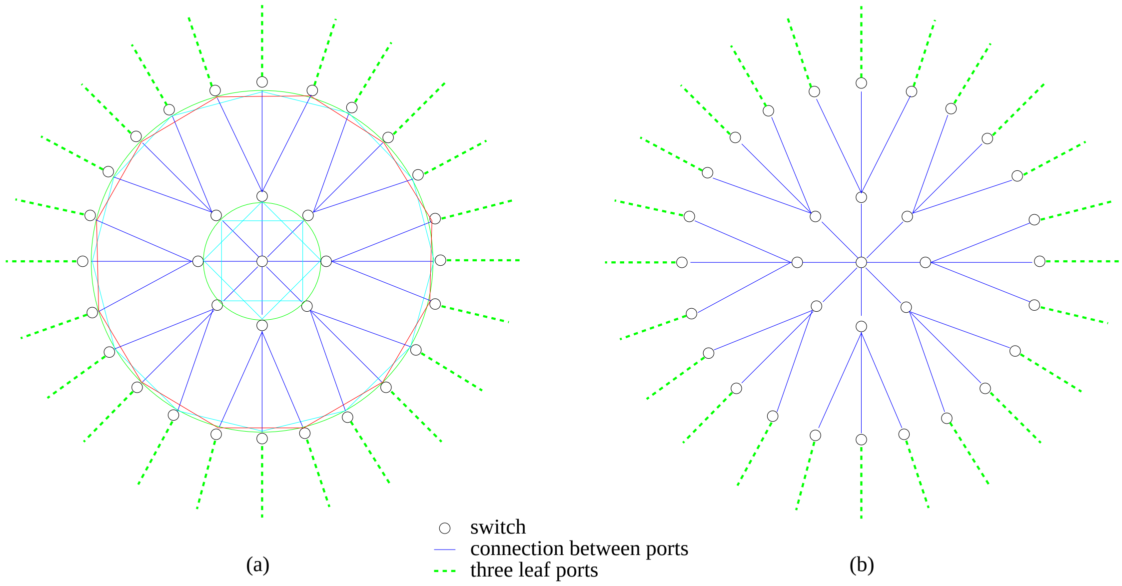


Figure 3.2: A CS(8, 2, 5) (a) and its spanning tree (b)

3.1.1 Topological Properties

The chordal spoke network CS(w, r, i) is symmetrical about the atomic center. This symmetry simplifies routing and node addressing and makes the CS network easy to scale up to large numbers of processors. The distance from each leaf node to the atomic center is the same, resulting in predictable communication latency between nodes. The *diameter* (the longest distance between two leaf nodes) of CS(w, r, i) is $2r + 1$. The distance from a leaf node to the atomic center is r and the distance from the atomic center to another leaf node is r, so there are $2r + 1$ switches in the path.

The atomic center for CS(w, r, i) has w leaf ports, which are connected to w perimeter switches. These w perimeter switches have both inside width ports connected to each other and outside width leaf ports connected to next-level perimeter switches. The *number of leaf ports* for CS(w, r, i) is:

Theorem 3.1.1

$$\begin{aligned} nlp(w,r,i) &= w \cdot (w - i_1) \cdot (w - i_2) \cdots (w - i_r) \\ &= w \prod_{j=1}^{r} (w - i_j) \end{aligned}$$

$i_j = 1, 3, 5, ..., w - 1$

Proof:

- The formula obviously holds for $r = 1$:

  $nlp(w, 1, i) = w(w - i)$

- We make the induction hypothesis that $nlp(w, r, i) = w \prod_{j=1}^{r} (w - i_j)$ for some positive integer r.

  This implies

$$\begin{aligned} nlp(w, r+1) &= (w - i_{r+1}) nlp(w, r, i) \\ &= (w - i_{r+1}) w \prod_{j=1}^{r} (w - i_j) \\ &= w \prod_{j=1}^{r+1} (w - i_j) \end{aligned}$$

□

From Theorem 3.1 we know that the number of leaf ports is dependent on the switch width w and inside width i_j at each level. The wider the switch width is, the greater the number of leaf ports is. The smaller the inside width at each level, the greater the number of leaf ports.

A CS(w,1,i) consists of a single ATM switch connected to its w perimeter switches. As the network grows, a next level perimeter switch will be added to each leaf port. The generalized *number of switches* in a CS(w, r, i) is

Theorem 3.1.2

$$ns(w,r,i) = 1 + w + \sum_{j=1}^{r-1} w \prod_{k=1}^{j} (w - i_k)$$
$$i_j = 1, 3, 5, ..., w-1$$

Proof:

- The formula obviously holds for $r = 1$ with $ns(w, 1, i) = 1 + w$
- The induction hypothesis:

  $ns(w,r,i) = 1 + w + \sum_{j=1}^{r-1} w \prod_{k=1}^{j} (w - i_k)$ for some positive integer r.

  This implies

$$\begin{aligned} ns(w, r+1, i) &= ns(w,r,i) + nlp(w,r,i) \\ &= 1 + w + \sum_{j=1}^{r-1} w \prod_{k=1}^{j} (w - i_k) + w \prod_{j=1}^{r} (w - i_j) \\ &= 1 + w + \sum_{j=1}^{r} w \prod_{k=1}^{j} (w - i_k) \end{aligned}$$

□

For a given network, the *bisection width* is the minimum number of connections that must be removed in order to divide the network into two equal halves. Since the atomic center of a CS must be cut in order to divide the network into two equal halves, the *bisection width* of the CS(w, r, i) is equal to the bisection width of the atomic center added to the number of chordal connections that must be severed. For an ATM switch with w ports, the *bisection width* is

Theorem 3.1.3

$$bw(w,r,i) = (\frac{w}{2})^2 + 2\sum_{k=1}^{r}\sum_{j=1}^{\frac{i_k-1}{2}} j$$

$$i_j = 1, 3, 5, ..., w-1$$

Proof:

An ATM switch can logically be viewed as a w-node fully connected graph. The number of edges for a w-node fully connected graph is $w(w-1)/2$. The bisection of a w-node fully connected graph will result in two $(w/2)$-node fully connected subgraphs. The number of edges that must be removed is equal to the difference between the number of edges in a w-node fully connected graph and the number of edges in the two $(w/2)$-node fully connected graphs.

$$\frac{w(w-1)}{2} - 2\frac{\frac{w}{2}(\frac{w}{2}-1)}{2} = \frac{w(w-1)}{2} - \frac{w}{2}(\frac{w}{2}-1)$$

$$
\begin{aligned}
&= \frac{w}{2}(w-1-\frac{w}{2}+1) = \frac{w}{2}(\frac{w}{2}) \\
&= (\frac{w}{2})^2
\end{aligned}
$$

The bisection width of CS(w, r, i) is equal to the bisection width of the atomic center switch plus the number of chordal ring connections that must be removed to bisect the network.

- The formula holds for $r = 1$:

  $bw(w, 1, i) = (\frac{w}{2})^2 + 2\sum_{j=1}^{\frac{i-1}{2}} j$

- We make the induction hypothesis that $bw(w, r, i) = (\frac{w}{2})^2 + 2\sum_{k=1}^{r}\sum_{j=1}^{\frac{i_k-1}{2}} j$ for some positive integer r.

  This implies

$$
\begin{aligned}
bw(w, r+1, i) &= (\frac{w}{2})^2 + 2\sum_{k=1}^{r}\sum_{j=1}^{\frac{i_k-1}{2}} j + 2\sum_{j=1}^{\frac{i_{r+1}-1}{2}} j \\
&= (\frac{w}{2})^2 + 2\sum_{k=1}^{r+1}\sum_{j=1}^{\frac{i_k-1}{2}} j
\end{aligned}
$$

□

Higher bandwidth ATM connections at the atomic center will result in a larger bisectional bandwidth for the CS networks. Larger values for inside width at each level will also result in greater bisection width. But these large values will limit the number of processors that can be connected to the network. In network structure design, high bisection width and a high number of leaf ports are both important goals. But the maximum bisection width cannot be achieved in the same network with the maximum number of leaf ports.

The *leaf port efficiency* in a CS network can be determined by dividing the number of leaf nodes by the total number of ports available on switches in the network.

$$effp(w, r, i) = \frac{nlp(w, r, i)}{w \cdot ns(w, r, i)}$$

The *efficiency of switch bandwidth utilization* in a CS network can be determined by dividing bisection width by half the number of leaf ports in the network. This metric shows the bandwidth available to each processor when half of the network tries to communicate with the other half. It represents the worst-case bandwidth availability for the network.

$$effb(w, i, r) = \frac{bw(w, i, r)}{\frac{nlp(w,i,r)}{2}} = 2\frac{bw(w, i, r)}{nlp(w, i, r)}$$

Optimization of Inside Width

To balance the bisection width with the reduction [22] in leaf ports caused by varying values for the inside width, our analysis will begin with a single switch. Figure 3.3 shows a switch with width 8 and its different inside and outside width configurations.

In figure 3.3(a) the inside width is five and the outside width is three. If all the inside links communicate in the (**o**) direction at the same time, congestion occurs. On the other hand, when all the outside ports want to communicate in the (**i**) direction at the same time, the excess inside paths are available (congestion can occur only when two or more outside ports communicate through the same inside link). Figure 3.3(a) shows that communication in the (**o**) direction has a greater chance of experiencing congestion compared with the (**i**) direction. Assuming randomly distributed communications, the level of congestion is dependent on the ratio of inside and outside widths and the amount of communications in both the (**i**) and (**o**) directions. Figure 3.3(b) shows the opposite results. When both (**i**) and (**o**) directions have the same communication demands, the minimum congestion will occur when the inside width is equal to the outside width (See figure 3.3(c)).

Congestion can occur even when excess ports are available if a single link is chosen for multiple communications. But since the ATM switches use leaky bucket algorithms to smooth the communication stream, low levels of congestion will merely result in increased delay. The amount of effort that can be justified in eliminating congestion will depend on the communication patterns for applications that will be run on the cluster.

The inside width in the CS network must be an odd number, so the inside and outside widths can be equal only when the switch width is $w = 2(2k+1)$ for some integer k. When the inside and outside widths are not equal, the ratio

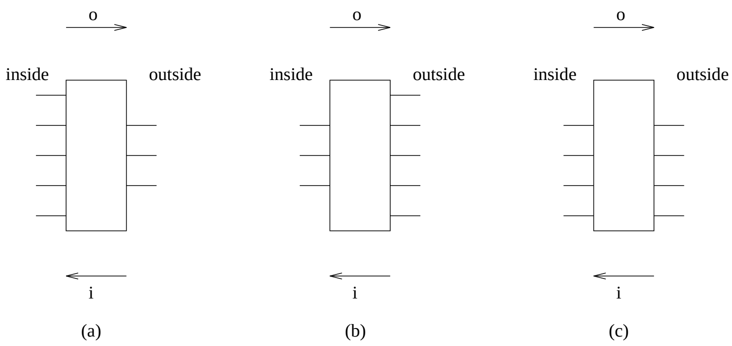


Figure 3.3: Three configurations of inside and outside widths for a switch with width 8

of (**i**) and (**o**) direction communication demands will decide the inside and outside widths. Broadcasts are an (**o**) type communication since a single message injected into the network will result in nlp(w,i,r) (**o**) type communications. Reductions are an (**i**) type communication, and point-to-point communications are neutral. If the ratio of broadcasts to reductions is known, then the network can be configured with the same ratio for outside width over inside width. Otherwise we will assume that a near-equal ratio between inside and outside widths will be best on average. In this case the inside width will be:

$$i = \begin{cases} \frac{w}{2} & \text{when } \frac{w}{2} \text{ is odd} \\ \frac{w}{2} \pm 1 & \text{otherwise} \end{cases}$$

When $\frac{w}{2}$ is even, the inside width $i = \frac{w}{2} + 1$ will be larger and the resulting network will have a greater bisection width. If we assume a constant inside width for all levels of the network, the parameters of the network can be simplified as follows:

$$nlp(w,r,i) \;=\; w(w-i)^r$$

$$ns(w,r,i) \;=\; 1 + w \cdot \frac{(w-i)^r - 1}{w-i-1}$$

$$bw(w,r,i) \;=\; (\frac{w}{2})^2 + \frac{r}{4}(i^2-1)$$

$$effp(w,r,i) \;=\; \frac{(w-i)^r}{1 + w \cdot \frac{(w-i)^r-1}{w-i-1}}$$

$$effb(w,r,i) \;=\; \frac{2((\frac{w}{2})^2 + \frac{r}{4}(i^2-1))}{w(w-i)^r}$$

The leaf port efficiency is affected only slightly by the radius, but it's affected greatly by switch width, as shown in figure 3.4(a). Switch width values of sixteen or larger result in near-optimal port efficiency. The bandwidth efficiency is affected by switch width and the radius of the network. Figure 3.4(b) shows that switch width values less than sixteen and radius values less than six result in networks with acceptable bandwidth efficiency.

Tables 3.1 and 3.2 show the topological properties of two CS networks with inside width (w/2)-1 and varying radius values. The leaf port efficiency remains constant at approximately 0.5 and the bisection width is large enough to maintain an acceptable bandwidth efficiency for up to 10,000 processors.

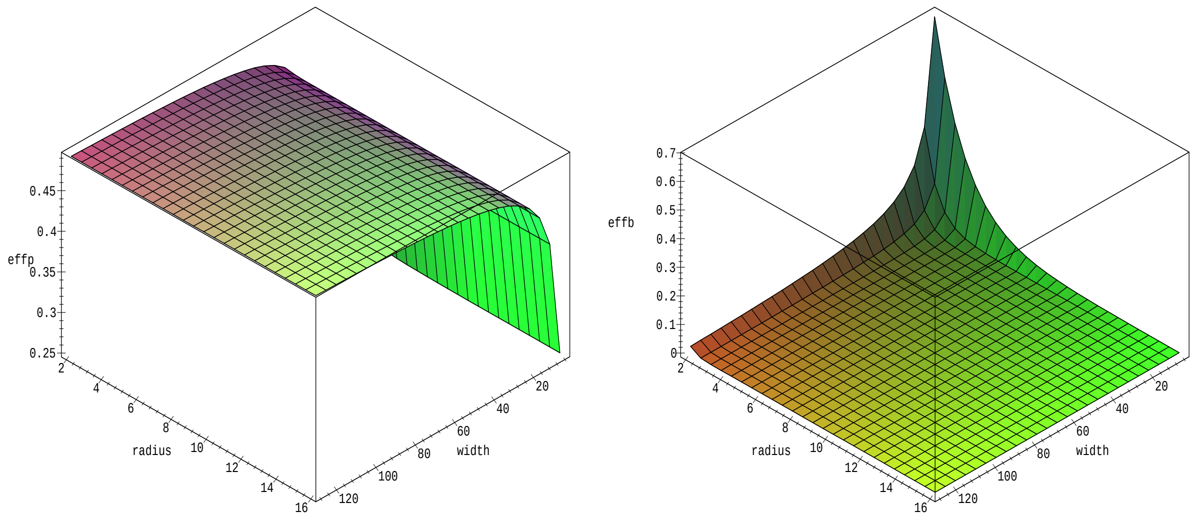


Figure 3.4: Leaf Ports Efficiency (a) and Bandwidth Efficiency (b)

Table 3.1: Topological Properties for CS(8,r,3)

Radius	Diameter	Bisection Width	Switches	Leaf Ports	Effp	Effb
1	1	18	9	40	0.555556	0.9
2	3	20	49	200	0.510204	0.2
3	5	22	249	1000	0.502008	0.044
4	7	24	1249	5000	0.5004	0.0096
5	9	26	6249	25000	0.50008	0.00208
6	11	28	31249	125000	0.500016	0.000448

3.2 Routing

In order to perform the routing in ATM networks, the source processor must specify the path through all the switches to the destination. Since ATM Networks are connection-oriented, the path through intermediate switches must be established before data can be sent [23, 24]. Signaling will be much easier if an appropriate addressing schema is adopted.

Table 3.2: Topological Properties for CS(16,r,7)

Radius	Diameter	Bisection Width	Switches	Leaf Ports	Effp	Effb
1	1	76	17	144	0.529412	1.055556
2	3	88	161	1296	0.503106	0.135802
3	5	100	1457	11664	0.500343	0.017147
4	7	112	13121	104976	0.500038	0.002134

3.2.1 Addressing

The identical address connection method (IACM) has been developed to simplify connection setup in the network. Using this schema, connections between two switches utilize identical port numbers. *Port 5* on the first switch will be connected only to *port 5* on the second switch. This addressing method enables the address of an entity to specify its exact position in the network.

For instance, in figure 3.5(a), port number 0 in the atomic center has a spoke connection to port 0 on the perimeter switch with address 0. Similarly, the additional perimeter switches are given an address that corresponds to the port on the center switch to which they are connected. In CS(8,1,1) network, the 8 perimeter switches are labeled from 0 to 7.

In addition to the spoke connections, the IACM will also be used for the chordal links. One way of making these connections is to choose the first pair of ports that have the same address between neighboring perimeter switches. The first port following the switch address that is shared between switches 3 and 4 in figure 3.5(a) is 5, so port 5 is used as the chordal connection between these switches in figure 3.5(b). Figure 3.5 shows the CS$(8, 1, i)$ networks with varying inside widths.

Using the addressing schema described above, the three leaf ports at the top of figure 3.5(c) are $6 \cdot 2, 0 \cdot 2$, and $1 \cdot 2$. The three leaf ports at the bottom of figure 3.5(c) are $2 \cdot 6, 4 \cdot 6$, and $5 \cdot 6$.

Addressing for multilevel networks begins with the single-level $\mathrm{CS}(w, 1, \frac{w}{2})$. We can think of it as a *high level center* that has $\mathrm{nlp}(w, 1, \frac{w}{2})$ ports. Each port is connected to a perimeter switch. The only difference between an *atomic center* and a *high-level center* is that a high-level center has redundant port addresses ($\mathrm{nlp}(w, 1, \frac{w}{2}) > w$) and is made up of multiple switches. Figure 3.6 gives an example of perimeter switch addressing in a multilevel network.

A leaf port p in $\mathrm{CS}(w, r, i)$ has address $\{a_0, a_1, ..., a_r\}$ where $a_i \epsilon \{0, 1, ..., w-1\}$ and i is from 0 to r.

For example, in figure 3.7, leaf ports A, B, and C have the following addresses:

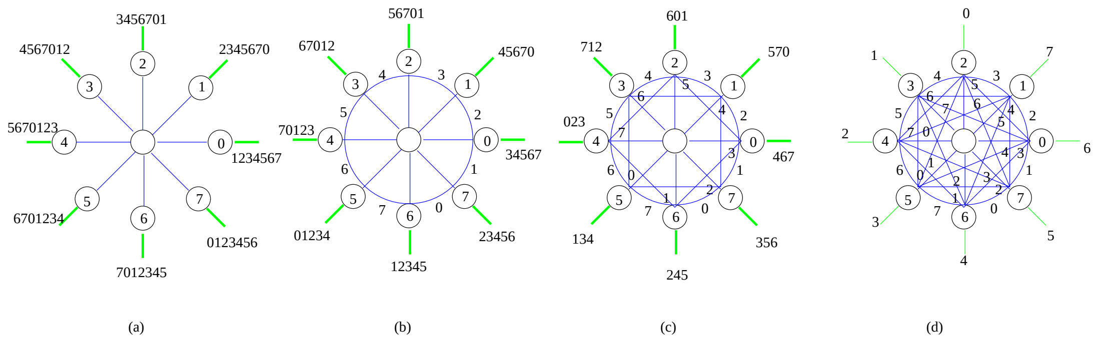


Figure 3.5: An example of the $\mathrm{CS}(8, 1, i)$ networks with (a) $\mathrm{CS}(8, 1, 1)$, (b) $\mathrm{CS}(8, 1, 3)$, (c) $\mathrm{CS}(8, 1, 5)$, (d) $\mathrm{CS}(8, 1, 7)$.

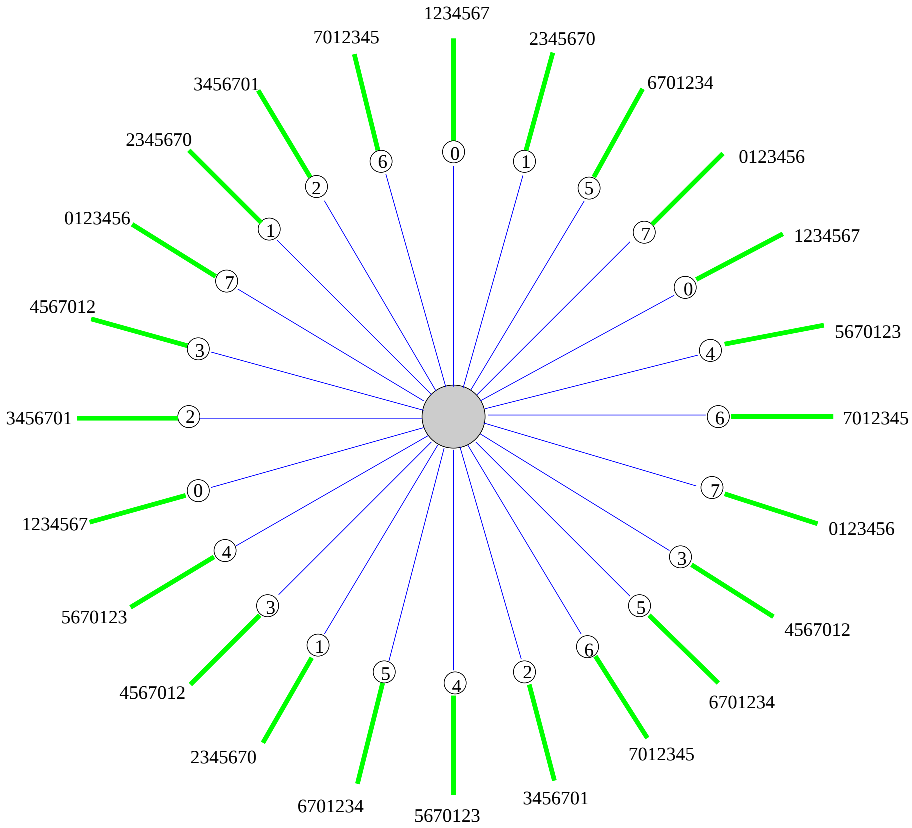


Figure 3.6: Addressing schema for perimeter switches in CS$(8, 2, 1)$

$$A : 1 \cdot 4 \cdot 0$$

$$B : 1 \cdot 6 \cdot 0$$

$$C : 1 \cdot 2 \cdot 6$$

3.2.2 Path Finding

Two techniques have been developed to determine the path between the source and destination node. These methods are as follows:

1. **Spanning Tree Path** The path from source leaf node to destination leaf node uses the natural spanning tree. This path is unique.

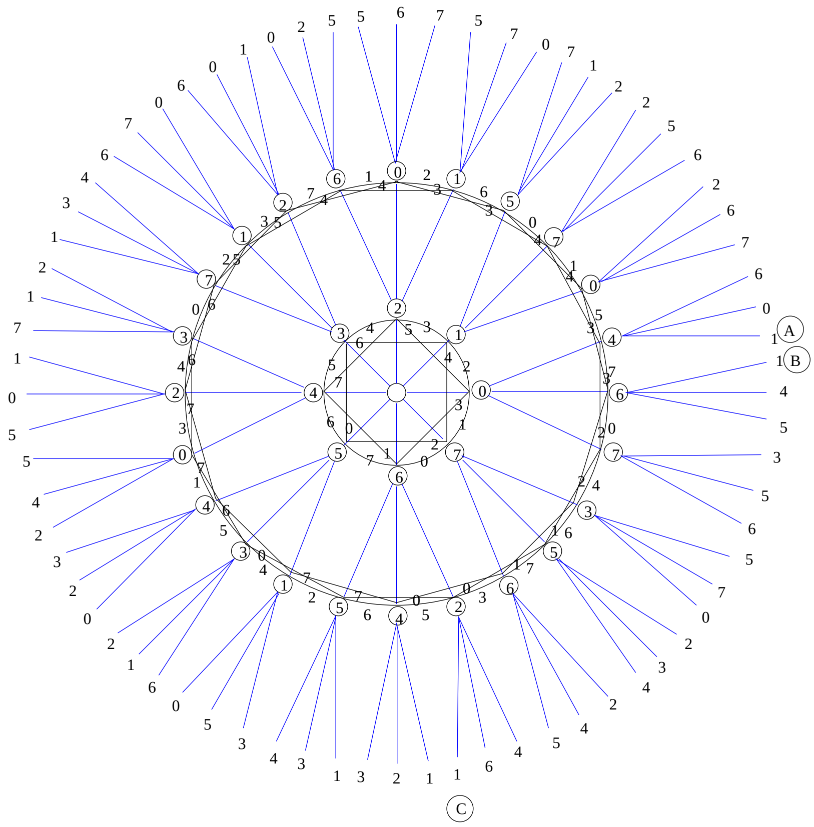


Figure 3.7: Addressing schema for leaf ports in CS(8, 2, 5)

Suppose there exist source leaf node s and destination leaf node d in CS(w, r, i) with the following addresses:

$s : \{s_0, s_1, ..., s_{r-1}, s_r\}$

$d : \{d_0, d_1, ..., d_{r-1}, d_r\}$

Then the spanning tree path is:

$\{s_0, s_1, ..., s_{k-1}, s_k, d_k, d_{k-1}, ..., d_1, d_0\}$ with $s_k \neq d_k$ and $k \leq r$

For example, in figure 3.7, the path from leaf node A to leaf node C is $\{1, 4, 0, 6, 2, 1\}$. The spanning tree path from leaf node A to leaf node B is

$\{1, 4, 6, 1\}$. The spanning tree path is shortened since $s_2 = d_2$.

2. Shortcut

Chordal links offer potentially shorter paths and can decrease the congestion on spanning tree connections. Instinctively, we should choose chordal paths only if the length is less than the spanning tree path. For this reason these paths will be called *shortcuts*. A shortcut will exist between two nodes only if at some level L the number of positions in the ring between s_L and d_L is less than $\frac{i-1}{2}$.

A *position table* is used to determine the number of ring positions between switches in each level. The following algorithm is used to find the shortcut path.

Shortcut Algorithm:

Input: Spanning tree path: $\{s_0, s_1, ..., s_{k-1}, s_k, d_k, d_{k-1}, ..., d_1, d_0\}$
and position table

Output: Shortcut path

For $j = 1$ to k {

If s_j and d_j are neighbors within $\frac{i-1}{2}$ then {

- Read chordal link number c from the position table.
- Output shortcut:

$\{s_0, s_1, ..., s_{j-1}, c, d_{j-1}, ..., d_1, d_0\}$ where $j \leq k$

- Terminate the algorithm.

}

}

If not terminated, output = Input.

□

If we use figure 3.7 for a communication between leaf node A and leaf node B, the spanning tree path is $\{1, 4, 6, 1\}$. Since switch $4 \cdot 0$ and $6 \cdot 0$ differ by one position in the ring, a direct chordal link exists. The corresponding shortcut is $\{1, 7, 1\}$.

3.3 Comparison

A major difference between the CS networks and traditional interconnection networks is that the network is constructed from ATM switches. The processing nodes are connected to leaf ports of the CS network. Processors are involved in communications only if they are the source or destination of the communications. These properties reduce communication overhead and allow the network to be constructed with lower-cost commodity ATM switches.

Table 3.3 compares metrics of interest for Hypercubes, Generalized Fat Trees (GFT), Extended Generalized Fat Trees (XGFT), and Chordal Spoke (CS) networks. These topologies were chosen for the following reasons: The Hypercube has been used extensively in previous research and many algorithms can be mapped

Table 3.3: $\mathrm{CS}(w, r-1, \frac{w}{2})$ compared with other topologies

Comparison of topological properties				
Network	Leaf Ports	Diameter	Degree	Bisection Width
Hypercube Q(2r)	4^r	$2r-1$	$2r$	2^{2r-1}
Fat Trees GFT($\frac{w}{2}, r, p$)	$(\frac{w}{2})^r$	$2r-1$	p	$\frac{w \cdot p^r}{4}$
XGFT(r;$w_1, ..., w_r; p_1, ..., p_r$)	$\prod_{i=1}^{r} w_i$	$2r-1$	p_1	$\frac{w_r \cdot \prod_{i=1}^{r} p_i}{2}$
chordal spoke $\mathrm{CS}(w, r-1, \frac{w}{2})$	$2(\frac{w}{2})^r$	$2r-1$	$\frac{w}{2}$	$(\frac{w}{2})^2 + \frac{r-1}{4}(\frac{w^2}{4} - 1)$

to this topology. Although generalized fat trees have not yet been implemented, in theory they compare favorably with many popular interconnection networks [17, 19, 25]. The topology used in the Connection Machine CM-5 is a special case of the extended generalized fat tree [17, 19].

Table 3.3 shows that $\mathrm{CS}(w, r, i)$ has nearly double the number of leaf nodes when compared to the GFT(r,w,p) and more than that found in a Hypercube with the same diameter. Both the Hypercube and GFT have greater bisection width than CS networks, but the CS(w,r,i) bisection width scales up with the number of nodes in the network. Table 3.4 gives a specific example of how the metrics compare for the three networks.

3.3.1 Embedding

Embedding can be used to show that a host graph is at least as powerful as a given guest graph. The *dilation* of the embedding is the maximum distance on

Table 3.4: CS(16,3,7) compared with other topologies

Comparison of topological properties				
Network	Leaf Nodes	Diameter	Degree	Bisection Width
Hypercube $Q(2 \cdot 4)$	256	7	8	128
Fat Tree $GFT(9,4,3)$	6561	7	3	324
chordal spoke $CS(16,3,7)$	11664	7	7	100

the host between adjacent nodes in the guest topology. The *load* is the maximum number of guest nodes mapped onto a single host node. The *edge congestion* is the maximum number of edges in the guest that are mapped to a single edge in the host.

Embedding a Square Mesh into the Chordal Spoke Network

Suppose we have a Mesh(k, q) with wrap-around connections between processors in adjacent rows or columns. The parameter k is the number of nodes in one row of Mesh and q is the number of dimensions of Mesh. Each node has degree $2q$ and the number of nodes is k^q (when $q = 1$, the Mesh becomes a Ring).

We use a customized chordal spoke network in order to embed a Mesh. The smaller switches can be used at the perimeter of the network to reduce the number of leaf ports. At the perimeter level the switch width is $2q + 2$ instead of the w ports switch used in the interior of the network (assuming $2q + 2 < w$). $2q$ ports correspond to the node degree on the Mesh, one port connects to the center, and the last port connects to the processing element. The inside width for the perimeter switches is $2q + 1$ and the outside width is 1. The *center* is a switch or a chordal spoke network, depending on the size of k^q. At the center, the number of leaf ports is equal to the number of nodes in Mesh ($k^q = w(\frac{w}{2})^r$). The relation between the radius (r) of the chordal spoke network and the dimension (q) of the Mesh is:

$$r = \frac{q \cdot log(k) - log(w)}{log(\frac{w}{2})}$$

Theorem 3.3.1 *In $CS(w, r, \frac{w}{2})$, a Mesh $M(k, q)$ can be embedded with optimal load 1, edge congestion 1, and dilation* $2^{\frac{q \cdot log(k) - log(w)}{log(\frac{w}{2})}}$.

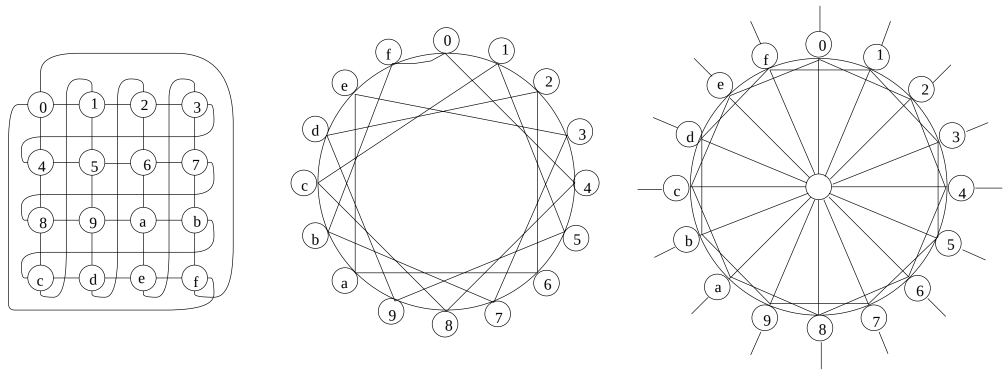
Proof:

Figure 3.8(a) shows M(4,2). In figure 3.8(b) the mesh is redrawn in a chordal ring shape. Figure 3.8(c) shows the embedding into the CS(16, 1, 5) network.

For a given Mesh with k^q nodes, we can always find a chordal spoke network with nlp$(w, r, i) \geq k^q$. Each node in the Mesh is mapped to a leaf port with $load = 1$.

Since there is at most one edge between any two nodes in the Mesh (except between the first node and the last node, which have q edges), the edge congestion is one (for the first and last nodes in Mesh, the edge congestion is q).

Since the center will provide a path when there is no shorter path between the source and the destination nodes, the dilation will be



(a) 4x4 Mesh with wrap-around connections (b) Chordal Ring Shaped Mesh (c) Chordal Spoke Network CS(16, 1, 5)

Figure 3.8: Embedding a Mesh into a chordal spoke network

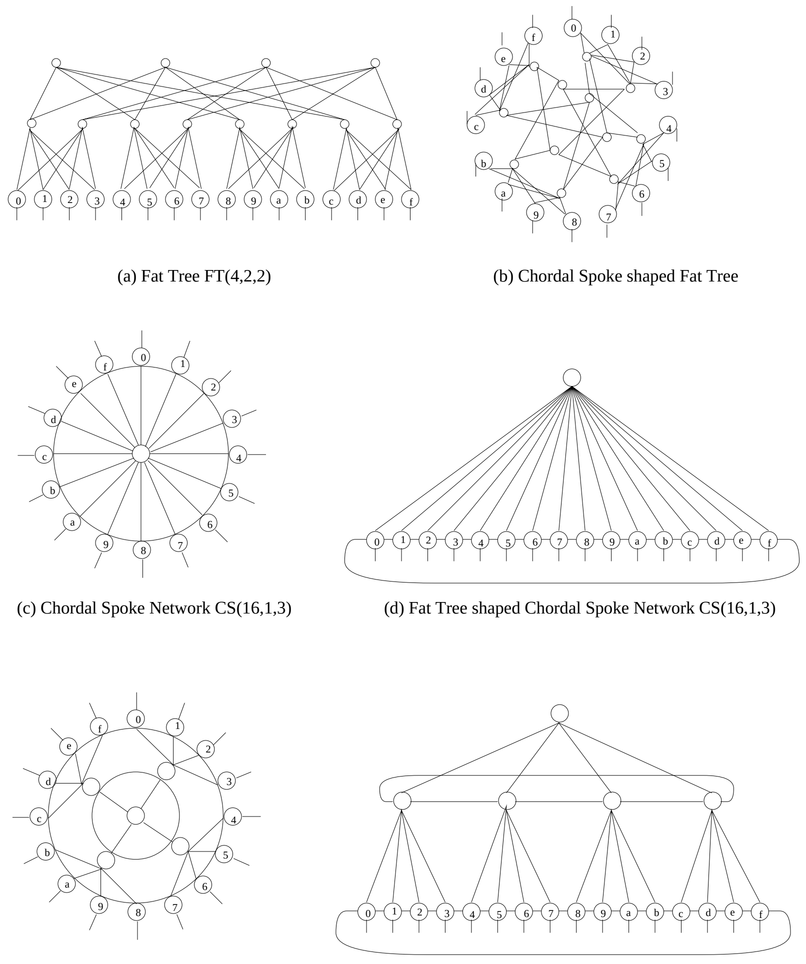
equal to the diameter $2\frac{q log(k) - log(w)}{log(\frac{w}{2})}$.

□

Embedding a fat tree into the chordal spoke network

The example in figure 3.9 is used to illustrate the general procedure. Suppose we have a fat tree FT$(4, 2, 2)$ with four child connections on each non-leaf node, height of two and two connections to parent nodes. Processors are connected to the leaf nodes [25]. There are sixteen leaf nodes and twelve non-leaf nodes [16]. See figure 3.9(a). Figure 3.9(b) shows the fat tree redrawn in a chordal spoke shape.

We can build a customized chordal spoke CS$(16, 1, 3)$ network that has small switches with width four (instead of sixteen) at the perimeter. Figure 3.9(c) illustrates one of these networks and figure 3.9(d) shows the CS$(16, 1, 3)$ redrawn in a fat tree shape.



(a) Fat Tree FT(4,2,2)

(b) Chordal Spoke shaped Fat Tree

(c) Chordal Spoke Network CS(16,1,3)

(d) Fat Tree shaped Chordal Spoke Network CS(16,1,3)

(e) Chordal Spoke Network CS(7,2,3)

(f) Fat Tree shaped Chordal Spoke Network CS(7,2,3)

Figure 3.9: Embedding a fat tree FT$(4, 2, 2)$ into a chordal spoke network

The sixteen leaf nodes in the fat tree are mapped into the sixteen perimeter switches in the chordal spoke network. The twelve non-leaf nodes in the fat tree are mapped into the center of the CS network. Processors are connected with a single edge to the leaf nodes in each topology. The load is twelve and the dilation is one. The edge congestion is two because the two edges in the fat tree that connect to the parents are mapped to a single edge in the CS network.

Another customized CS$(7, 2, 3)$ could be built that uses switches with

Table 3.5: FT(4,2,2) compared with chordal spoke networks

Comparison of topological properties				
Network	Bisection Width	Diameter	Switches	Inside Edges
Fat Tree $FT(4,2,2)$	8	5	28	48
chordal spoke $CS(7,2,3)$	8	5	21	40
chordal spoke $CS(16,1,3)$	66	3	17	32

width four in the atomic center and perimeter. Switches with a width of seven are used for the interior of the network. See figures 3.9 (e) and (f). In this case, the sixteen leaf nodes in the fat tree are mapped into sixteen perimeter switches in the CS network. The eight second-level nodes in the fat tree are mapped into four interior switches in chordal spoke, and the four highest-level nodes in fat tree are mapped into the atomic center of the chordal spoke. The load is four, the dilation is one and edge congestion is two.

Although these embeddings are not perfect, the CS network host graph has several properties that are superior to the guest graphs. Table 3.5 shows that the CS(16,1,3) has a greater bisection width and smaller diameter, and that it uses fewer switches and inside edges than the fat tree with the same number of processors.

3.4 Conclusion

The chordal spoke SASI network has scalable bisection width and acceptable ports and bandwidth efficiency, and it compares well with other interconnection topologies. Because of its regular structure, a routing can be performed in a structured manner.

When the information highway is deployed, parallel processing can be performed using machines that are geographically distant. Chordal spoke networks provide a flexible and economical alternative to existing MPP networks and promise to improve the functionality of clustered computing. They also may present a variable alternative for public network switch interconnection on the information superhighways.

CHAPTER 4

SIMULATION

Simulations of the implementation of two SASI networks – round table and chordal spoke networks – were performed on Ethernet based Hewlett-Packard (HP) workstations. We chose to simulate the networks for two reasons: First, there are not enough ATM switches present in our department to effectively evaluate the topologies. Second, the protocols which were developed for communication in SASI network topologies could not be validated without a simulation. The purpose of simulations is to show a proof of concept for the addressing schema, signaling, and common communication behaviors such as point to point, broadcasting, and reduction.

4.1 Hardware and Software Environment

HP series 700 workstations and Ethernet are used for SASI network behavior simulation. The C language is chosen for simulation programs. communication servers are used to simulate the ATM switches, and communication clients are used to simulate the processors.

The ATM Standard is used throughout the simulation design. The simulation includes addressing, signaling, point to point message sending and receiving, message broadcasting, and message reduction. Message packets are divided into fixed-size ATM cells (53 byte).

The following assumptions are made to simplify the simulation. The diameter of the model network will not exceed 40. Each server is designed to append at most 32 other servers or clients. The number of servers and clients will not exceed 256. There are no restrictions on connections so that any kind of topology can be simulated using the server. Each client is allowed to connect to only one switch.

4.2 Addressing

Addressing is the basic issue for any network topology design. A good addressing schema will make the network easy to understand and easy to design signaling protocols for. The identical address connection method is used as an addressing schema. One problem found in our switch simulation was that it is difficult to specify a complete path in a single ATM cell. A port number for TCP/IP can have four digits. Since an ATM cell has only 53 bytes in it, at most 10 ports can be specified in a cell, this limits the path length to 10. We overcome this problem by setting a base port number. For each switch in a path, a special one-byte-value will be assigned. The real port number will be a switch's special one-byte-value added to the base port number and this real port number is used only when binding the switch to TCP/IP. The path is specified using the switch's one-byte-value.

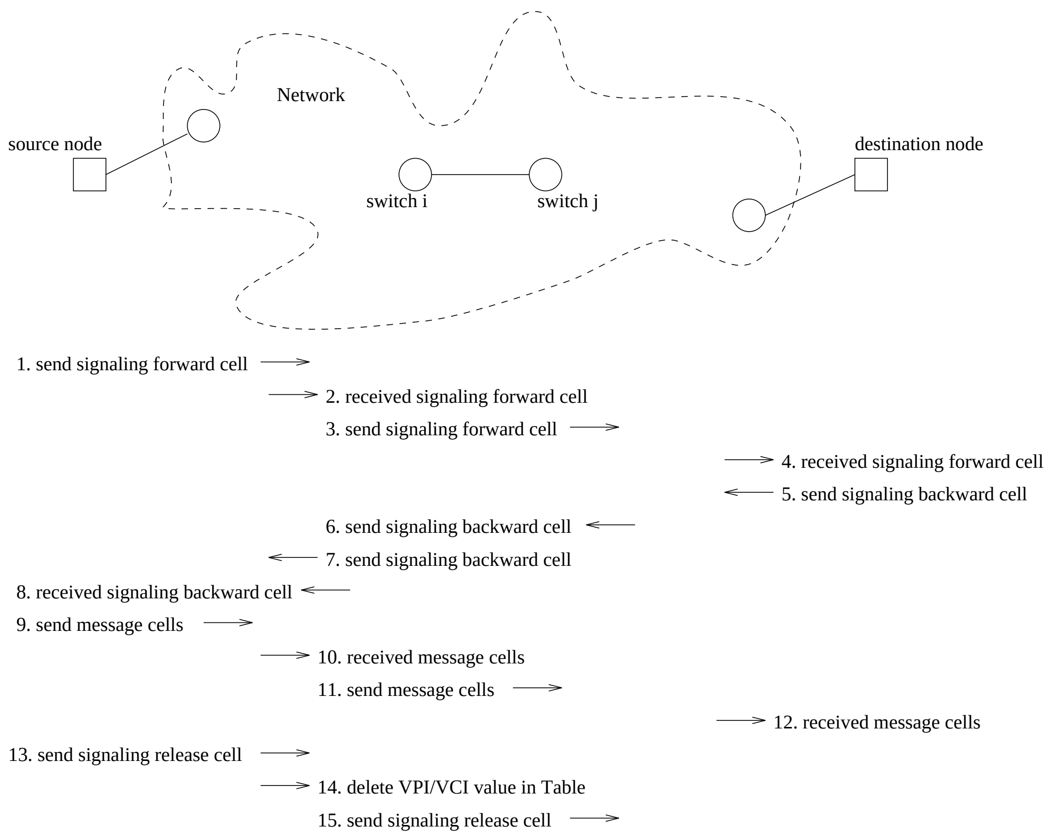


Figure 4.1: Signaling schema for CS Network

The connection between servers uses a host-name and a server TCP/IP port number, so the simulation port number will be omitted for this connection.

4.3 Signaling

When a source node wants to communicate to a destination node, it will find all shortcut and spanning tree path. Once the paths are found, the signaling begins [14]: (See figure 4.1.)

- Source Node

- The source node will send a *signaling forward cell* (request connection) into the network for each path (the path will be included in the cell).
- When the source node receives a *signaling backward cell* (accept connection), it will begin to send the message data which is divided into cells.

- Switches
  - When an ATM switch receives a *signaling forward cell*, it will send the cell into the next switch according to the label on the path;
  - When an ATM switch receives a *signaling backward cell*, it will send the cell into the next switch according to the label on the path and set the Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI) in the ATM path table.
  - When an ATM switch receives a *signaling release cell*, it will delete the corresponding VPI and VCI values on the Path Table.
- Destination Node
  - When the destination node receives a *signaling forward cell* for the first time, it will set the *signaling backward flag* in the cell, and send the cell back.

– When the destination node receives the same *signaling forward cell* for the second time, it will ignore the cell. This prevents loops from occuring.

4.4 Data Transferring

Data packets will be handled by using the ATM standard Segmentation and Reassembly (SAR) schema to divide data into ATM cells, then these cells will be transfered from source client through the ATM network to a destination client. In the simulation, point to point, broadcast, and reduction communications are implemented.

4.4.1 Point to Point Communication

Communication is done in two steps: signaling and data transferring. The ATM standard requires a path from the source client to the destination client to be built before the data can be transfered. In this simulation, we maintain a destination client table called a phoneBook in each client and a path table in each switch. For simplification, once the path is built, no release is needed.

Since the IACM is used, the destination address combined with the source address specifies the path. The whole signaling process is very easy: First the destination address is looked up in the phoneBook. If the destination is found, then the path is already built and the VPI-VCI value can be returned as the token. If the destination client is not in the phoneBook, then a special cell that includes the whole path will be send into the ATM network:

Figure 4.2 shows the information in payload of signaling cell:

L: the length of the path.

C: the current switch pointer.

S: source client's address.

D: destination client's address in reverse order.

The source client's address concatenated with the reverse of the destination client's address specifies the path. When the switch receives the cell, it will create an entry in the path table using information from the cell. The complexity of the switch task is minimized by this method.

Once the path is built, data transferred as a sequence of cells. In the simulation, the size of data packets is chosen as a random value from 1 to 29 cells.

4.4.2 Broadcast Communication

Broadcasting is an important communication. One source client sends a message to all other clients in the network. SASI network topologies have several advantages when performing broadcasts. Since ATM switches have hardware support for multicasting, no signaling is needed and no data must be replicated

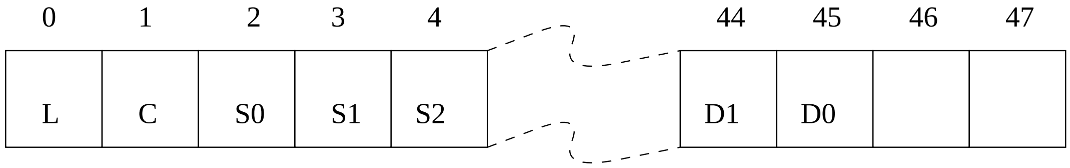


Figure 4.2: Path information in Payload of Signaling cell

for a broadcast. Data cells are transfered through each switch by sending a received data cell from an incoming port to all output (in the chordal spoke network, broadcasting is performed using the natural spanning tree).

Broadcast cells are cells which have the VPI-VCI value set to zero. When a switch receives a data cell with zero VPI-VCI, broadcasting happens.

4.4.3 Reduction Communication

Reduction is a communication by which all clients send a message to one client at the same time. One problem with reduction is that congestion can occur. The simulation performs reductions right after a broadcast. The reduction is treated as a point to point communication.

The difficult part of a reduction implementation is the problem of reassembling the multi-source data cells at the destination. The simulation did this by using the VPI-VCI value to create a mailbox, the reassambly occurs within each mailbox.

4.5 Conclusion

The simulation shows that the round table and chordal spoke network are implementable. The addressing schema is very simple, and the signaling schema is very efficient. There is no need for the switch to have high performance processing capability nor large memory space. Since ATM switches can perform multicast operations, broadcasting is very efficient. Some congestion occured during reduction operations in the round table network. The chordal spoke network performed much better because the chordal connections offer additional paths for the reduction.

CHAPTER 5

CONCLUSION

Parallel processing platforms based on networks of workstations are being utilized extensively to solve challenging computational problems. ATM network technology promises to provide an inexpensive and flexible interconnection media for applications on workstation clusters that require high bandwidth, congestion free communication services.

This thesis introduces a new class of topologies for constructing multicomputers called Small ATM Switch based Interconnection Networks (SASI). They are hierarchical, symmetrical, have low diameters, scalable bisection width, and are easy to implement. This thesis studies and analyzes both port efficiency and bisection width. We found that the round table network achieved maximum port efficiency and the chordal spoke network achieved scalable bisection width. Since the bisection width of the chordal spoke network can be varied, scalable communication abilities can be chosen for different networks. Not only can we use chordal spoke network for parallel topologies, but also general communication networks. When SASI networks are used as general communication networks, it will overcome the gaps between LANs and WANs.

An addressing schema that takes advantage of the hierarchical properties of these networks is presented. This method simplifies the communication primitives and allows low cost switches to be used.

Comparisons and embeddings between the SASIs and other topologies are given. They show that SASIs have equivalent or better properties than traditional MPP models and have a simpler structure and low cost.

A simulation of the implementation of the SASIs is developed over TCP/IP. It shows the SASI networks are implementable. The simulations were based on ATM standards. The results show that these networks have a performance advantage over traditional workstation clusters.

When SASI networks are used as general communication networks, they can take advantage of ATM capabilities and provide for high performance Parallel Virtual Machine (PVM) [3, 6] and multi-media communication tasks. The topologies developed in this thesis are not restricted to ATM switches. They can be applied to other communication switches. When the information highway is deployed, parallel processing can be performed using machines that are geographically distant. SASI networks will provide an important alternative for information highway implementation.

BIBLIOGRAPHY

[1] G. Chen D. Duh and J. Fang, "Algorithms and Properties of a New Two-Level Network with Folded Hypercubes as Basic Modules", *IEEE Transactions on Parallel and Distributed Systems*, **6**(7):714–723, July 1995.

[2] C. P. Kruskal and M. Snir, "A Unified Theory of Interconnection Network Structure", *Theoretical Computer Science*, **48**:75–94, 1986.

[3] K. Leung and M. Hamdi, "Evaluating PVM and Express on Various Network Clusters", In *First International Workshop on High-Speed Network Computing*, pages 57–66. IEEE, 1995.

[4] I. D. Scherson and A. S. Youssef, *"Interconnection Networks for High-Performance Parallel Computers"*, IEEE Computer Society Press, 1994, ch.1, pp. 1-2.

[5] C. L. Seitz, "The Cosmic Cube", *comm. ACM*, **28**(1):22–33, Jan. 1985.

[6] H. Zhou and A. Geist, "Faster Message Passing in PVM", In *First International Workshop on High-Speed Network Computing*, pages 67–73. IEEE, 1995.

[7] P. W. Dowd, S. M. Srinidhi, E. Blade, and R. Claus, "Issues in ATM Support of High Performance Geographically Distributed Computing", In *First International Workshop on High-Speed Network Computing*, pages 19–28. IEEE, 1995.

[8] M. D. Prycker, *"Asynchronous Transfer Mode"*, Ellis Horwood, 1993, ch.3, pp. 103-146.

[9] FORE Systems, Inc, *"ForeRunner ASX-200 ATM Switch User's Manual"*, FORE Systems, Inc, 1994, ch.1 pp. 1-12.

[10] L. Kalampoukas and A. Varma, "Performance of TCP over Multi-Hop ATM Networks: A Comparative Study of ATM-Layer Congestion Control Schemes", In *1995 IEEE International Conference on Communications*, pages 1472–1477. IEEE, 1995.

[11] T. Kwon and C. Jeong, "A Simple, Extendible ATM Switch with Load-balanced Rounding Copy Network", In *1995 IEEE International Conference on Communications*, pages 1122–1126. IEEE, 1995.

[12] K. Murakami and H. S. Kim, "Joint Optimization of Capacity and Flow Assignment for Self-Healing ATM Networks", In *1995 IEEE International Conference on Communications*, pages 216–220. IEEE, 1995.

[13] J. Sole-Pareta, D. Sarkar, J. Liebeherr, and I. F. Akyildiz, "Adaptive Multipath Routing of Connectionless Traffic in an ATM Network", In *1995 IEEE International Conference on Communications*, pages 1626–1630. IEEE, 1995.

[14] C. Partridge, *"Gigabit Networking"*, Addison-Wesley Publishing Company, 1994, ch.1, pp. 1-13, ch.2, pp. 28-31, and ch.3, pp. 51-53.

[15] W. Wei, M. J. Clement, and J. K. Flanagan, "The Round Table ATM Interconnection Network", In *Proceedings of the 1995 International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 445–454. CSREA, 1995.

[16] M. J. Quinn, *"Parallel Computing: Theory and Practice"*, McGraw-Hill, Inc., 1994, ch.3, pp. 52-89.

[17] S. R. Ohring, M. Ibel, S. K. Das, and M. J. Kumar, "On Generalized Fat Trees", In *9th International Parallel Processing Symposium*, pages 37–44. IEEE, 1995.

[18] G. Ramanathan, M. Clement, and P. Crandall, "Hyperweave: A Fault-tolerant Expandable Interconnection Network", In *Proceedings of the Fourth IEEE Symposium on Parallel and Distributed Processing*, pages 479–482. IEEE, 1992.

[19] C. E. Leigserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong, S. Yang, and R. Zak, "The Network Architecture of the Connection Machine CM-5", In *Proc. Fourth Ann. ACM Symp. Parallel Algorithms and Architectures*, pages 272–285. ACM, 1992.

[20] S. B. Akers, D. Harel, and B. Krishnamurthy, "The Star Graph: An Attractive Alternative to the n-Cube", In *Proceedings International Conference on Parallel Processing*, pages 393–400. IEEE, 1987.

[21] R. F. Browne and R. M. Hodgson, "Symmetric degree-four chordal ring networks", *IEE Proceedings*, **137 Pt.E**(4):310–318, July 1990.

[22] S. Miguet and Y. Robert, "Reduction Operations on a Distributed Memory Machine with a Reconfigurable Interconnection Network", *IEEE Transactions on Parallel and Distributed Systems*, **3**(4):500–505, July 1992.

[23] C. Huang, M. Devetsikiotis, I. Lambadaris, and R. Kaye, "Fast Simulation for Self-Similar Traffic in ATM Networks", In *1995 IEEE International Conference on Communications*, pages 438–443. IEEE, 1995.

[24] R. Winkler, "A Routing Framework for Multimedia Multipoint Communications in ATM Networks", In *1995 IEEE International Conference on Communications*, pages 815–820. IEEE, 1995.

[25] C. E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing", *IEEE Trans. Computers*, **C-34**(10):892–901, Oct. 1985.

APPENDIX A

SOURCE CODES OF THREE MAJOR PROGRAMS

This appendix presents the source codes of three programs needed in the simulations.

A.1 Header

```
#define SWITCH_WIDTH 4
#define CELL_SIZE 53
#define PAYLOAD_SIZE 48
#define HEADER_SIZE 5
#define CLIENT_ADDRESS_SIZE 20
#define SWITCH_TABLE_SIZE 30
#define CLIENT_TABLE_SIZE 30
#define PATH_TABLE_SIZE 40
#define BROAD_TABLE_SIZE 20
#define RG 0x04 /* register  */
#define SF 0x05 /* setup forward */
#define SB 0x06 /* setup back */
#define SD 0x01 /* send data */
#define SDL 0x02 /* send last cell of data */

typedef struct {
  char  header[HEADER_SIZE],
        payload[PAYLOAD_SIZE];
} ACell;          /* 53-byte ATM AAL 5 SAR cell format */

typedef struct {
  char  vpi_in,  vci_in,
vpi_out, vci_out;
  int   fd;
} APath;

typedef struct {
  char  sWitch;
  int   fd;
} ASwitch;

typedef struct {
  char  address[CLIENT_ADDRESS_SIZE],
        seq;
} ABroad;

typedef struct {
```

```
  char  vpi, vci,
        num;
} AMailbox;

typedef struct {
  char  address[CLIENT_ADDRESS_SIZE],
 vpi, vci;
} AClient;

setVpiVci(p, c, pt) char p, c; ACell *pt; {
/* with NNI format  */
}

incVpiVci(pt) ACell *pt; {
/* with NNI format  */
}

getVpi(pt) ACell *pt; { }
getVci(pt) ACell *pt; { }
incPathCur(pt) ACell *pt; { }
decPathCur(pt) ACell *pt; { }
getSwitch(pt) ACell *pt; { }
setPayloadType(seed, pt) char seed; ACell *pt; { }
payloadType(typ, pt) char typ; ACell *pt; { }
displayHeader(pt) ACell *pt;{ }
displayPayload(pt) ACell *pt;{ }
broad(pt) ACell *pt; { }
```

A.2 Switch

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <signal.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <stdlib.h>
#include <netdb.h>
#include <string.h>
#include "cell.h"

struct sockaddr_in sockaddr, caller;
struct sockaddr_in socketname;
struct hostent * remote_host;

char  selfSwitch;
APath   pathTable[PATH_TABLE_SIZE];
int     pathTableCur=0;
ABroad  broadTable[BROAD_TABLE_SIZE];
int     broadTableCur=0;
ASwitch switchTable[SWITCH_TABLE_SIZE];
int     switchTableCur=0;
ACell rec_cell, send_cell;

int flag=0;  /*  flag is for debug tracing!   */

/* arg0 arg1 [arg2    arg3]     ... arg[last]
   prog name;   self;   remoteH  remoteS   ... debug flag(!)
*/

main(argc,argv) int argc; char *argv[];
{
   int sock;
   int tempear;
   int sockaddr_in_length = sizeof(struct sockaddr_in);
   int read_ret;
   int fromlen = sizeof(struct sockaddr_in);
   int rsel = 0, maxfds = 32;
   extern int errno;

   system("clear");
   selfSwitch= argv[1][0];
   if( argv[argc-1][0] == '!' ) flag= 1;

/* open a net socket, using stream (file-style i/o) mode,
   with protocol irrelevant (==0)
*/
   sock = socket( AF_INET, SOCK_STREAM, 0 );
```

```
   rsel |= 1 << sock;

/* register the socket
*/
   sockaddr.sin_family = AF_INET;
   sockaddr.sin_addr.s_addr = INADDR_ANY;
   sockaddr.sin_port = htons(selfSwitch + 2000);

   bind( sock, (struct sockaddr *) &sockaddr, sizeof sockaddr );

   getsockname( sock, (struct sockaddr *) &sockaddr,
(int *)&sockaddr_in_length );
   printf("\nSWITCH: %c\n" , selfSwitch);

/* waiting for plug...
*/
   if( argc > 3 ) {   /* plug into other remote switch(es)   */
      int i;
      for( i= 2; i< argc - 1; i += 2 ) {
         tempear = socket(AF_INET, SOCK_STREAM,0);
         rsel |= 1 << tempear;
         socketname.sin_family = AF_INET;
         if ((remote_host=gethostbyname(argv[i]))==
      (struct hostent *)NULL){
           fprintf(stderr, "unknown host: %s\n", argv[i]);
           exit(__LINE__);
         }
         (void) bcopy((char *)remote_host->h_addr,
      (char *) &socketname.sin_addr,
      remote_host->h_length);
         socketname.sin_port = htons(argv[i+1][0] + 2000);
         if (connect(tempear, (struct sockaddr *) &socketname,
     sizeof socketname) < 0){
           printf("..connect failed....\n");
           exit(__LINE__);
         }
         switchTable[switchTableCur].sWitch= argv[i+1][0];
         switchTable[switchTableCur].fd=tempear;
         switchTableCur++;
         setPayloadType(RG, send_cell);
         send_cell.payload[0]= selfSwitch;
         write(tempear, send_cell, CELL_SIZE);
      }
   }

/* put an ear to the socket. up to 20 clients
*/
  listen( sock, 20 );

  while(1) {
    char fd;
```

```
    int rval, trsel;
    trsel = rsel;
    rval = select(maxfds,&trsel,0,0,0);
    for(fd = 0; fd < 32; fd++) {
      if((trsel & (1 << fd)) == 0) continue;
      if(fd == sock) {
/* remote register coming ...
*/
        if ((tempear = accept(sock,(struct sockaddr *)&caller,&fromlen ))
     >= 0 ) {
          rsel |= 1 << tempear;
  bzero(rec_cell, CELL_SIZE);
  read(tempear, rec_cell, CELL_SIZE);
  if( payloadType(RG, rec_cell)) { /*remote register in */
    switchTable[switchTableCur].sWitch= rec_cell.payload[0];
            switchTable[switchTableCur++].fd=tempear;
    printf("Node < %c > registered!\n", rec_cell.payload[0]);
  }
        }
      }
      else {
        if(flag) printf("----------remote message coming.-----------\n");
bzero(rec_cell, CELL_SIZE);
        read_ret = read( fd, rec_cell, CELL_SIZE );

/* There are three kinds of communications:
1. Point to Point: caused by one client or by one client
    answering broadcast.
2. broadcast     : caused by one client.
3. reduction     : caused by answering broadcast.

*/
if( payloadType(SF, rec_cell) ) sf(fd);
if( payloadType(SB, rec_cell) ) sb();
if( payloadType(SD, rec_cell) ||
    payloadType(SDL, rec_cell) ) sd(fd);

      } /* else */
    } /* end of for fd loop */
  } /* end of while loop */
  exit(0);
}

sf(fd_in) char fd_in; { }
/* --------- function sf: remote send setup forward request! ------
  used by first time point to point communication with sb to
  build a new path.
input: rec_cell, output: send_cell.
1. move forward path position in cell.
2. build new path into path table.
3. increase vpi vci values in cell.
```

```
4. find out next switch and send cell.
*/

sb() {}
/* ----- function sb: remote send setup backward request! ----------
input: rec_cell, output: send_cell.
1. move backward path position in cell.
2. find out next switch and send cell.
*/

sd(fd) int fd; {}
/* ----- function sd: receive send data !  -----------------------
  path table has already built.
  data may be all kinds of:
     i.  point to point:
     ii. broadcast.
     iii. reduction( included in point to point).

  data may be the first cell, middle cell or last cell. the switch
  does not know and does not need to know.

  input: rec_cell, output: send_cell.

  case broadcast:
1. make sure the input cell is not the repeat one.
2. send the cell to all the switches but comming one.

  case point to point:
1. from given cell's vpi_vci, find out the outgoing fd.
2. from given cell's vpi_vci, find out the outgoing vpi_vci and
   set it as cell's new vpi_vci.
3. send cell.
*/

setPathTable(pt) ACell *pt; {}
/*
  1. create a new tuple in path table.
  2. put cell's vpi, vci into tuple's in and put vpi, vci +1 into
     tuple's out.
  3. from cell's switch got next fd number and put it into tuple.
*/

getPathTableIndex(vpi, vci) char vpi, vci; {}
/*
  for given vpi, vci, find the position in path table.
*/

getFd(swch) char swch; {}
/*
  for given switch, find the position in switch table. return
  corresponding fd.
```

```
*/

fitBroadTable() {}
/* seek through broadTable:
if there is no source client exist, then create it;
else if the current number is greater than the table's
        then return 0
             else return -1
*/

displayPathTable(){ }
displaySwitchTable(){ }
```

A.3 Client

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <signal.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <stdlib.h>
#include <netdb.h>
#include <time.h>
#include "cell.h"

void handler();
char  selfAddress[CLIENT_ADDRESS_SIZE];
AClient phoneBook[CLIENT_TABLE_SIZE];
int phoneBookCur= 0;
ACell  rec_cell,send_cell;
int  plug, address_size_cur, flag=0;
AMailbox mailboxTable[CLIENT_TABLE_SIZE];
int mailboxTableCur= 0;
unsigned char broadSeed=0, vpi, vci= 0;
unsigned long broad_start;
int     broad_flag= 0;

/* arg0 arg1 arg2 arg3 arg4
   prog name; remote host;  remote switch;  client address  flag
flag=1 is for showing debug info.
*/
/* ------- main function ----------------------------*/
main(argc,argv) int argc; char *argv[]; {
   struct sockaddr_in socketname;
   struct hostent *remote_host;
   char seed;

   broad_start= clock();
   system("clear");
   address_size_cur= strlen(argv[3]);
   memcpy(selfAddress, argv[3], address_size_cur);
   seed = selfAddress[0];
   vpi= selfAddress[0] - '0';
   srand(seed);
   if( argv[4][0] == '!' ) flag= 1;

   /* plug to switch  */
   plug = socket( AF_INET, SOCK_STREAM, 0 );
   socketname.sin_family = AF_INET;
   if ((remote_host = gethostbyname( argv[1] ))==(struct hostnet *)NULL )
   {
      fprintf( stderr, "unknown host: %s\n", argv[1] );
```

```
      exit(__LINE__);
   }
   (void) bcopy((char *)remote_host->h_addr,(char *) &socketname.sin_addr,
remote_host->h_length);
   socketname.sin_port = htons(argv[2][0] + 2000);
   if( connect( plug, (struct sockaddr *) &socketname,sizeof socketname )
< 0 ) {
      printf("\nunknown switch!\n");
      exit(__LINE__);
   }
   /* send first byte of address: local address
   */
   send_cell.payload[0]= selfAddress[0];
   setPayloadType(RG, send_cell);
   write(plug, send_cell, CELL_SIZE);

   while (1) {
      waiting();
   }
   close(plug);
   exit(0); /* all the socket connections closed automatically */
}

waiting(){ }
/* ----- waiting for message ----------------------------
  the function will receive the cells unless ctl-c happens which will
  cause the client send the data to others (point to point or broadcast).
  receive:
  input: rec_cell, output: send_cell.
     case SF:
1. move backward path position in cell.
2. set signal back for cell.
3. send cell.

     case SB:
1. set client table.
2. call point to point sending: send PtoP().

     case SD:
1. find out the mail bucket. or build the mail bucket( first cell).
2. increase the totle number by one in the mail bucket.

     case SDL: ( last cell of the one mail bucket ).
1. find out the mail bucket. or build the mail bucket(single cell).
2. display the totle number in the mail bucket.
3. delete this mail bucket.
*/


void handler(sig) int sig;{ }
/* ------ signal handler  ------------------------- */
```

```
sendMessage(){ }
/* ------ send Data function ----------------------
   input: remote client address or "broad".
   output:broadcast or point to point.
*/

sendBroad() { }
/* ------ send broad -----------------------------
   prepare: send_cell with plt(SD), vpi_vci(0)( broadcasting),
self address and self address's length in it.
   send send_cell couple of times then send send_cell with "SDL" in it.
   each time the increased sequential number will put into header.
*/

sendPtoP(rap) char *rap; { }
/*  send point to point -------------------------
    input : remote client address (phone #).
    output: if the client exist, send the data
    else send signal.
*/

PtoPData(id) char id; { }
/* -------- send point to point -------------
   1. find out if the remote address is in client table.
      case (new remote client): send a signal cell: SF, vpi_vci, path.
      case (old remote client):
*/

incMail(pt) ACell *pt; {
}

finishMail(pt) ACell *pt; { }
/*  finishing mail -------------------------
   input : cell point.
   output: 1. display the number of mails in mail bucket.
   2. delete this mail bucket.
   3. if this mail is broadcast, then send reduction ( point to
      point ).
*/

setPath(selfA, remoteA, len, pt) char *selfA, *remoteA; int len;
    ACell *pt; { }
getPhoneBookId(address) char *address;{ }
noDupPhone(vp, vc) char vp, vc;{ }
setPhoneBook(addp, vp, vc) char *addp, vp, vc;{ }
displayPhoneBook(){ }
setBroad(seed, pt) char seed; ACell *pt; { }
incBroadSeed(pt)  ACell *pt;{ }
displayMail() { }
```