Open access • Journal Article • DOI:10.1109/TC.1980.1675586

Processor Interconnection Strategies — Source link

Finkel, Solomon

Institutions: University of Wisconsin-Madison

Published on: 01 May 1980 - IEEE Transactions on Computers (IEEE)

Topics: Distributed memory, Network topology, Routing protocol, Routing table and Static routing

PROCESSOR INTERCONNECTION STRATEGIES

by

Raphael A. Finkel

Marvin H. Solomon

Computer Sciences Technical Report #301

July 1977

PROCESSOR INTERCONNECTION STRATEGIES

by

Raphael A. Finkel

Marvin H. Solomon

Technical Report 301

ABSTRACT

In this paper we describe four topologies for interconnecting many identical processors into a computer network. Each topology is investigated with respect to average interprocessor distance, bus load, and routing algorithms. These topologies share the property that each processor can communicate directly with at most a small number of other processors.

Processor Interconnection Strategies

INTRODUCTION

During the past few years, the prospect of connecting together many small computers has become more attractive. Small machines are becoming less expensive and will very likely continue to do so for some time to come. On the other hand, very large machines remain costly. In order to make a fast machine faster, it may be cost-effective to interconnect many small computers, perhaps hundreds or thousands. Such a combination has been called a mega-micro computer [Wittie 76].

In this paper, we address ourselves to one aspect of the design of such mega-mico computers: the combinatoric properties of several interconnection topologies for very large ensembles of identical processors. The topologies we consider have several factors in common. They are uniform both in the kinds of processors and links involved and the patterns of interconnection, they are all extendable to arbitrarily large ensembles, and they all have a number of connections that increases linearly with the number of processors. We investigate each topology with respect to distances between processors (the number of intermediate processors required to relay a message), possible traffic bottlenecks, complexity of routing algorithms, and geometric aspects relevant to physical placement of components.

From many points of view, the ideal way to connect N processors is one in which each processor can communicate directly with

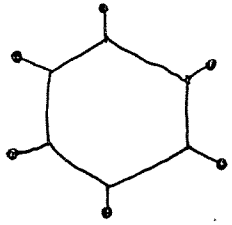
any other. However, this connection strategy requires a number of connections that grows quadratically with the number of processors. As N grows large, the complexity of interconnection may become unacceptable. Let us assume, therefore, that up to p processors can be connected together in a clique: a network in which each pair of processors is connected directly. We will also consider two processors to be directly connected if the complexity of sending a message from one to the other is so small that the transmission can be considered to be an atomic action.

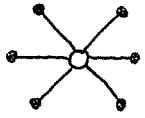
One way to provide complete connection is to dedicate a two-way transmission line between each pair (Figure 1a). Another is to place the p processors on a ring (Figure 1b). We shall not be concerned with the mechanisms for sending, relaying, and receiving messages on a ring; rather, we will consider message transmission to be a single act. Yet another scheme is to have the processors communicate via a common bus or shared memory (Figure 1c). Again, we will assume that problems of contention are not relevant at our level of detail. (However, we will derive some facts about bus traffic that can be used to estimate the amount of contention expected.) Throughout the remainder of this paper, we will use the term bus and illustrations similar to 1c to denote a clique of p processors, but all results apply equally well to other local connection schemes. We will often refer to processors as nodes in the network.

Processors on more than one bus must be able to transfer messages from one bus to the other, but the destination node may be unable to accept the message. In that case, the transfer node

Figure 1

a)  Fully Connected

b)  Ring

c)  Bus

Ways to Connect
p processors

must hold on to the message until it can send it. Problems occur if each of two adjacent nodes is holding as much as it can in its local buffers and must send a message to the other node. There is no way that the message can be sent or the buffers can be emptied. Such a situation is known as buffer deadlock.

This problem can often be overcome in networks arranged so that each processor lies on at most two busses. Messages pass in two directions across the processor. One message buffer for each direction will often be enough at each processor. The buffers need to be long enough only for one message. This solution will work if the network is acyclic. The only way a deadlock could occur would be for a node to have a message that could not be delivered. This situation can only arise if the next node in the direction of the message has a full buffer, none of whose messages could be delivered. But all of those messages must be destined for processors further down the path; since there is no circularity, there is no danger that any of those messages need to pass through the first node. Therefore, the cause of the obstruction must be in the network minus the first processor. A similar argument can show that the obstruction must not include the second processor, and in fact there is no possiblilty for deadlock at all.

Even if the network contains cycles, the two-buffer technique can serve to lessen the severity of buffer deadlock. The only deadlock situation is then a cycle of processors each of which has a message for the next processor in the cycle. If the buffers are made large enough to hold several messages, it may be

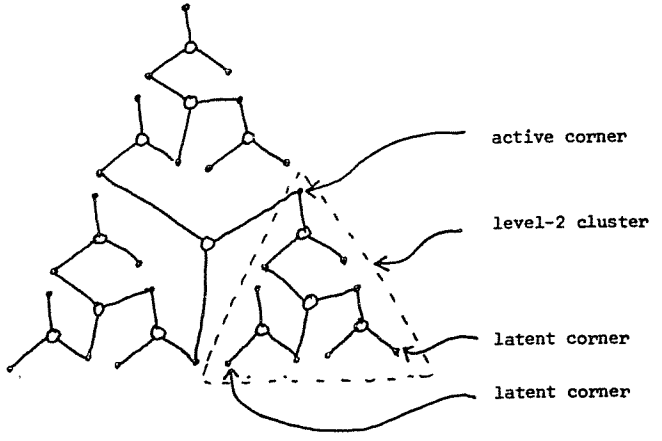
possible to make the probability of deadlock very small.

The sections that follow treat four topologies of interconnection. The most interesting are the ones we call the snowflake and the dense snowflake. Others that exhibit extreme properties are the star and the p-cube.

THE SNOWFLAKE

This structure is based on the hypercube of D. Wittie [Wittie 76]. Supposing that N, the total number of processors, is much larger than p, the number of processors on a bus, how can a number of p-processor clusters be connected together to form a network? One way to build a network with p^2 processors is to construct p p-processor clusters and link them together by connecting one processor from each cluster to a new bus. This process can be iterated to give the following recursive construction. A level-one cluster is composed of p processors connected by one bus. Each of the p processors is at a corner of the network. To make a level-two cluster, introduce a new bus that connects together p level-one clusters by linking to a corner of each one. P of the remaining processors, one from each level-one cluster, are designated as corners of the level-two cluster. In general, to form a level-n cluster, take p level-(n-1) clusters and introduce a new bus that connects together a corner of each. These corners are called the active corners of the level-(n-1) clusters. Choose a different corner from each of those p clusters to be a corner of the level-n cluster. These corners are called latent corners of the level-(n-1) clusters. Each cluster

Figure 2



Snowflake: n = 3, p = 3

thus has one active corner, one latent corner, and p-2 other corners. A single processor forms a level-0 cluster, whose p corners are all that same processor. The construction for n = 3, p = 3 is pictured in Figure 2.

Internode distance

The maximum distance between any two nodes in an n-cluster is the distance between two corners: $2^n - 1$. This formula solves the recurrence

$$\text{MaxDist}(n) = \begin{cases} 1 + 2\text{MaxDist}(n-1) & \text{if } n > 0 \\ 0 & \text{if } n = 0. \end{cases}$$

The average internode distance can be calculated from AvgCornDist(n), the average distance from a corner of a level-n cluster to any node in that cluster:

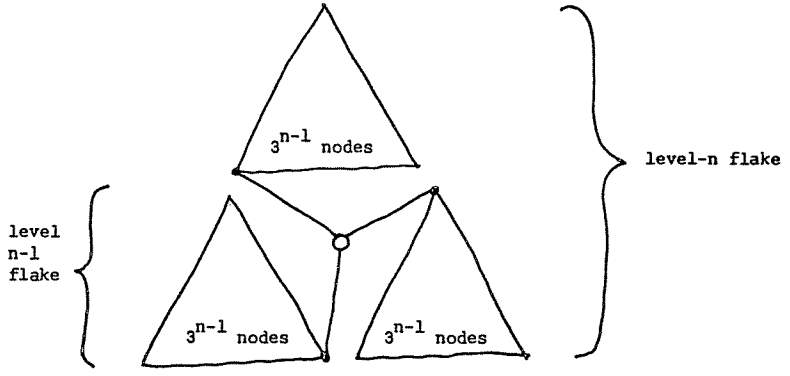
$$\text{AvgDist}(n) = \frac{1}{p}\,\text{AvgDist}(n-1) + \frac{p-1}{p}(1 + 2\,\text{AvgCornDist}(n-1)),$$

$$\begin{aligned}\text{AvgCornDist}(n) &= \frac{1}{p}\,\text{AvgCornDist}(n-1) \\ &\quad + \frac{p-1}{p}(2^{n-1} + \text{AvgCornDist}(n-1)) \\ &= \text{AvgCornDist}(n-1) + \frac{(p-1)\,2^{n-1}}{p},\end{aligned}$$

$$\text{AvgDist}(0) = \text{AvgCornDist}(0) = 0.$$

These relations come from the recursive nature of the cluster. If two nodes are chosen randomly in a level-n cluster, there is a chance of 1/p that they fall in the same level-(n-1) cluster, and a chance of (p-1)/p that they do not. In the latter

Figure 3



Use of the Central Bus

case, a connection between them will require a cross-cluster bus. Figure 3 shows this argument pictorially for p = 3.

The solutions to these recurrences are as follows:

$$\text{AvgDist}(n) = \frac{2^n (2p^2 - 4p + 2)}{2p^2 - p} - \frac{1}{p^n (2p - 1)} - \frac{p - 2}{p}$$

$$= \frac{2p^2 - 4p + 2}{2p^2 - p} 2^n + O(1),$$

$$\text{AvgCornDist}(n) = \frac{2^n (p - 1)}{p} - \frac{p - 1}{p} .$$

For p=3, AvgDist(n) = (8/15) 2^n + O(1); for large p and n, AvgDist(n) $\cong$ 2^n. Since N, the total number of nodes, is p^n, we can express the highest term of AvgDist(n):

$$\text{AvgDist}(n) \cong \frac{2p^2 - 4p + 2}{2p^2 - p} N^{1/\log p}$$

Thus, as the number of processors N rises, the average distance between them rises as the log p root of N. For p = 4, the the distance rises as the square root of N. For p = 16, the distance rises as the fourth root.

Number of busses

A level-n cluster has p^n processors. The number of busses in a level-n cluster is

$$\text{NumBus}(n) = \begin{cases} 1 + p\ \text{NumBus}(n-1) & \text{if } n > 0 \\ 0 & \text{if } n = 0. \end{cases}$$

The solution to this recurrence is

$$\text{NumBus}(n) = \frac{p^n - 1}{p - 1} = \frac{N - 1}{p - 1},$$

so the number of busses grows linearly with the number of processors.

Most processors lie on only one bus and take no part in relaying messages. The number of such processors is

$$\text{OneBusProcessors}(n) = \begin{cases} p\ \text{OneBusProcessors}(n-1) - p & \text{if } n > 1, \\ p & \text{if } n = 1. \end{cases}$$

or, in closed form,

$$\text{OneBusProcessors}(n) = p^n - \frac{p^n - p}{p - 1} = N - \frac{N - p}{p - 1} .$$

Thus the proportion of processors on one bus is

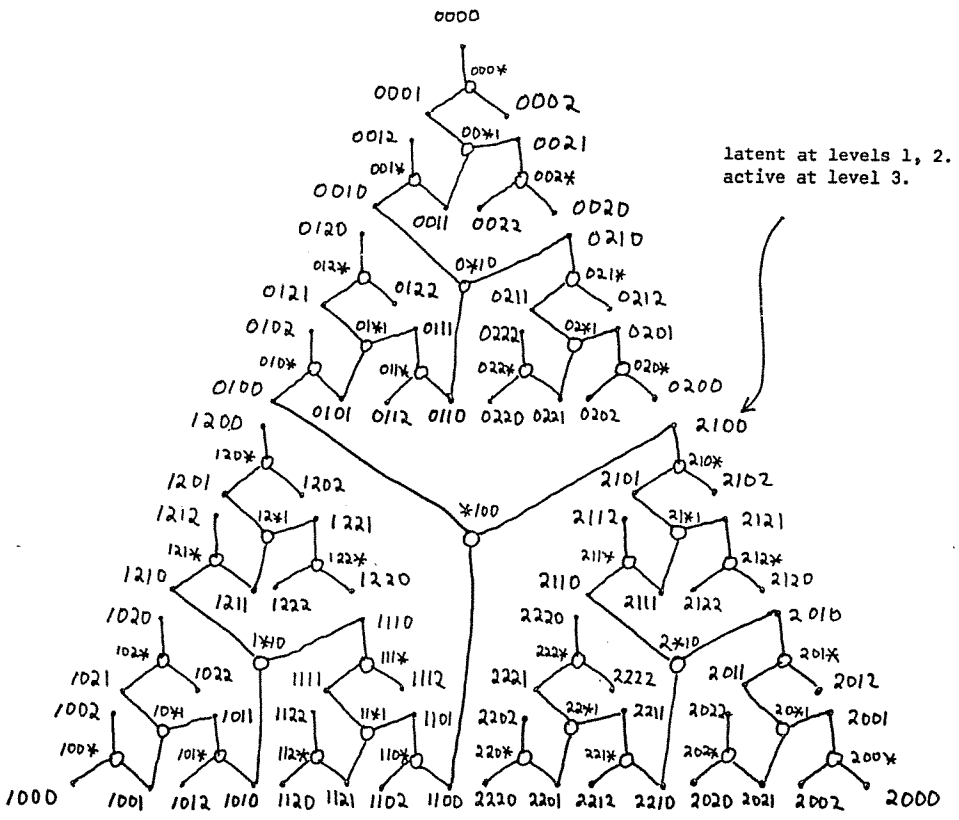
$$\frac{\text{OneBusProcessors}(n)}{\text{NumProc}(n)} = 1 - \frac{1}{p-1} + \frac{p}{p-1}\frac{1}{N} \cong 1 - \frac{1}{p-1} \text{ for large N.}$$

For p = 3, half the processors have no relaying function, and as p approaches infinity, a vanishingly small fraction are used for relaying messages. The sparsity of connection leads one to believe that some advantage might be gained by adding extra busses so that every processor connects two busses. The dense snowflake, to be discussed later, builds on this idea.

Routing and addresses

A unique path leads directly from any node in the snowflake to any other. The correct numbering scheme makes routing easy. Assign a unique n-digit address (base p) to each node in a level-n cluster. When p level-(n-1) subclusters are combined to form a level-n cluster, assign a distinct digit to each subclus-

Figure 4



ter, and concatenate this digit to the beginning of the address of each of its processors. The active and latent corners of the new cluster are the latent corners of the constituent clusters numbered one and zero, respectively. See Figure 4 for the case $n = 4$, $p = 3$.

This tying strategy gives rise to some new terminology: A node with address $d^{n-j}10^{j-1}$ (where d means any digit) is the latent corner of its i-clusters for $1 \leq i \leq j-1$, and it is the active corner of its j-subcluster. It will be the point of connection to the level-(j+1) cluster. This scheme has the property that the processors on any one bus are those whose addresses differ in a given position. For example, the central bus in Figure 4 connects processors 0100, 1100, and 2100. It is natural to denote this bus by *100. Our conventions for choosing active corners insure that each bus has an address of the form $d^{n-1}*$ or $d^{n-j}*10^{j-2}$. In the former case, the bus is introduced to form a level-1 cluster; in the latter case, it is introduced to form a level-j cluster. A level-j bus creates a level-j cluster.

To demonstrate routing, let us direct a message from node 2101 to node 2021. A glance at Figure 4 will show that the unique path is as follows:

```
node    bus
2101
        21*1
2111
        211*
2110
        2*10
2010
        201*
2011
        20*1
2021
```

The algorithm to find the route can be written recursively on the highest level bus that needs to be employed. It works by finding the smallest j such that the source and destination are in the same level-j cluster. The message then goes from the source to the active corner of its level-(j-1) cluster, across the level-j bus to the active corner of the destination cluster, and thence to the destination.

```
type digit = 0 .. p-1 union asterisk;
     address = array[1 .. n] of digit;

procedure route(source, dest: address);
     begin {route}
          route1(source,dest,n)
     end {route}
```



```
procedure route1(source,dest : address; level : integer);
     {source[i] = dest[i] for i > level}
     begin {route1}
     if level = 0 then
          {trivial path} output(source)
     else if source[level] = dest[level] then
          route1(source,dest,level-1)
     else begin
          bus := source[n .. level+1] cat asterisk
               cat 0^(j-1);
          if level > 1 then bus[j-1] := 1;
          dest1 := bus;
          dest1[level] := source[level];
          route1(source,dest1,level-1);
          output(bus);
          source1 := bus;
          source1[level] := dest[level];
          route1(source1,dest,level-1)
          end
end {route1}
```

This algorithm can be modified in a straightforward way so that each node on the path does only as much computation as necessary to transmit the message the next step on its way.

Bus Load

By the load on a bus, we mean the probability that a message between a random pair of processors will use that bus. We will express the load on a bus in terms of loads due to messages restricted to various classes of sources and destinations. In each case, a load of 1 means that a message of the given type certainly goes through the given bus.

Let ActLoad(bus) be the load on a bus due to messages between arbitrary nodes in the cluster and the active corner of that cluster. Similarly, LatLoad(bus) is the load on a bus due to messages between arbitrary nodes and the latent corner. Let AACrossLoad(bus) be the load on a bus in a cluster connected by its active corner to another cluster; the paths that contribute

to this load are those that connect any node in the first cluster to any node in the second. In each of these cases, the size of the cluster concerned is determined by the number of digits in the bus name.

The load on a bus can now be expressed recursively on its address. We will represent addresses by means of the following conventions: w stands for an arbitrary string, d for an arbitrary digit, * for the asterisk that appears in each bus address, $\overline{0}$ and $\overline{1}$ for any digit not 0 or 1, respectively, and h for any digit greater than 1.

The recurrences themselves follow:

$$\text{Load}(*w) = \frac{p-1}{p}$$

$$\text{Load}(dw) = \frac{\text{Load}(w) + 2(p-1)\ \text{AACrossLoad}(w)}{p^2}$$

$$\text{AACrossLoad}(*w) = \frac{p-1}{p}$$

$$\text{AACrossLoad}(\overline{1}w) = \frac{\text{ActLoad}(w)}{p}$$

$$\text{AACrossLoad}(1w) = \frac{p-1}{p}\ \text{ActLatLoad}(w) + \frac{1}{p}\ \text{LatLoad}(w)$$

$$\text{ActLoad}(*w) = \frac{p-1}{p}$$

$$\text{ActLoad}(\overline{1}w) = \frac{\text{ActLoad}(w)}{p}$$

$$\text{ActLoad}(1w) = \frac{p-1}{p}\ \text{ActLatLoad}(w) + \frac{1}{p}\ \text{LatLoad}(w)$$

$$\text{LatLoad}(*w) = \frac{p-1}{p}$$

$$\text{LatLoad}(\overline{0}w) = \frac{\text{ActLoad}(w)}{p}$$

$$\text{LatLoad}(0w) = \frac{p-1}{p}\ \text{ActLatLoad}(w) + \frac{1}{p}\ \text{LatLoad}(w)$$

$$\text{ActLatLoad}(*w) = 1$$

$$\text{ActLatLoad}(0w) = \text{ActLatLoad}(w)$$

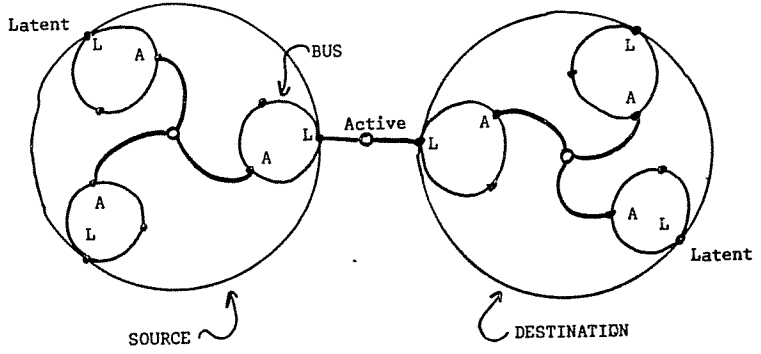
$$\text{ActLatLoad}(1w) = \text{ActLatLoad}(w)$$

ActLatLoad(hw) = 0

All these recurrences have the same flavor; we will describe AACrossLoad(1w) in detail. Figure 5 depicts the situation. The source and the bus are, without loss of generality, in the left-hand cluster, and the destination is in the righthand cluster. Since the bus address begins with 1, it is in the active subcluster of the lefthand cluster. If the source should be in any other subcluster, its message must pass through the subcluster of the bus. It will enter that subcluster by its active corner and leave by its latent corner. Therefore, it will contribute a load of ActLatLoad(w) to the bus. This situation occurs (p-1)/p of the time. If the source is in the active subcluster, then the message will travel out through the subcluster's latent corner, contributing LatLoad(w). This case occurs 1/p of the time. The other recurrences are derived similarly.

It is straightforward to apply these recurrences to compute the expected bus load for any bus in the cluster. In the case n = 3, p = 3, the load on busses is as follows:

Figure 5



AACrossLoad(1w)

bus	Load
*10	.6666667
0*1	.3703704
1*1	.3703704
2*1	.3703704
00*	.1399177
10*	.1399177
20*	.1399177
01*	.4362140
11*	.4362140
21*	.4362140
02*	.1399177
12*	.1399177
22*	.1399177

The maximum bus load appears for the center bus, $*10^{n-2}$.

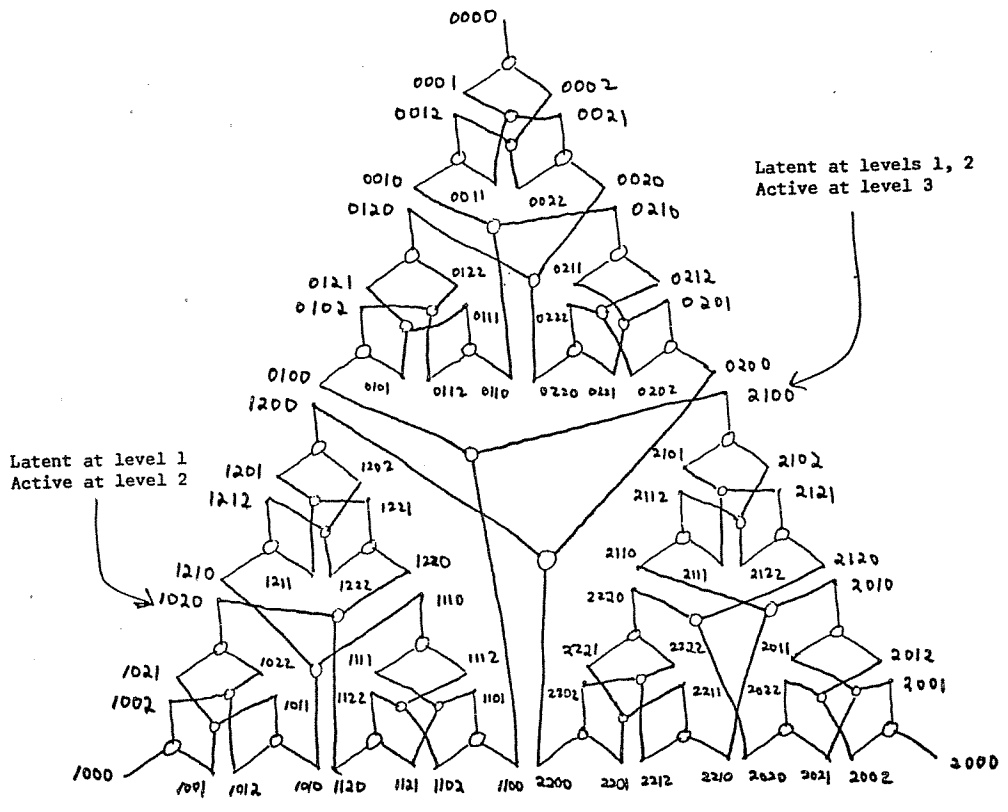
THE DENSE SNOWFLAKE

In our discussion of the snowflake, we noticed that a high proportion of all processors are attached to only one bus. Furthermore, the load on the higher-level busses is substantial. The dense snowflake (which we will usually call the dense flake) attempts to remedy this situation by using p-1 busses instead of only one to connect the p subclusters that make up a level-n cluster. A dense flake for n = 4, p = 3 is shown in Figure 6. Now p-1 corners of a cluster are active and the remaining corner is latent. In a level-j cluster, the latent corner has address 00^{j-1}. The other corners are active, and have addresses $\bar{0}0^{j-1}$, where $\bar{0}$ means any digit but zero.

These new connections introduce alternate pathways between processors. For example, to send a message from node 2101 to node 2021, as we did for the sparse flake, a new path exists through bus 21*2.

In contrast to the sparse flake, in which most processors are on only one bus, all but p processors in a dense flake are on two

Figure 6



Dense Snowflake: n = 4, p = 3

busses. These p defective nodes may be joined by one additional bus with address $*0^{n-1}$, which we call the outer bus. Although this bus completes the structure, it causes some irregularities in routing.

Internode Distance

We will define the distance between two nodes in the dense flake to be the length of a minimal path between them. (There may be several minimal paths. We will discuss routing in a later section.) Let ActCornDist(n) be the average distance from a random node in a level-n cluster to the closest of the p-1 active corners, and let CornDist(n) be the average distance from a random node in a level-n cluster to a given corner. hen we have:

$$\text{ActCornDist}(n) = \frac{1}{p}(\text{ActCornDist}(n-1) + 2^{n-1}) + \frac{p-1}{p}\text{CornDist}(n-1)$$

$$\text{CornDist}(n) = \frac{1}{p}\text{CornDist}(n-1) + \frac{p-1}{p}(2^{n-1} + \text{ActCornDist}(n-1)).$$

These recurrences can be understood in reference to Figure 7a. If a node is picked randomly in a level-n flake, it lies with probability (p-1)/p in a subcluster that owns an active corner of the whole cluster. We will call this an active subcluster. In that case, the distance is CornDist(n-1), since the desired active corner is in the same subcluster. With probability 1/p, the

node is in the latent subcluster, and must pass to the closest active corner of that subcluster and then cross to any of the other subclusters. The distance across any cluster is MaxDist(n) = 2^n-1, and one more link must be added to this quantity to account for the level-n bus that was used. Similar arguments yield the formula for CornDist.

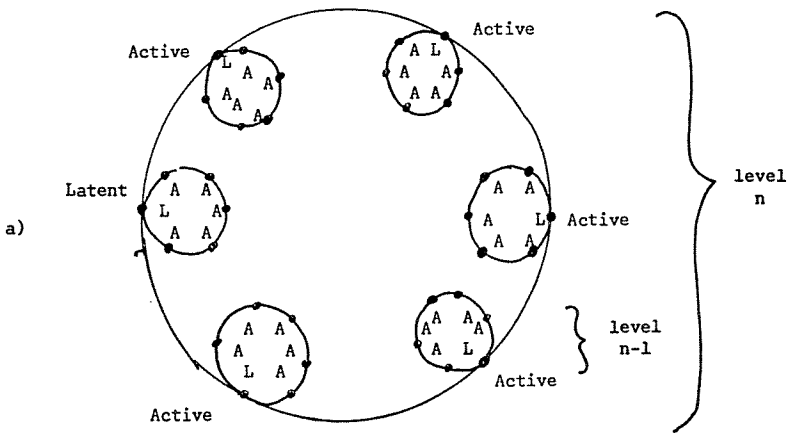
We can define the average distance (without an outer bus) AvgDist(n) in terms of an intermediate quantity we will call the AACrossDist, which is the average distance from a node in one level-n cluster to a node in a different level-n cluster when corresponding active corners of the two clusters are connected by links of length 0. (Hence the name AA.) Figure 7b shows the situation for AACrossDist(n). The average distance is then:

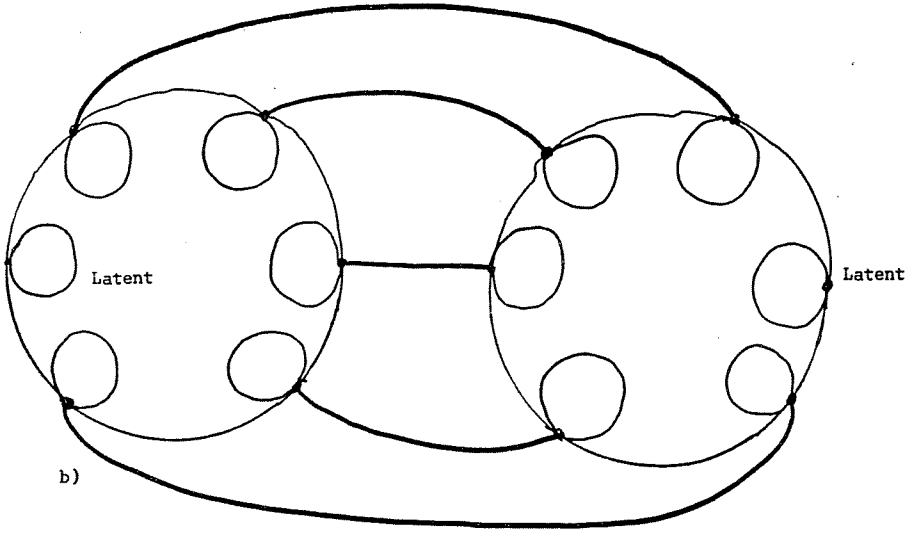
$$\text{AvgDist}(n) = \frac{1}{p}\text{AvgDist}(n-1) + \frac{p-1}{p}(1 + \text{AACrossDist}(n-1))$$

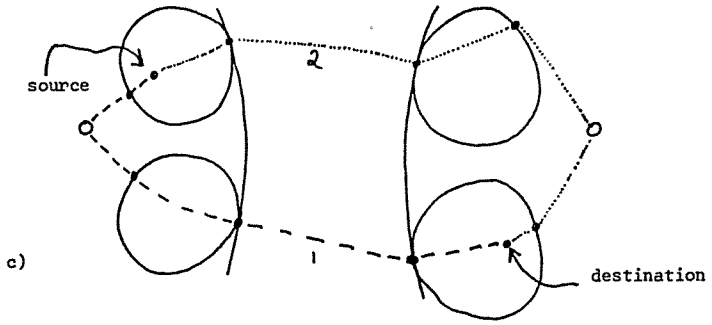
This formula is based on picking two random nodes in a level-n network. With probability 1/p, they fall in the same level-(n-1) subcluster, and with probability (p-1)/p they fall in different ones. In the latter case, a level-n bus must be taken, and the two subclusters are connected as in Figure 7b.

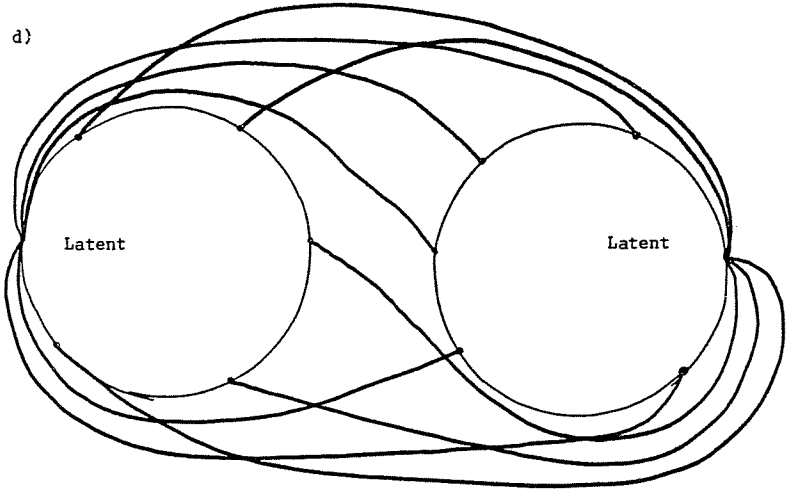
If random nodes are picked from each level-n cluster of Figure 7b, four situations may arise. If both nodes lie in latent subclusters, which occurs $1/p^2$ of the time, the path will lead to any active corner of the source subcluster, out to any active corner of the source cluster, and similarly in the other cluster.

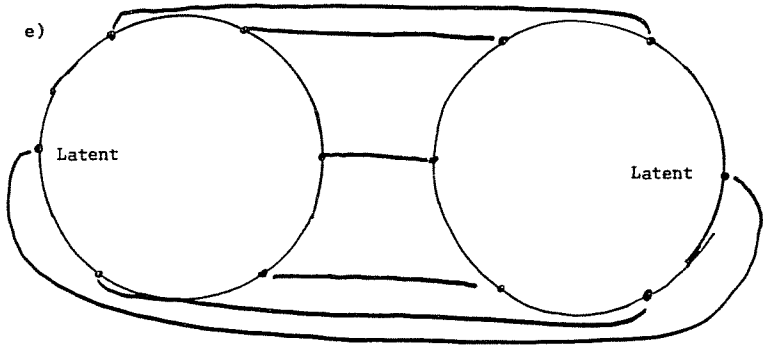
Figure 7











Dense Snowflake Interconnection Structures

The second case is when one node is in a latent subcluster, and the other in an active subcluster. The third case occurs when both nodes are in corresponding active subclusters. The only hard case is when both nodes are in active subclusters, but they are not directly connected. This situation is shown in Figure 7c with the two possible paths that might be taken. Each path has a component of length 2^{n-1} and a remaining portion as shown Figure 7d. We will call the average distance between nodes in clusters connected as in Figure 7d the ALCrossDist(n), since links of length zero connect active corners to latent corners. These four cases give rise to the four terms in the recurrence:

$$\begin{aligned}
\text{AACrossDist}(n) = \frac{1}{p^2} [\ & (2\ \text{ActCornDist}(n-1) + 2\ 2^{n-1}) \\
& + 2(p-1)(\text{ActCornDist}(n-1) + \text{CornDist}(n-1) + 2^{n-1}) \\
& + (p-1)(2\ \text{CornDist}(n-1)) \\
& + (p-1)(p-2)(2^{n-1} + \text{ALCrossDist}(n-1))\ ] \\
= \frac{1}{p^2} [\ & (2p\ \text{ActCornDist}(n-1) + (4p-4)\ \text{CornDist}(n-1) \\
& + (p^2-p+2)\ 2^{n-1} + (p^2-3p+2)\ \text{ALCrossDist}(n-1)\ ].
\end{aligned}$$

The analysis of ALCrossDist(n) is quite similar to that for AACrossDist(n). Three situations can arise when two random nodes are picked from the clusters shown. In the first case, one lies in a latent subcluster and the other in an active subcluster. A link connects these subclusters directly. In the second case, both nodes lie in latent subclusters. Here there are two possible paths, as shown in Figure 7d. Each path has a component of length 2^{n-1} and a remainder corresponding to figure 7d. In the final case, both nodes are in active subclusters, and once again there are two paths. These also share a component of 2^{n-1}, and the remaining connection is again as in Figure 7d. Thus we have:

$$\begin{aligned}
\text{ALCrossDist}(n) = \frac{1}{p^2} [\ & 2(p-1)(2\ \text{CornDist}(n-1)) \\
& + (2^{n-1} + \text{ALCrossDist}(n-1)) \\
& + (p-1)^2\ (2^{n-1} + \text{ALCrossDist}(n-1))] \\
= \frac{1}{p^2} [\ & 4(p-1)\ \text{CornDist}(n-1) + (p^2-2p+2)\ 2^{n-1} \\
& + (p^2-2p+2)\ \text{ALCrossDist}(n-1)].
\end{aligned}$$

Simultaneously solving these recurrences with the aid of Macsyma [Macsyma 75] we derive the formula:

$$\begin{aligned}
\text{AvgDist}(n) = & \frac{p^3 - p}{2\ p^3 + 3\ p^2 - 6\ p + 2}\ 2^n \\
& + \frac{p}{p^2 + 2\ p - 2} \left(\frac{p^2 - 2\ p + 2}{p^2}\right)^n \\
& - \frac{p}{2\ p - 1} \frac{1}{p^n} = O(2^n).
\end{aligned}$$

When p = 3, AvgDist(n) = (24/65)2^n + O(1), which is a 31% improvement over the sparse flake. As p and n increase, AvgDist(n) approaches 2^{n-1}, or half the average distance in the sparse flake.

It may seem that an outer bus could substantially improve interprocessor distances since may pairs of processors that lie near the edges of the flake would be able to use the outer bus rather than route their messages across the diameter. However, the effect of an outer bus is surprisingly small. If AvgDistO(n)

is the average distance in a level-n network with an outer bus, an analysis similar to the one above gives us the recurrences:

$$\text{AvgDist0}(n) = \frac{1}{p}\,\text{AvgDist}(n-1) + \frac{p-1}{p}\,(1 + \text{FCrossDist}(n))$$

$$\text{FCrossDist}(n) = \frac{1}{p^2}\,[2\,\text{CornDist}(n-1) + p(p-1)\,(2^{n-1} + \text{ALCrossDist}(n-1))].$$

Here, FCrossDist is the function suggested by Figure 7e. The solution to these recurrences is:

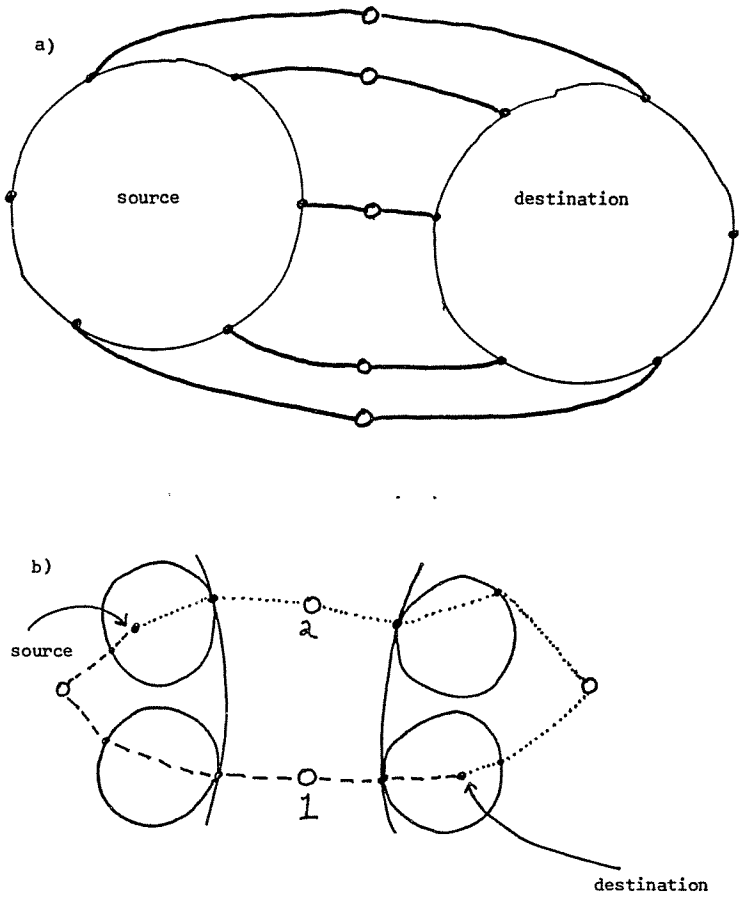
$$\text{AvgDist0}(n) = \frac{p^4 - 3p^2 + 3p - 1}{2p^4 + 3p^3 - 6p^2 + 2p}\,2^n + \frac{p^3}{p^4 - 4p^2 + 8p - 4}\left(\frac{p^2 - 2p + 2}{p^2}\right)^n - \frac{p}{2p - 1}\,\frac{1}{p^n}$$

When $p = 3$, $\text{AvgDist0}(n) = (62/195)2^n + O(1)$, which represents only a 14% improvement over AvgDist(n) for large n, and as p increases, the advantage obtained from the outer bus drops to zero. Since the outer bus leads to special cases, we will usually assume that it has not been included.

Routing

The dense flake allows alternate paths between nodes, so a routing algorithm must be able to find a shortest path. In addition, equally good choices should be equally likely, so that individual busses do not get used more than their share of the time.

Figure 8



Dense Snowflake Routing Decisions

Figure 8a represents the situation in which the source and the destination nodes lie in different clusters. If both nodes lie in corresponding active subclusters, there is no doubt which intercluster bus to take to cross between the source and the destination subclusters. If both nodes lie in latent subclusters, then any one of the intercluster busses may be used with equal cost. In this case, one should be chosen randomly. If one node lies in a latent subcluster, and the other in an active one, then the intercluster bus that leads to the active cluster should be used. The only question arises when the source and destination lie in non-corresponding active subclusters. Figure 8b shows the two paths that might be chosen. The first uses intercluster bus 1, which is more convenient to the destination, and the other uses intercluster bus 2, which is more convenient to the source. Both paths require a traversal of a subcluster, which always takes 2^n-1 steps for a subcluster of level n, and both require both an intercluster bus and an intersubcluster bus. The first path also requires a segment to the closest active corner of its subcluster and from the latent corner of the destination subcluster (which is active in its cluster) to the destination. Let us call these lengths SA and DL, respectively. Likewise, the second path requires DA and SL. The first path should be chosen if SA + DL < DA + SL.

Each of the distances SA, SL, DA, and DL is of a special form, since each measures distance to a corner. It is possible to determine the distance from a given node to the 0-corner quite easily. Figure 6 shows the dense flake for p = 3, n = 4. If we

list all the nodes that lie at various distances from the top corner, we find that all the nodes at each distance have the same form. For example, nodes 0010 and 0020 have distance 3. We will represent them both by 0010. A partial list of distances follows.

distance	bus
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101

This sequence appears in several other guises in the mathematical literature. For example, it is a list of adjacent legal states in the Chinese Ring puzzle, where 0 means a ring is off the pole, and an h means that it is on the pole. (The goal of this puzzle is to start at situation 0000 and reach 1111.) This sequence is the reversing binary number representation of the integers [Knuth 69, p. 178, exercise 4.1.27]. The place value of the j^{th} place from the right (starting with j=1) is 2^j-1. Thus the places represent 1, 3, 7, 15, and so forth. Starting with the most significant 1, the sign of the places occupied is alternated as their contribution is added. Therefore, revvalue(0111) = 7-3+1 = 5, and revvalue(1101) = 15-7+1 = 9. This observation provides an algorithm for computing the distance from any node to a given corner.

The sum of the distances from any node of a level-n cluster to the latent corner and to the nearest active corner is always

2^n-1: One of these distances if revvalue(1w), and the other distance is revvalue(0w) for some string w of length n-1. Thus, SA + SL = DA + DL = 2^n-1, so SA + DL < DA + SL if and only if SA < DA. These observations lead to the following algorithm for routing messages:


```
function pick(place:integer; source, dest:address) : digit;
    {picks which bus to use, out of 1 .. p-1,
    where place = firstdiff(source,dest). }
    var sdigit, ddigit : digit;
        sdist, ddist : integer;
    function dist(place:integer; proc:address):integer;
        begin {dist}
        if place = 0 then dist := 0
        else if proc[place] = 0 then
             dist := dist(place-1,proc)
        else dist := (2**place - 1) - dist(place-1,proc)
        end {dist};
    begin {pick}
    sdigit := source[place-1];
    ddigit := dest[place-1];
    if (sdigit=0) and (ddigit=0)
        then {latent-latent} pick := randomchoice([1 .. p-1])
    else if (sdigit=0) or (ddigit=0)
        then {latent-active} pick := max(sdigit,ddigit)
    else if sdigit=ddigit
        then {matching active-active} pick := sdigit
    else begin {unmatched}
        {This case can only happen once for any message.}
        sdist := dist(place-2,source);
             {The distance is the reversing binary
             interpretation of source[1..place-2]}
        ddist := dist(place-2,dest);
        if sdist < ddist then pick := sdigit
        else if ddist < sdist then pick := ddigit
        else pick := randomchoice([sdigit,ddigit])
        end {unmatched};
    end {pick};
```

The procedure "route" given earlier for routing in the sparse flake may be used for the dense flake by replacing the statement "bus[level-1] := 1" by "bus[level-1] := pick(level,source,dest)". If an outer bus is used, the algorithm changes slightly.

Bus Load

In calculating bus load for the dense flake, we assume that messages follow shortest paths, and that when more than one shortest path exists, all are equally likely. Recurrences similar to those for bus load in the sparse flake apply to bus load in the dense flake. In addition to the quantities described there, it

is also necessary to introduce ALCrossLoad(bus), which describes the load on a bus due to communications between its cluster and another cluster connected to it according to Figure 7d. This quantity is analogous to ALCrossDist(n), where n is the number of digits in the address of the bus. The quantity ActLoad(bus) now accounts for traffic from arbitrary nodes in a cluster to their closest active corner. ActLatLoad(bus) measures load on the bus due to messages between an arbitrary active corner and the latent corner. The recurrences are these:

$$\text{Load}(*) = \frac{p - 1}{p}$$

$$\text{Load}(*w) = \frac{1}{p}$$

$$\text{Load}(dw) = \frac{\text{Load}(w) + 2\,(p - 1)\,\text{AACrossLoad}(w)}{p^2}$$

$$\text{AACrossLoad}(*) = \frac{p^2 - p + 2}{2\,p^2}$$

$$\text{AACrossLoad}(*w) = \frac{p^2 - p + 2}{2\,(p - 1)\,p^2}$$

$$\text{AACrossLoad}(0w) = \frac{\text{ActLoad}(w)}{p}$$

$$\text{AACrossLoad}(\bar{0}w) = \frac{2\,\text{LatLoad}(w) + (p - 2)\,\text{ALCrossLoad}(w)}{p^2} + \frac{p^2 - p + 2}{2\,(p - 1)\,p^2}\,\text{ActLatLoad}(w)$$

$$\text{ALCrossLoad}(*) = \frac{p^2 - 2\,p + 2}{2\,p^2}$$

$$\text{ALCrossLoad}(*w) = \frac{p^2 - 2\,p + 2}{2\,(p - 1)\,p^2}$$

$$\text{ALCrossLoad}(0w) = \frac{(p - 1)\,\text{LatLoad}(w) + \text{ALCrossLoad}(w)}{p^2} + \frac{(p - 1)^2}{2\,p^2}\,\text{ActLatLoad}(w)$$

$$\text{ALCrossLoad}(\bar{0}w) = \frac{\text{LatLoad}(w) + (p - 1)\,\text{ALCossLoad}(w)}{p^2}$$

$$+ \frac{\text{ActLatLoad}(w)}{2\,(p-1)\,p^2}$$

$$\text{ActLoad}(*) = \frac{1}{p}$$

$$\text{ActLoad}(*w) = \frac{1}{p\,(p-1)}$$

$$\text{ActLoad}(0w) = \frac{\text{ActLoad}(w)}{p}$$

$$\text{ActLoad}(\bar{0}w) = \frac{\text{LatLoad}(w)}{p} + \frac{\text{ActLatLoad}(w)}{p\,(p-1)}$$

$$\text{LatLoad}(*) = \frac{p-1}{p}$$

$$\text{LatLoad}(*w) = \frac{1}{p}$$

$$\text{LatLoad}(0w) = \frac{\text{LatLoad}(w) + (p-1)\,\text{ActLatLoad}(w)}{p}$$

$$\text{LatLoad}(\bar{0}w) = \frac{\text{ActLoad}(w)}{p}$$

$$\text{ActLatLoad}(*) = 1$$

$$\text{ActLatLoad}(*w) = \frac{1}{p-1}$$

$$\text{ActLatLoad}(0w) = \text{ActLatLoad}(w)$$

$$\text{ActLatLoad}(\bar{0}w) = \frac{\text{ActLatLoad}(w)}{p-1}$$

For the case n = 3, p = 3, the bus loadings are as follows:

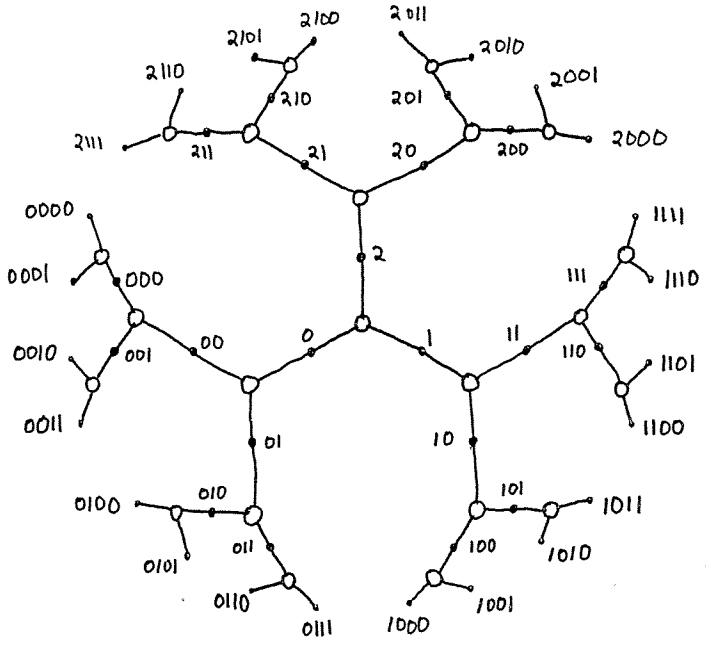
bus	load
*0̲0̲	.3333333
0*0̲	.1358025
1*0̲	.1358025
2*0̲	.1358025
00*	.0795610
10*	.0795610
20*	.0795610
01*	.2085048
11*	.2085048
21*	.2085048
02*	.2085048
12*	.2085048
22*	.2085048

THE STAR

The snowflake has many processors that perform no routing functions, since they are connected to a single bus. When extra connections are introduced, as in the dense snowflake, uniqueness of routing is lost. It is of some interest to see how small the average internode distance can be made without introducing alternate pathways through the cluster.

To build an optimal cluster in this sense, start with a single bus with p processors. This bus will be called the <u>center</u> of the cluster, and its p processors form the first <u>ring</u>. Since we wish to fully connect each processor, we form the second ring by attaching a new bus to each of the p processors. Each of these second-ring busses should have a total of p processors. One of them is the <u>inner</u> node to which the bus is connected. The other p-1 nodes are <u>outer</u> nodes, and will be used to form the next ring. A level-n <u>star</u> is formed by the combination of all rings 1 to n. Figure 9 shows a star with n = 4, p = 3.

Figure 9



The Star

p = 3, n = 4

Number of busses and processors

The number of busses in each ring follows this recurrence:

$$\text{RingBus}(r) = \begin{cases} (p-1)\ \text{RingBus}(r-1) & \text{if } r \geq 3 \\ p & \text{if } r = 2 \\ 1 & \text{if } r = 1 \end{cases}$$

$$= \begin{cases} p\ (p-1)^{r-2} & \text{if } r \geq 2 \\ p & \text{if } r = 1. \end{cases}$$

The total number of busses in a level-n star is the sum of these quantities, namely

$$\text{NetBus}(n) = \frac{p\ (p-1)^{n-1} - p}{p-2} + 1$$

The number of processors in each ring is given by

$$\text{RingProc}(r) = \begin{cases} (p-1)\ \text{RingBus}(r) & \text{if } r \geq 2 \\ p & \text{if } r = 1 \end{cases}$$

$$= \begin{cases} p\ (p-1)^{r-1} & \text{if } r \geq 2 \\ p & \text{if } r = 1. \end{cases}$$

The total number of processors in the star is given by

$$\text{NetProc}(n) = 1 + (p-1)\ \text{NetBus}(n)$$

$$= \frac{p\ (p-1)^{n} - p}{p-2} + 2$$

The number of processors in a level-n star is thus approximately $(p-1)^n$. The maximum distance is MaxDist(n) = 2n - 1. The formulas given above become simpler if the level 1 bus only has p-1 processors associated with it. We call this structure the partial star. In this case, we get the following results:

$$RingBus(n) = (p - 1)^{n-1}$$

$$NetBus(n) = \frac{(p-1)^n - 1}{p - 2}$$

$$RingProc(n) = (p - 1)^n$$

$$NetProc(n) = \frac{(p-1)^{n+1} - 1}{p - 2}$$

Even though the maximum distance between two processors is quite small, and there are no alternate paths, physical crowding among outlying processors becomes severe. This situation is due to the goal of the star: to place as many processors as close together as possible. If the physical radius of the network is to increase linearly with the number of rings, then a three dimensional space can accomodate a quadratic increase in the number of processors per additional ring. However, the number of processors in a star is exponential in the number of rings, and therefore cannot fit without either increased crowding or physically broader and broader rings.

Routing and addresses

The star has an exceptionally easy addressing and routing scheme. A processor of ring r will have an r-digit address. Each of the processors on the next ring out will have the same address followed by one of the digits 0 to p-1. The numbering scheme is shown in Figure 9.

A node at ring r can receive a message either from its inner bus or its outer bus. If the message comes from the inner bus, either it has r digits in its address, in which case it is intended for the node itself, or it has more. In the latter case, the node directs the message to the appropriate node in the next outer ring by examining the (r+1)th digit from the right in the address. If the message comes from the outer bus, it might be for the node itself, in which case the address will match exactly, or it might be for some other node. In the latter case, the message is drifting inwards until it finds the level at which it can begin to travel back out. This reversal will happen at the node whose address matches the destination address for all but one place in the node address. If this criterion is met, the node directs the message through the inner bus back out to the node on its ring with the proper final digit. If this criterion is not met, the node directs the message through the inner bus to the node of the next inner ring.

Bus load

As before, we are interested in how many processors lie in each direction from any given bu. Because of the central symmetry of the star, the calculation will yield the same result for any bus in the same ring r.

The p nodes on a bus in ring r > 1 lie in two directions: one is inward, and p-1 are outward from this ring. Let Out(n,r) represent the number of nodes reachable in a level-n star from a ring-r bus along one of its outward nodes. Then

$$Out(n,r) = \sum_{1 \le k \le n-r} (p-1)^k = \frac{(p-1)^{n-r+1} - 1}{p - 2} .$$

Likewise, we can define In(n,r) to be the number of processors that can be reached by along the path inward from a ring-r bus in a level-n star. For the sake of mathematical simplicity, we will deal with the partial star. We then derive that:

$$In(n,r) = \begin{cases} 1 + (p-2)\, Out(n,r-1) + In(n,r-1) & \text{for } r > 1, \\ 0 & \text{for } r = 1. \end{cases}$$

$$= \frac{(p-1)^{n+1}}{p - 2} - \frac{(p-1)^{n-r+2}}{p - 2} \quad \text{for } r >= 1.$$

Now we can express the load through a ring r bus in a level-n star by computing:

$$p^{2n}\, Load(r) = (p-1)(p-2)\, Out(n,r)^2 + 2(p-1)\, Out(n,r)\, In(n,r)$$

$$= (p-2)^{-1}\left[(p-1)^{2n-2r+3} - 2(p-1)^{n-r+2} + p - 1\right]$$

$$+ 2(p-2)^{-2}\left[(p-1)^{2n-r+3} - (p-1)^{n+2} - (p-1)^{2n-2r+4} + (p-1)^{n-r+3}\right]$$

The extreme values of Load(r) occur for r = 1 and r = n. At these points, we find that

$$Load(1) \equiv 2\,\frac{(p-1)^{2n}}{p^{2n}}$$

$$Load(n) \equiv 2\,\frac{(p-1)^{n+1}}{p^{2n}}$$

Extensions

All the processors on the outermost ring of a star are still connected to only one bus. There are several ways to close the graph. Sutures could be made from each outermost node to p-1 other leaf nodes in other arms of the star. A processor at the

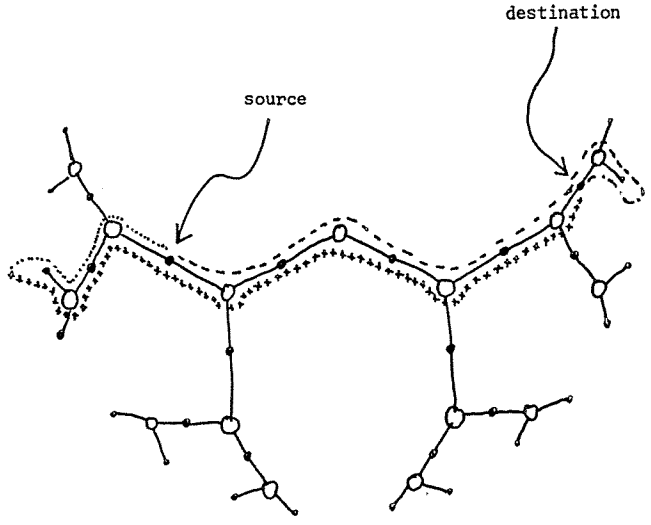
edge will have address $D_1D_2...D_r$ for an r-ring network. The neighbors across the rift have address $d_1D_2...D_r$, where $d_1 \neq D_1$. There are RingProc(r)/p = $(p-1)^{r-1}$ suture busses needed for this connection when $r \geq 2$. We call this form of network the cross-star.

Another way to close the graph is to form a nova from p identical copies of a star placed over each other so that the edges are aligned. These copies are sutured together at the edge by $p(p-1)^{r-1}$ suture busses. Now each outermost node has found its p-1 mates, which have, in fact, the same address. It will be necessary to add one digit to the address of each node to indicate on which of the p planes it lies. Each processor has p-1 shadow processors with the same address in the other planes.

The algorithm for routing messages must deal with the alternate pathways through the structure. Since the initial purpose of the star was to see how densely processors could be arranged without alternate pathways, the extension under consideration is no longer in the mood of the previous discussion. However, it is an interesting exercise to find a good routing strategy that will deliver any message across the shortest path that leads from the source to the destination.

Suppose that two nodes wish to communicate. If they lie on the same plane, then they follow the path dictated by the routing algorithm outlined above. If they inhabit different planes, the message can cross the suture either outward from the source or outward from the destination. Define the parent ring for the source-destination pair as the outermost ring whose addresses

Figure 10



path one, source plane
path one, destination plane
path two, source plane
path two, destination plane

SO = 2, SI = 2, DO = 1, DI = 3, PO = 4.
Path one has 7 links; path two has 9 links

Routing in the Nova

form a prefix for the address of both the source and the destination. One way to send the message is for the source to direct it inwards to the parent ring, then to lead it outwards past the destination's shadow (since it is still in the wrong plane) and then randomly out to the edge. It then crosses the suture into the destination plane and comes back in to the destination. The other path directs the message outward from the source randomly to the edge, across to the destination plane, back in past the source's shadow, in to the parent ring, and then out to the destination. These alternate paths are depicted in Figure 10. If either the source or the destination lies on the parent ring, these two methods will yield the same path (if random motion outward means to pass the shadow if it lies outward of the current location). In other cases, the paths are different. The lengths can be expressed in terms of several ring differences. Let SI be the ring difference from the source in to the parent. Let SO be the ring difference from the source out to the edge. Define DI and DO similarly. Let PO be the ring difference from the parent out to the edge. Then the first type of path has this distance:

Dist(1) = SI + PO + DO.

The other path has this distance:

Dist(2) = SO + PO + DI.

Path one is to be preferred when SI + DO < SO + DI. Since PO = SI + SO = DI + DO, path one is better when SI < DI. A simi-

lar computation applies to the case of the cross-star.

THE P-CUBE

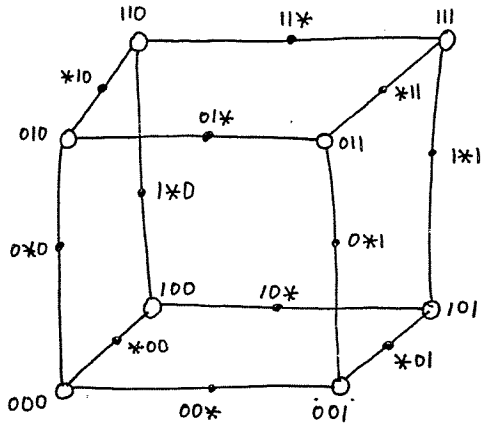
We examined the star to find the smallest diameter in the graph without introducing alternate paths. Now we would like to examine a structure that minimizes internode distance, but allows multiple paths, while maintaining the constraints that each bus have exactly p processors and each processor be connected to exactly two busses. Arrange 2^p busses at the vertices of a p-dimensional unit cube. The three-dimensional case is shown in Figure 11. Each bus can be addressed by its position in space, which is a sequence of p binary numbers. Adjacent busses are those differing by one digit. Between each pair of adjacent busses sits a processor, whose address is the same as that of each of its busses, except it holds the character "*" in that position in which the busses differ. (This nomenclature is dual to the form that we have been following so far.) The only way to enlarge a p-cube is to increase p. The number of processors is the number of edges in a p-cube, namely

$$\text{ProcNum}(p) = p\, 2^{p-1} .$$

Internode distance

It is more convenient in this case to discuss the interbus distance. The distance between two busses is the Hamming distance (the 1-norm distance) between their addresses. It is clear then that

Figure 11



P-Cube, p = 3

MaxDist(p) = p.

The average distance between two busses can be found by noticing that there are exactly C(p,d) busses at a distance d from any bus, where C(n,k) is the binomial coefficient of n and k. Thus the average distance is

$$\text{AvgDist}(p) = \sum_{0 \le d \le p} d\, C(p,d) = p/2.$$

Routing

It is very straightforward to route messages in the p-cube. The source node has an address formed of a binary string with a "*" in it. Either the destination is the same, in which case the message has been delivered, or there is a difference in some position. Choose arbitrarily any position of difference. Turn the * into the proper digit for the destination at that position (thereby determining which bus to take) and replace the difference bit with a new *, thereby determining which of the p nodes to go to on the other side of that bus. If there are d differences in the addresses, then there are d! ways to choose the order in which the discrepant digits can be fixed. Each of these ways forms a different path, although many paths share common subpaths. Any path will go through d busses, but there are 2^d busses that could be chosen.

Bus Load

Since all busses are functionally identical, the fraction of messages that crosses any bus is the inverse of NumBus, so we get

Load(bus) = 2^{-p} for each bus.

CONCLUSION

We have compared several related topologies for interconnecting large collections of identical processors and evaluated them with respect to various combinatoric properties. The highlights of our results are summarized in the table at the end of this section.

Several questions remain unanswered. The recurrences that describe bus load in the snowflake might be solvable, and the problem of bus load in the dense snowflake seems substantially more difficult. More generally, we would like to derive statistics about the distribution of bus loads in the sparse and dense snowflakes. The authors are currently attacking this area.

An issue not addressed in this paper is the survivability of various topologies in the event of component failures. The regularity of the topologies we describe should make answers to this question follow easily from general graph theoretic techniques such as those summarized in [McQuillan 77]. For example, the star and the sparse flake are both trees. Therefore, failure of any bus disconnects the network. The dense flake and the hypercube are much better in this respect.

Although we feel that the structures we describe are natural

consequences of the goals mentioned in the introduction, we make no claim that they are exhaustive. There may very well be other topologies that perform better than the ones we considered. We hope that the techniques presented here will prove useful in the investigation of other structures. For example, if we alter the model to assume that processors are connected by links rather than busses, with at most p links from any processor, we are led to a graph very similar to Figure 1c. Interpreting the circles as processors rather than busses, we can calculate interprocessor distances in the processor-link model as interbus distances in the processor-bus model.

The biggest problem involved in the construction of a mega-micro computer is software. Very little is known about the task of writing operating systems to make such a multi-processor usable except that the task is hard [Wulf 1974, Ornstein 1975]. The authors hope to tackle this problem in the near future. In the meantime, we hope the present work will be helpful to the designers of hardware.

	Sparse Flake	Dense Flake	Star	P-Cube
Number of processors	p^n	p^n	$\frac{(p-1)^{n+1}-1}{p-2}$	$p\ 2^{p-1}$
Number of busses	$\frac{p^n-1}{p-1}$	$2p^n - 1$	$\frac{(p-1)^n-1}{p-2}$	2^p
Distances				
max	2^n-1	2^n-1	$2n - 1$	p
mean	$\sim 2^n$	$\sim 2^{n-1}$	$\sim 2n - 2$	$p/2$
Bus load				
max	$\frac{p-1}{p}$	$\frac{1}{p}$	$\sim 2\ \frac{(p-1)^{2n}}{p^{2n}}$	2^{-p}
min	?	?	$\sim 2\ \frac{(p-1)^{n+1}}{p^{2n}}$	2^{-p}
Uniformity among processors	good	good	very good	excellent
Crowding	mild	mild	very severe	severe
Ease of routing	good	fair	good	excellent

REFERENCES

Knuth, D. E. Seminumerical Algorithms, Addison-Wesley, 1969.

Macsyma Reference Manual, Massachusetts Institute of Technology, November 1975. (The work of the Mathlab group is supported by the Defense Advanced Research Projects Agency work order 2095, under the Office of Naval Research Contract #N00014-75-C-0661.)

McQuillan, J. M. "Graph Theory Applied to Optimal Connectivity in Computer Networks," Computer Communication Review (SIGCOMM), 7,2 (April 1977).

Ornstein, S. M., Crowther, W. R., Kraley, M. F., Bressler, R. D, Michel, A., and Heart, F. E. "Pluribus -- A Reliable Multiprocessor," Proceedings of the National Computer Conference, AFIPS press, 1975.

Wittie, L. D. Efficient Message Routing in Mega-Micro-Computer Networks, State University of New York at Buffalo Technical Report, 1976.

Wulf., W., et al, "HYDRA: The Kernel of a Multiprocessor Operating System," Communications of the ACM 17, 6 (June 1974).