Self-Routing Lowest Common Ancestor Networks

Chi-Kai Chien and Isaac D. Scherson

Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717
(714) 856-8144
chikai@ics.uci.edu
isaac@ics.uci.edu

Technical Report #92-31

April 1, 1992

Self-Routing Lowest Common Ancestor Networks *

Chi-Kai Chien and Isaac D. Scherson
Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717
(714) 856-8144
chikai@ics.uci.edu
isaac@ics.uci.edu

Abstract

Multistage interconnection networks (MIN's) allow communication between terminals on opposing sides of a network. Lowest Common Ancestor Networks (LCAN's) [1] have switches capable of connecting bi-directional links in a permutation pattern that additionally permits communication between terminals on the same side. *Self-routing LCAN's* have interesting permutation routing capabilities and are highly partionable. This paper characterizes self-routing LCAN's and analyzes their permutation routing capabilities. It is shown that the routing network of the CM-5 is a particular instance of an LCAN.

Keywords: SIMD, interconnection network, self-routing, permutation routing

*This research was supported in part by the Air Force Office of Scientific Research under grant number AFOSR-90-0144 and NSF under grant number MIP9106949.

1 Introduction

Multistage interconnection networks (MIN's) can be used to effect communication between processing elements (PE's) in SIMD machines, as well as MIMD machines. SIMD communication takes the form of *permutation routing* where given a unique labelling of the PE's, source-destination pairs are obtained from a one-to-one mapping of the set of labels onto itself. MIN's differ in many respects, e.g. permutation routing capabilities, network topology, and switch size [3], [5], [7], [18], [19]. However, all MIN's share one characteristic: the routing of a source-destination pair progresses in only *one* direction through the network (this is called *uni-directional routing*) and must utilize a switch in *every* stage. This characteristic is called the MIN "model of interconnectivity."

Lowest Common Ancestor Networks (LCAN's) [1] permit routing in both directions across a network (this is called *bi-directional routing*). In this paper LCAN's are re-characterized and the advantages of their model of interconnectivity over the MIN model of interconnectivity are demonstrated. *Self-routing LCAN's* are classified and shown to possess interesting permutational capabilities. They are also shown to be easily partitioned hierarchically.

Self-routing LCAN's exploit a communication hierarchy imposed by the actual implementation of massively parallel computers. The MIN model of interconnectivity does not inherently allow utilization of the hierarchy. The hierarchy is exemplified by the MasPar MP-1: the MP-1 is constructed with a number of boards attached to a backplane; each board holds a number of chips; and each chip holds a number of PE's [4] (see Figure 1). Due to physical limitations, this construction leads to a natural hierarchy with respect to communication time: on-chip communication is performed the quickest, then on-board communication, and lastly, communication requiring the backplane. Connecting boards to the backplane is also expensive in terms of pinouts; this is because of physical limitations upon the number of pinouts per board.

Typically, using a MIN for interconnection forces *every* permutation to require backplane communication, since each source-destination pair utilizes a switch in every stage. The LCAN

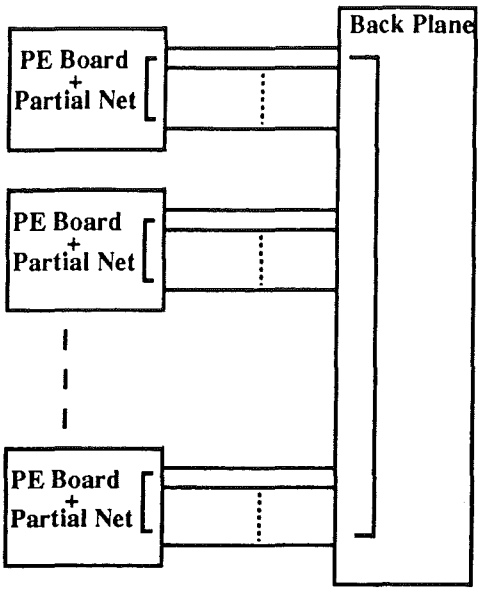

Figure 1: Router network construction

model of interconnectivity forces off-chip and off-board communication only when necessary. As an extreme example, consider the identity permutation: why should each source PE need to find a path through the entire network to its destination PE? Using LCAN's, each source-destination pair need only route to the first stage of the network and back; thus, only on-chip communication is necessary.

An LCAN switch in stage i has d links to stage $i-1$ and u links to stage $i-1$. Previous research on networks allowing bi-directionality between stages can be categorized as comprising either switches with $d > u$ or $d = u$, but not $d < u$. In both cases, the networks are used not in SIMD context, but rather MIMD, i.e. they have strategies for PE-to-PE routing, not permutation routing.

X-tree, the DAC, and the P-tree [6], [10], [11], [12] employ a tree of processors, and Hypertree, KYKLOS and fat-trees [8], [15], [14] use a tree of switches as an interconnection network between processors. LCAN's with switches where $d > u$ fall into the latter category. Trees are attractive to use as interconnection networks because they are highly scalable and require a number of switches that is linear with respect to the number of PE's. However, trees are unappealing because of the high degree of contention near the root of

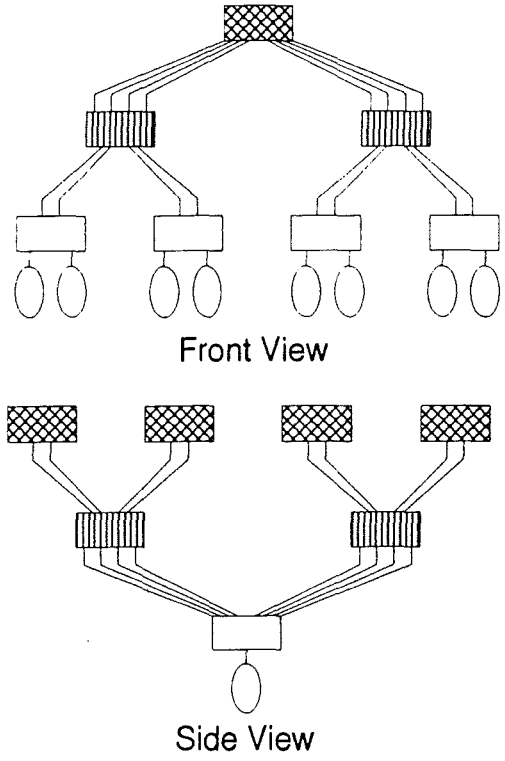

Figure 2: Front and side views of LCAN(2,2,8,3,(2, 2) – *complete bipartite*) drawn in 3-D

the tree. Hypertree and X-tree are variations upon complete binary trees, both networks add links between nodes on the same level to reduce the average interprocessor distance. KYKLOS comprises replications of k-ary trees; the isomorphic replications provide alternate paths, again reducing the average interprocessor distance. Fat-trees reduce root contention by increasing the "capacity" of the links (i.e. each link has more bandwidth) progressively approaching the root switch in a binary tree. However, they require switches that proportionally increase in size. In [6], [8] and [15], X-tree, Hypertree and KYKLOS were analyzed in a MIMD context.

Thinking Machines Corporation's CM-5 is a MIMD computer using a router network they call a "hyper-tree" [17]. On the bottom level, each PE connects to two switches; all switches are identical. Each switch upwardly connects to either two or four switches, each via one link (likewise for the downward links). In higher stages, some switch connectors are not used, thereby decreasing the bandwidth. The router network is similar to LCAN(2,2,8,3,(2, 2) –

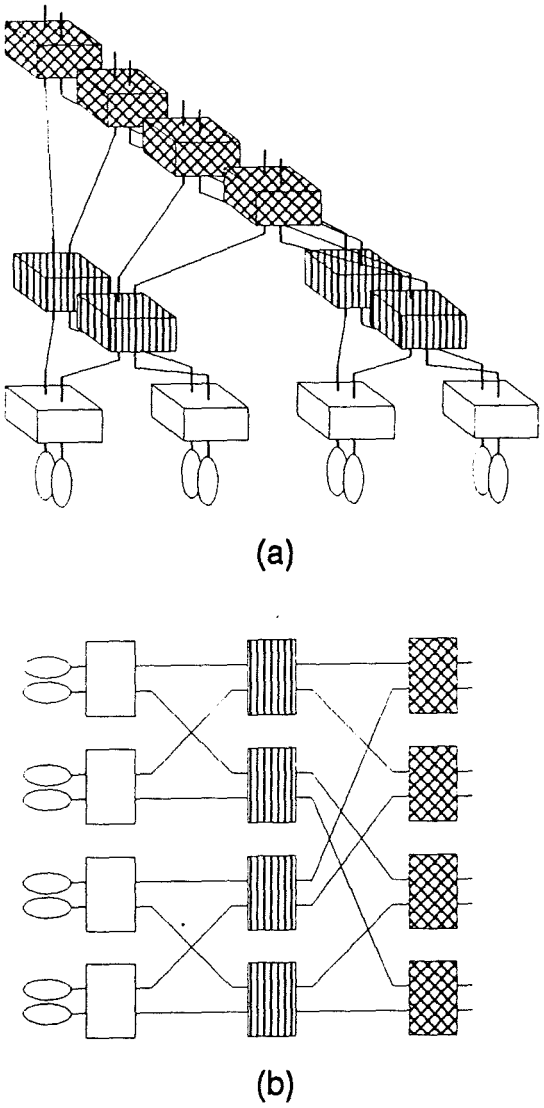

Figure 3: (a) 3-D view of LCAN(2,2,8,3,(2, 2) – *complete bipartite*), (b) the same network drawn in two dimensions

complete bipartite) shown in Figures 2 and 3. The CM-5 network is a particular instance of LCAN's where $d = u = 4$.

Full communication, a network's ability to connect one terminal to any terminal on either side of the network was introduced in [20] (in this paper, full communication is called bi-directional routing). A routing strategy for full communication using a baseline network was described; a baseline network is a particular instance of LCAN's where $d = u = 2$.

In [1] a permutation routing algorithm for LCAN's where switches with $d > u$ was presented and analyzed. Now, in this paper, LCAN's with all switches of all sizes are analyzed. Section 2 characterizes self-routing LCAN's, and uni-directional and bi-directional routing schemes for them are explained in Section 3. Section 4 presents permutation routing algorithms, Section 5 shows the analyses, and Section 6 concludes.

2 Characterization

2.1 LCAN's

This section re-characterizes LCAN's from [1] and explains their advantages. Then, a class of LCAN's, *self-routing LCAN's*, is parameterized. In [1], LCAN's were parameterized by $(u, d, n, l, \overline{SP})$-tuples. There are n PE's, and l levels (stages) of switches in the network. $\overline{SP}$ is a vector of size l, each vector element describing the permutations between stages, the manner in which the switches in adjacent stages are connected. Each switch in level i, where $0 \leq i \leq l-1$, has d bi-directional downers, links which connect to switches in the next lower level, level $i-1$, and u bi-directional uppers which connect to switches in the next higher level, level $i+1$ (see Figures 4 and 5). Thus, switches in level i only connect to switches in levels $i-1$ and $i+1$. The exceptions are level 0 (the lowest level in the network), in which case the uppers do not connect to anything, and level $l-1$ (the highest level in the network), in which case the downers connect to the PE's. Note that the levels are now re-labeled; previously in [1], level 0 was the highest level. A switch accepts up to

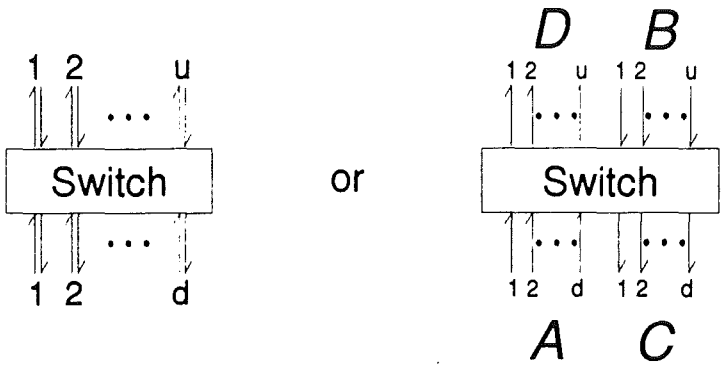

Figure 4: An LCA switch

$u + d$ inputs and switches them to $u + d$ outputs. Any upper and downer can connect to any upper or downer, including itself. An example LCAN is shown in Figure 6, its stage permutation $(2, 3)$ – *complete bipartite* is defined later in this paper.

LCAN's were characterized as networks with identical switches. However, it has become clear that the key advantage of LCAN's lies in the proper usage of the bi-directional links. Non-identical switches in an LCAN do not preclude this usage. Thus, LCAN's are now re-characterized such that every switch need not have the same number of bi-directional connectors (d and u). At the same time however, attention is restricted to LCAN's with identical switches so that the development and analyses of permutation routing schemes are simplified.

With respect to a tree, the lowest common ancestor of two nodes (PE's) is the node (a switch) at greatest depth which counts both nodes (PE's) among its descendants, this node corresponds to an *LCA switch.* All LCAN's exhibit the following characteristic: given a source-destination pair, there are one or more LCA switches, and communication need progress upwards only to a stage containing an LCA switch, this stage is called the *LCA level.* At this point, routing can return downwards to the destination. Thus, given a network, there clearly exist source-destination pairs that do not require routing through *every* stage of the network; whereas in MIN's, all source-destination pairs must route through every stage.

This common LCAN characteristic is useful because it permits the network to be easily

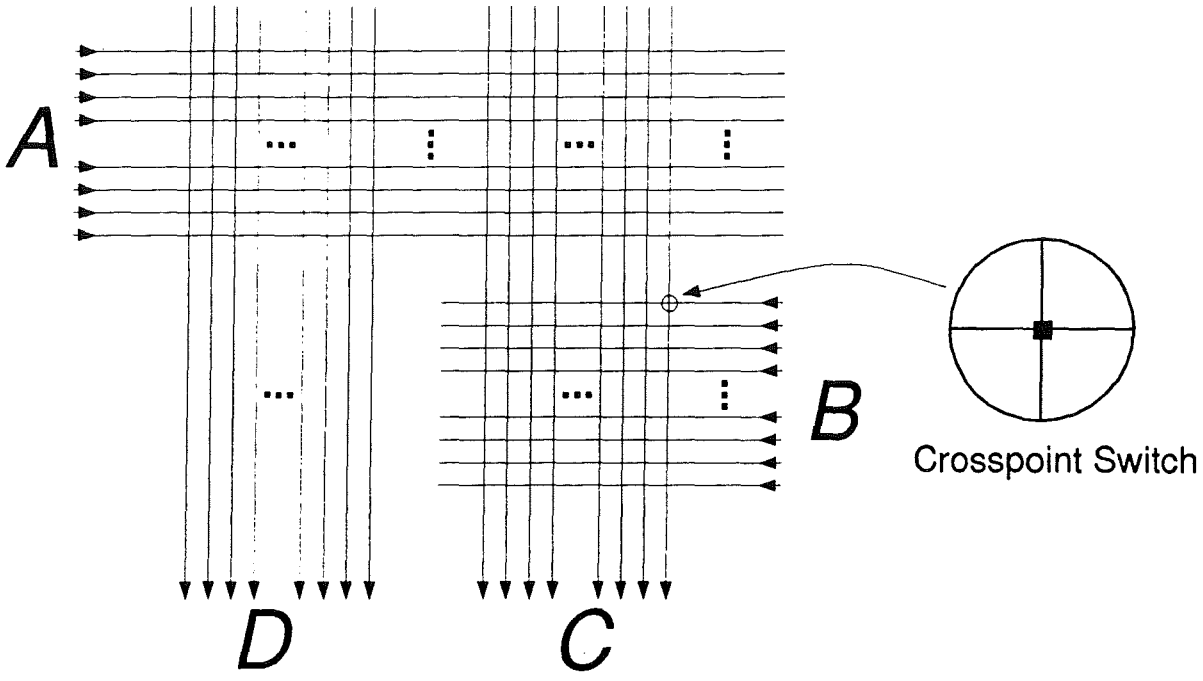

Figure 5: LCAN switch showing crosspoints

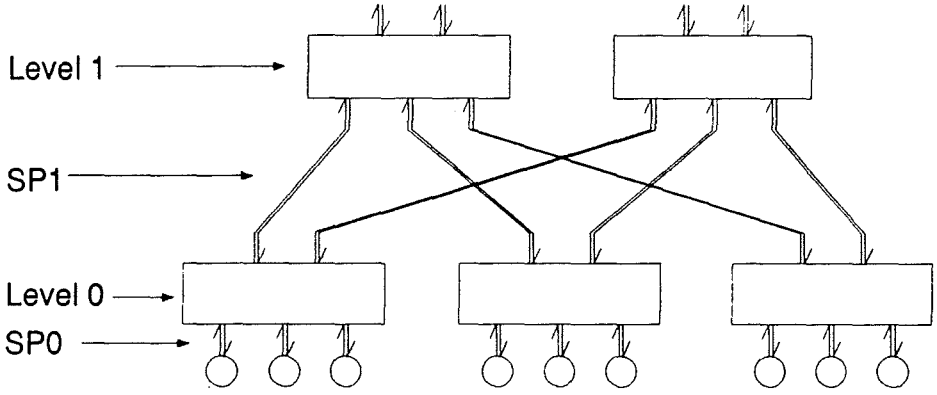

Figure 6: An example LCAN$(2, 3, 9, 2, (2, 3)$ – *complete bipartite*)

partionable onto chips and boards in a manner that minimizes the number of pinouts per board and allows certain permutations to be performed quickly (permutations that only require on-chip and on-board communication; determination of these permutations depends on a given partioning). Even permutations that require backplane communication gain: source-destination pairs not routing through the uppermost stage route downwards at some lower stage, thus reducing link contention at the upper switches.

Exploiting this characteristic requires determining network parameters that permit routing to and from an LCA switch, based solely on local knowledge, i.e. the source and destination addresses. Without the ability to determine when an LCA switch has been reached, all source-destination pairs must route through the topmost level, then return downwards - negating the advantages of LCAN's.

2.2 Self-Routing LCAN's

A MIN has a set of terminals at both sides of the network, the two sets correspond to one set of PE's. Uni-directional routing is allowed, but bi-directional routing is not. Certain MIN's, *delta networks* [16] are capable of self-routing (i.e. require only a destination address to route). A subclass of delta networks, *bidelta networks* [13] are also self-routing if the network is reversed. *Self-routing LCAN's* are topologically similar to delta and bidelta networks, however they additionally allow bi-directional routing, using only the source and destination addresses.

In [1] it was shown that LCAN's with identical switches with $d > u$ and a "tree" stage permutation allowed simple routing to and from LCA switches. Now, a generalized stage permutation, called (d, u)-*complete bipartite*, that applies to all switch sizes ($d > u$, $d = u$ and $d < u$) is described. Networks with this stage permutation are self-routing both up and down the network, and have $l = log_d\, n$ stages. Thus, self-routing LCAN's are parameterized by (u,d,n)-tuples, where $n = d_k$ for some integer k. Each stage i has S_i switches, where $S_i = n \cdot \frac{u^i}{d^{i+1}}$, where $0 \leq i \leq l-1$ Stage $l-1$ is the highest stage and stage 0 is the lowest stage. For a given u, d and n, the total switch cost $= \sum_{i=0}^{log_d\, n-1} S_i$.

$\overline{SP} = (d, u)$ – *complete bipartite* connects d switches in stage i and u switches in stage $i+1$ in a complete bipartite fashion, i.e. there is exactly one link between each switch of the first set and each switch of the second set. Properly choosing which switches to connect in this fashion leads to self-routing properties.

Label the PE's consecutively using $log_d\, n$ digits base d: $< p_{log_d n-1} p_{log_d n-2} \cdots p_0 >$. Label the switches in stage i consecutively using $log_d\, n - 1$ digits such that the $log_d\, n - 1 - i$ most significant digits are base d and the i least significant digits are base u: $< w_{log_d n-2} w_{log_d n-3} \cdots w_i w_{i-1} \cdots w_0 >$.

Definition 1 *The stage permutation of an LCAN is (d,u)-complete bipartite iff 1) switches in stage i,* $0 \leq i < log_d\, n - 1$, *labelled* $< w_{log_d n-2} w_{log_d n-3} \cdots w_{i+1} j w_{i-1} \cdots w_0 >$, *where* $j = 0, 1, ..., d-1$, *are connected to switches in stage* $i+1$ *labeled* $< w_{log_d n-2} w_{log_d n-3} \cdots w_{i+1} w_{i-1} \cdots w_0 k >$, *where* $k = 0, 1, ..., u-1$, *via the* $j+1$*st upper of the switch in stage i and the* $k+1$*st downer of the switch in stage* $i+1$*; and 2) a PE labelled* $< p_{log_d n-1} p_{log_d n-2} \cdots p_0 >$ *is connected to switch* $< p_{log_d n-1} p_{log_d n-2} \cdots p_1 >$ *via the* p_0*th downer of the switch.*

Baseline networks [20] are a particular instance of LCAN's with $d = u = 2$ and $\overline{SP} = (d, u)$ – *complete bipartite.* An example of self-routing LCAN's where $d = u$ is shown in Figure 7. Examples of self-routing LCAN's with $d > u$ and $d < u$ are shown in Figures 8 and 9, respectively. The examples show the inherent partitionality of self-routing LCAN's. The interconnection networks in the last two examples are isomorphic to each other, thus self-routing LCAN(d,u,n) is the same as self-routing LCAN(u,d,$n \cdot (u/d)^{log_d n}$).

3 Routing

Routing in the baseline network is accomplished using routing tags in base two. In the generalized network for all $d = u$ where $\overline{SP} = (d, d)$ – *complete bipartite*, routing tags are in base d. The generalized networks with $d > u$ or $d < u$ and $\overline{SP} = (d, u)$ – *complete bipartite* facilitate routing upwards using routing tags in base u and downwards with routing tags in

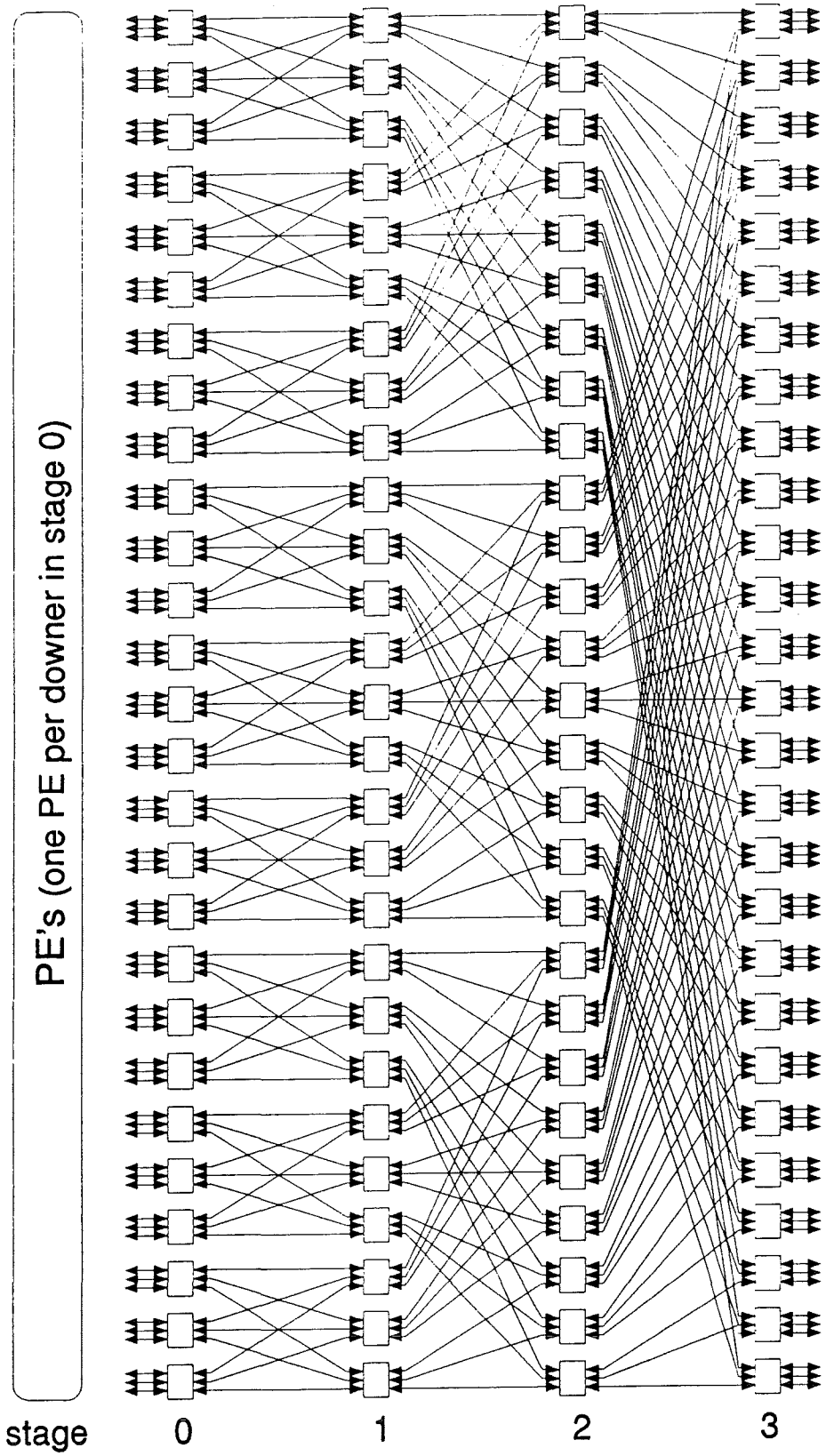

Figure 7: A self-routing LCAN (3,3,81)

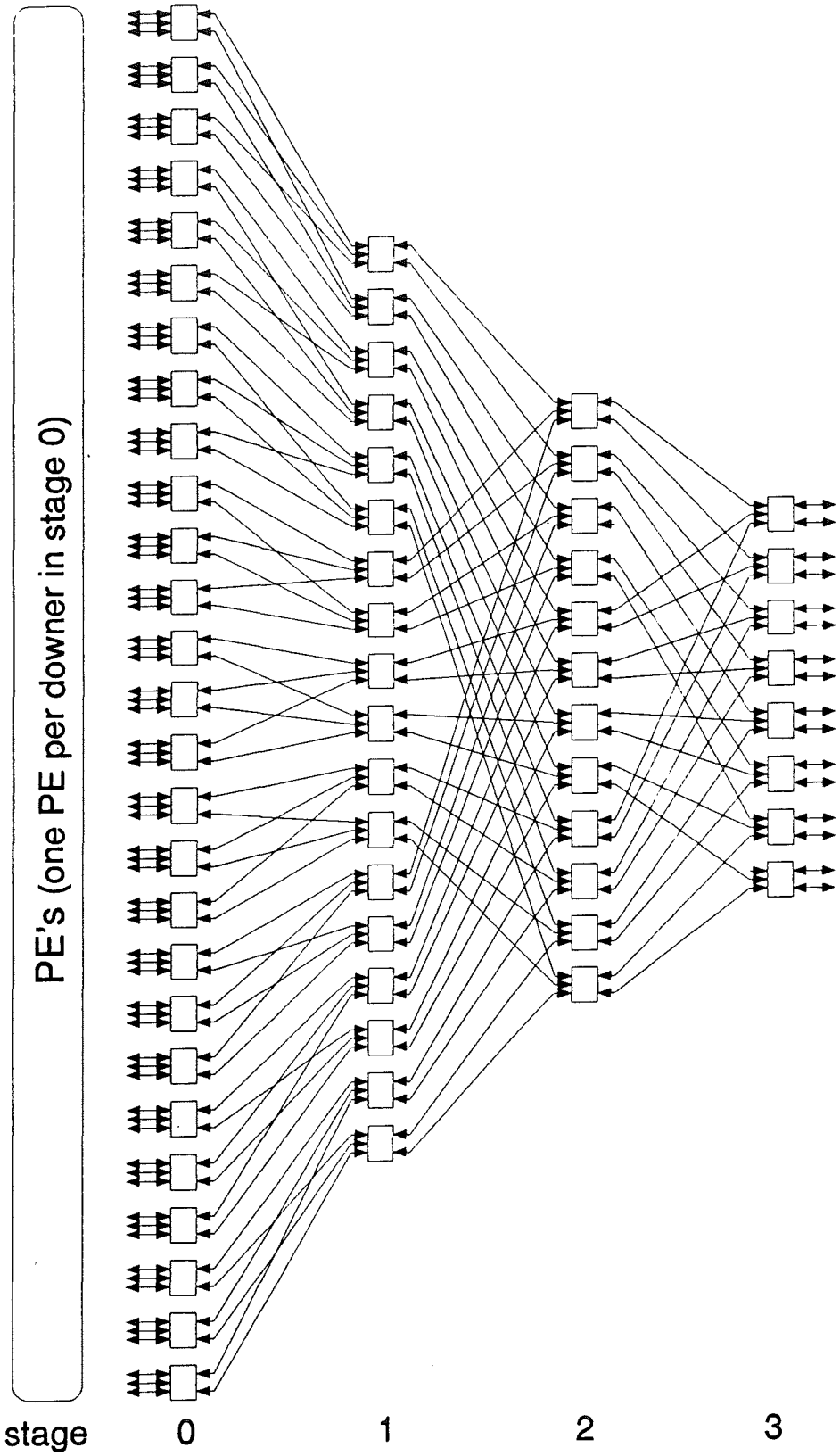

Figure 8: A self-routing LCAN (3,2,54)

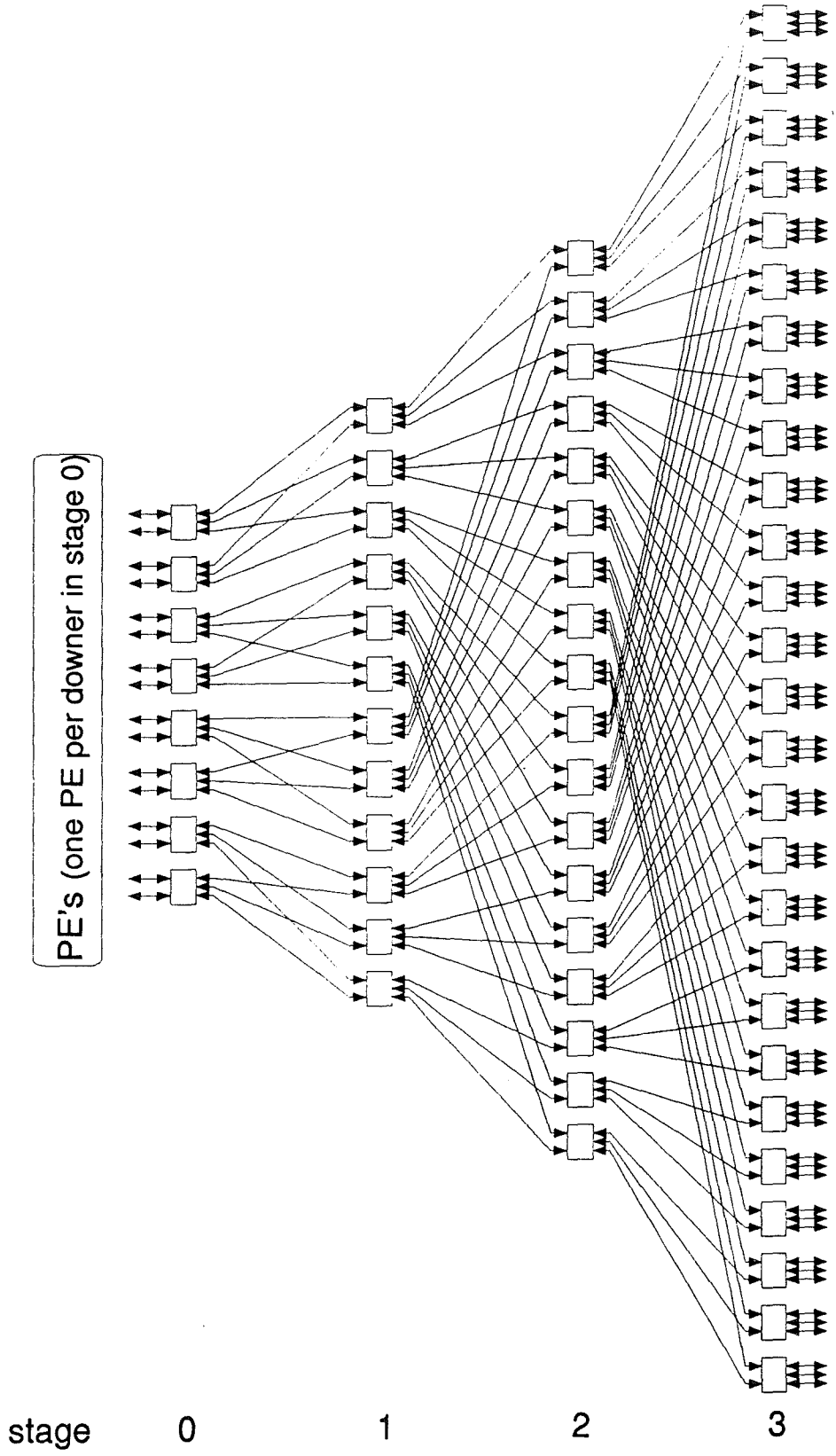

Figure 9: A self-routing LCAN (2,3,16)

base d. In this section, PE-to-PE routing in self-routing LCAN's is explained: the procedure for uni-directional routing is shown, and bi-directional routing is shown using an LCA level computation.

3.1 Uni-Directional Routing

Uni-directional routing uses routing tags in base u to route upwards, and tags in base d to route downwards. Each stage "retires" a unique digit of the destination tag, i.e. a switch in a given stage uses one digit to determine to which link to route. From the bottom of the network, routing upwards to a terminal at the top of the network, let $< t_{l-1}t_{l-2}\ldots t_0 >$ be the destination tag. Then, routing to stage i, t_i is used to select which of the u uppers to connect to. From the top of the network, routing downwards to a terminal at the bottom of the network, let $< r_{l-1}r_{l-2}\ldots r_0 >$ be the destination tag. Then, routing from stage i, r_i determines to which of the d downers to connect.

3.2 Bi-Directional Routing

Bi-directional routing is necessary when PE's are connected to only one side of the network, as in LCAN's. With this placement of PE's, a source-destination pair need not route to a specific LCA switch, only to its LCA level. That is, any switch in the source-destination pair's LCA level that is reachable from the source is guaranteed to have a path to the destination, because of the nature of LCAN construction and the definition of the LCA level. The routing procedure is now simplified: randomly route upwards anyway possible to the LCA level, then deterministically route downwards to the destination PE. This strategy provides some degrees of freedom, i.e. multiple paths, when routing to the LCA level.

Given a source-destination pair requiring bi-directional routing, assume the LCA level is i and the destination address is $< r_{l-1}r_{l-2}\ldots r_0 >$. Then, digits $< r_i r_{i-1}\ldots r_0 >$ are used to route downwards once the LCA level has been reached.

3.3 LCA Level Computation

Let $X =< x_{n-1}x_{n-2}\ldots x_0 >$, $Y =< y_{n-1}y_{n-2}\ldots y_0 >$ and $Z =< z_{n-1}z_{n-2}\ldots z_0 >$ such that all x_i's and y_i's are digits base b, $b > 1$, and all z_i's have value either 0 or 1. Define a function $f(X,Y) = Z$, where $z_i = 0$ if $x_i = y_i$ and $z_i = 1$ if $x_i \neq yi$, $0 \leq i \leq n-1$.

Given a source-destination pair, let the source address $S =< s_{l-1}s_{l-2}\ldots s_0 >$ and the destination address $D =< d_{l-1}d_{l-2}\ldots d_0 >$. Compute $f(S,D) = Z, < Z = z_{l-1}z_{l-2}\ldots z_i\cdots z_0 >$. Then, the LCA level is j such that $z_k = 0$ for $j+1 \leq k \leq l-1$, and $z_j = 1$; i.e. j is the most significant digit where S and D differ.

4 Permutation Routing

Self-routing LCAN's can be classified into networks with switches that have: $d > u$, $d = u$ or $d < u$. The bi-directional routing schemes for self-routing LCAN's simplify permutation routing. All classes of self-routing LCAN's follow the same procedure:

Each source-destination pair computes its LCA level and each source PE sends a header packet upwards through the network until reaching an LCA switch (any switch in its LCA level). Next, it uses the destination address in the header packet to route downwards to its destination PE. The switches are set as the packets route through them. If a PE receives a tag, a path has been successfully established; it sends a packet back to the source PE through the path to confirm the connection.

When $d > u$, there may be contention routing upwards. Up to d packets may compete for u uppers; u packets are chosen randomly. If $d = u$ or $d < u$, there is no contention upwards since there is enough bandwidth to accommodate all packets, and packets are routing to LCA levels, not particular switches. Packets are randomly assigned to uppers.

Routing downwards, all three classes may experience contention since the routing is deterministic. At each unit time step, at most u packets may enter a switch from its uppers. There are two possibilities for contention: 1) if the requested downer is already set, the

requesting packet is dropped; and 2) if the downer is free but more than one packet requests it, one packet is randomly is assigned to it. After all the switches are set, the successful PE's pipeline data to their destinations. The process of setting switches and sending data is repeated until all source-destination pairs of the permutation have been satisfied.

5 Analysis

This section gives a probabilistic analysis of the permutation routing algorithm presented in the previous section. Clearly, self-routing LCAN's do not route all permutations in the same number of passes. For example, the identity permutation is routed in one pass, whereas bit-reversal more than likely requires multiple passes.

Given a network with l stages, consider a "worst-case" set of permutations: permutations where all source-destination pairs have LCA levels equal to l. This set is "worst-case" in a probabilistic sense: the permutations are likely to take more passes to route than permutations with some source-destination pairs having LCA levels less than l, since the latter introduce less contention into the network.

Following the permutation routing algorithm, these worst-case permutations initially randomly route packets to stage l. At this point, a number of packets are at stage l. Assuming an initial load of 1.0 at the bottom of the network, self-routing LCAN's where $d > u$ or $d = u$ now have a load of 1.0 at the top of the network. If $d < u$, then the load at the top is $(d/u)^{log_d n}$. Load is defined as the fraction of terminals on a given side of the network with a header packet to route. Since the upwards routing is random, the packets are distributed pseudo-randomly: routing choices are made randomly, however the choices force a "spreading out" of the packets from a given switch at stage 0. This should in fact reduce contention when routing downwards. However, for the purposes of analysis, it is assumed that the packets are distributed randomly.

In the specific instance where $d = u = 2$, the analysis becomes the same problem as determining the number of passes needed to route a permutation in a butterfly network. This

analysis is given in [13]. The same problem formulation is now generalized for any d and u. Assume the labelling of the stages is reversed, i.e. the top stage is stage 0 and the bottom stage is stage $log_d\, n$. Consider an upper of a switch in stage i. Let p_i be the probability that a packet comes in on it. Then, p_{i+1} is the probability that one or more incoming packets from all uppers want to utilize a particular downer. So, $p_{i+1} = 1 - P$[no packet wants the downer]. P[no packet wants the downer] = P[a given upper does not want the downer]^u, since there are u uppers. P[a given upper does not want the downer] = $1 - P$[the given upper has a packet]$\cdot$ P[the upper wants the particular downer] = $1 - (p_i/d)$. Thus, $p_{i+1} = 1 - (1 - p_i/d)^u$, where $0 \leq p_i \leq 1$, and d and u are integers greater than 0.

This is a non-linear recurrence; its solution is currently unavailable. Attempting to lower bound p_i by expanding the binomial expansion yields another non-linear recurrence of the form $p_{i+1} = a \cdot p_i^2 + b$, where $0 \leq p_i \leq 1$, and a and b are constants in terms of d and u.

This is another non-linear recurrence that has been solved for particular cases of a and b [2], [9]; however, the general solution is currently unavailable. To understand the behavior of the recurrence in a limited fashion, p_i is plotted as a function of i for particular values of u and d.

In Figure 10, $d = u$. As d (and u) becomes larger, the p_i's approach 0 more quickly. Considering this plot in terms of throughput (define throughput as the fraction of PE's that get routed assuming a given load), note that for a given number of PE's, the larger the switch, the smaller the number of stages required. Thus, it is not fair to compare $p_i's$ at $i = 16$ for all switch sizes. For example, in a network comprising 2^{16} PE's and switches with $d = u = 2$, there are 16 stages (each stage retires one bit). Since the number of stages is $log_d\, n$, another network with the same number of PE's and switches with $d = u = 16$ uses only 4 stages (each stage retires four bits). Thus, the throughputs of networks with different switch sizes need to be considered at the proper i coordinate on the plot. In general, as expected according to the recurrence, when $d = u$ it appears that throughput increases with increasing switch size.

Plotting values of p_i for $d > u$ (see Figure 11), the same consideration needs to be given to

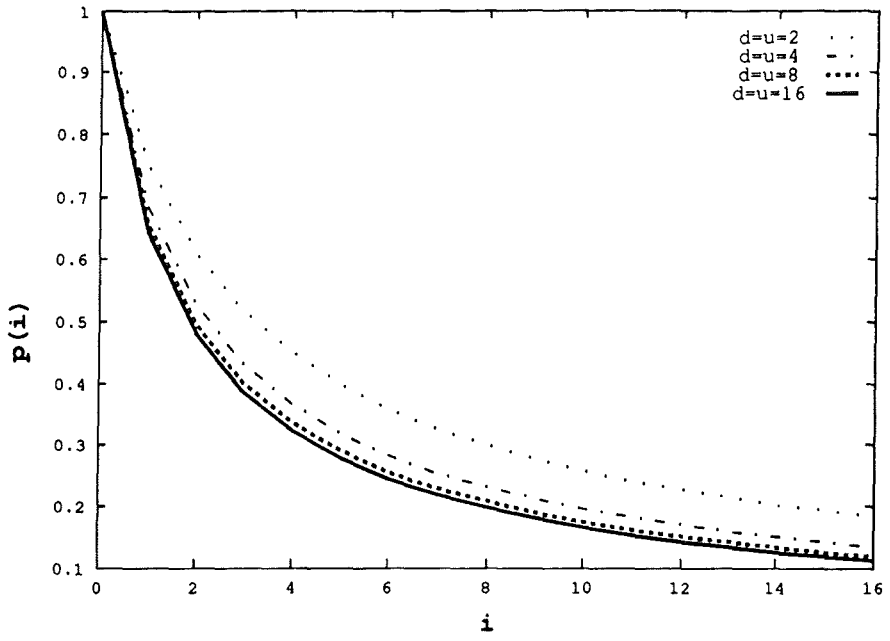

Figure 10: Plot of the recurrence for $d = u$

the i coordinates. However, throughput decreases when u is held constant and d is increased. This is expected because the number of PE's allowed to utilize the top stage links decreases, thereby increasing contention.

In Figure 12, as u gets larger and d is held constant, p_i approaches 1.0 faster (as expected from the recurrence). In terms of the analysis formulation, both of the previous cases, $d = u$ and $d > u$ have loads of 1.0 after routing packets to the top stage. This case, however, leads to a load of $(d/u)^{log_d n}$ in the top stage. Figure 12 shows the behavior of the recurrence, but in order to compare throughputs, given n, the differing loads at the top stage need to be considered (see Figure 13). Here, the p_i increases as i increases since there are of less wires for the packets to be "shared" among. Also, greater values of u, given a fixed value of d, yield better performance since there is less contention in higher stages.

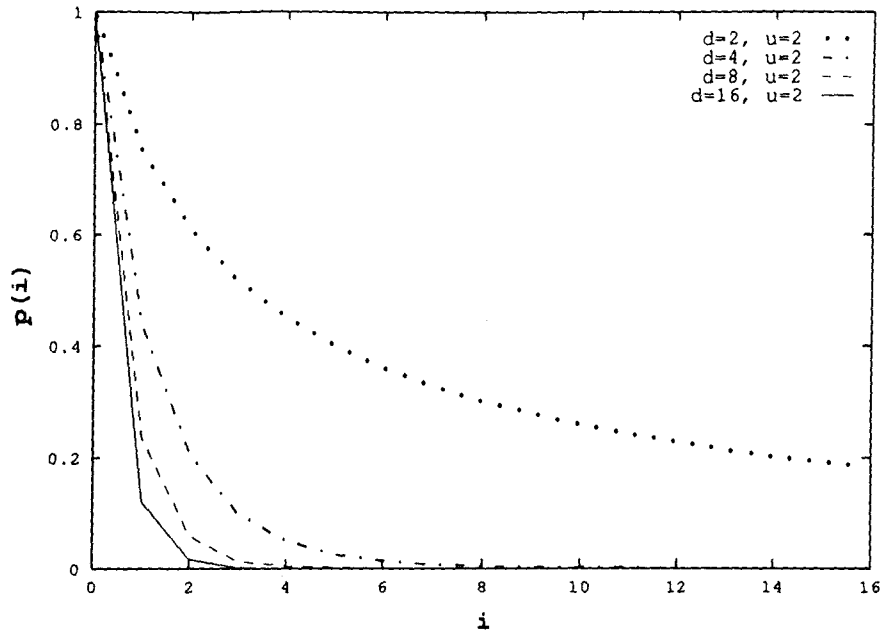

Figure 11: Plot of the recurrence for $d > u$

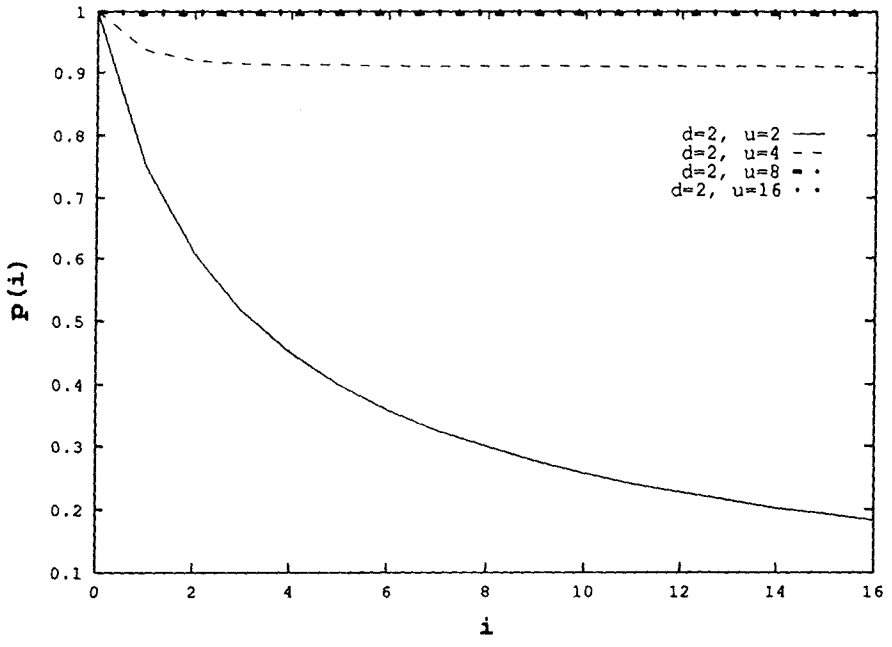

Figure 12: Plot of the recurrence for $d < u$

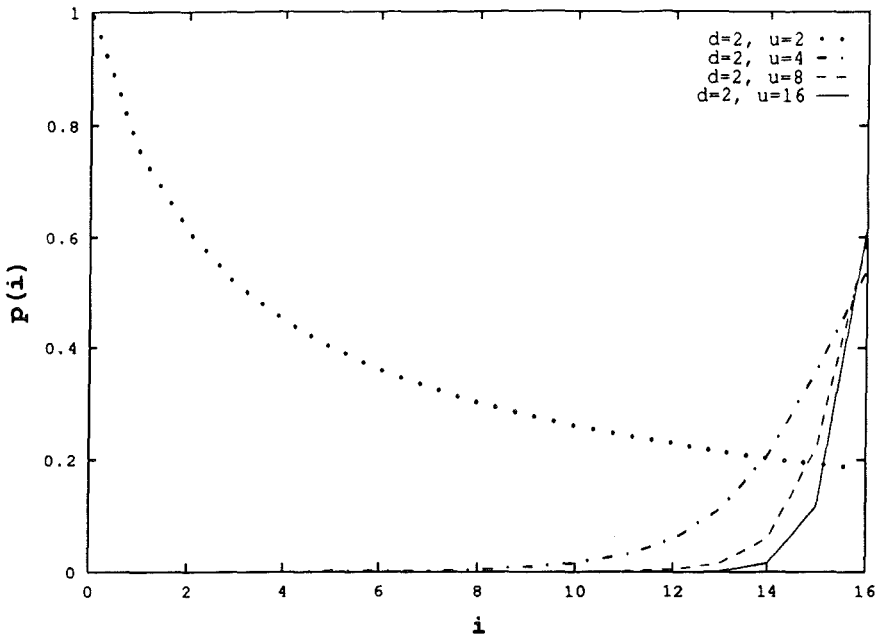

Figure 13: True throughput comparison for $d < u$

6 Conclusion

In this paper, self-routing LCAN's were characterized, and uni-directional and bi-directional routing schemes were given. The partionability of these networks was shown. The routing schemes facilitated a simple permutation routing algorithm. An analysis of a set of worst-case permutations resulted in a non-linear recurrence, its solution is not available at this time. The recurrence's behavior for particular values of d and u was shown and discussed in the context of throughput.

Most self-routing LCAN's where $d > u$ are not practical because of the likelihood of contention at the upper stages of the network, given a random permutation. However, if it is determined that many "commonplace" permutations do not require usage of the upper stages, these networks become very attractive because of hardware considerations (low pinout cost from the backplane) and their permutation routing capabilities. LCAN's with $d < u$ are also likely to be not practical because of the high switch cost. However, LCAN's with $d = u$ are feasible - the CM-5 router network. Future work includes simulating the routing

algorithm and network. Also, it is important to identify permutations that can be routed quickly on specific LCAN's.

References

[1] B.D. Alleyne, C.K. Chien and I.D. Scherson, *Lowest common ancestor interconnection networks,* submitted to ICPP 92, 1992.

[2] C.M. Bender and S.A. Orszag, *Advanced mathematical methods for scientists and engineers,* McGraw-Hill, 1982.

[3] V.E. Benes, *Mathematical theory of connecting networks and telephone traffic,* Academic, New York, 1965.

[4] T. Blank and R. Tuck, *Personal communications,* MasPar Computer Corporation, 1991.

[5] C. Clos, *A study of non-blocking switching networks,* Bell System Technical Journal, Vol. 32, 1953, pp. 406-424.

[6] A.M. Despain and D.A. Patterson, *X-Tree: A tree structured multiprocessor computer architecture,* Proc. Fifth Int. Symp. Comp. Architecture, April 1978, pp. 144-151.

[7] T. Feng, *A survey of interconnection networks,* Computer, Vol. 14, December 1981, pp. 12-27.

[8] J.R. Goodman and C.H. Sequin, *Hypertree: A multiprocessor interconnection topology,* IEEE Transactions on Computers, C-30, December 1981, pp. 923-933.

[9] D.H. Greene and D.E. Knuth *Mathematics for the analysis of algorithms,* 2nd ed., Birkhauser, 1982.

[10] J. Harris and D. Smith, *Simulation experiments of a tree organized multicomputer,* Proc. 6th Annual Symp. on Comp. Arch., IEEE, April 1979, pp. 83-89.

[11] E. Horowitz and A. Zorat, *A divide and conquer computer,* CS Dept. Technical Report, USC, July 1979.

[12] E. Horowitz and A. Zorat, *The binary tree as an interconnection network: Applications to multiprocessor systems and VLSI,* Proc. of the Workshop on Interconnection Networks, April 21-22, 1980, pp. 1-10.

[13] F.T. Leighton, *Introduction to parallel algorithms and architectures: arrays, trees, hypercubes,* Morgan Kaufmann Publishers, 1992.

[14] C. Leiserson, *Fat trees: Universal networks for hardware-efficient supercomputing,* IEEE Trans. on Comp., Vol. C-34, No. 10, October 1985, pp. 892-901.

[15] B.L. Menezes and R. Jenevein, *The KYKLOS multicomputer network: Interconnection strategies, properties, and applications,* IEEE Transactions on Computers, Vol. 40, No. 6, June 1991, pp. 693-705.

[16] J.H. Patel, *Performance of processor-memory interconnections for multiprocessors,* IEEE Trans. Comput., Vol. C-30, No. 10, October 1981, pp. 771-780.

[17] J. Richardson, *Personal communications,* Thinking Machines Corporation, 1992.

[18] I. D. Scherson, *Orthogonal graphs for the construction of a class of interconnection networks,* IEEE Trans. on Parallel and Distr. Sys., Vol. 2, No. 1, January 1991, pp. 3-17.

[19] H.J. Siegel, *Interconnection networks for large-scale parallel processing,* Lexington Books, 1985.

[20] C.W. Wu and T. Feng, *On a class of multistage interconnection networks,* IEEE Trans. Comput., Vol C-29, No. 8, August 1980, pp. 694-702.