Least Common Ancestor Networks

Isaac D. Scherson and Chi-Kai Chien

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

Technical Report #92-105

October 27, 1992

Least Common Ancestor Networks *

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

Abstract

Least Common Ancestor Networks (LCANs) comprise levels of switches that enable communication in two directions. LCANs are a generalization of previously developed networks including fat-trees, baseline networks, SW-banyans and the router networks of the TRAC and the CM-5. In this paper, LCANs are characterized and the routing capabilities of important subclasses are analyzed.

Keywords: SIMD, interconnection networks, 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

There have been many multistage interconnection networks (MINs) proposed for effecting communication between processing elements (PEs) in SIMD machines, as well as MIMD machines [2, 5, 6, 20]. SIMD communication takes the form of *permutation routing* where given a unique labelling of the PEs, source-destination pairs are obtained from a one-to-one mapping of the set of labels onto itself. Past research suggests that determining switch settings to route permutations in MINs capable of routing any permutation in one network cycle takes at least $O(N \log N)$ sequential time [13, 16, 21]. Because of this high set-up cost, it is important to analyze networks that could take multiple network cycles to route a permutation; in each cycle, a subset of the permutation's source-destination pairs are set up and data is transmitted.

Routing a source-destination pair in a MIN typically progresses in only *one* direction through the network (this is called *uni-directional routing*) and must utilize a switch in *every* stage.

Most research concerning MINs has involved studying uni-directional routing; however, procedures for routing in both directions (this is called *bi-directional routing*) in two MINs, SW-banyans and baseline networks, are described in [14] and [21][1]. Bi-directional routing requires a different switch definition. KYKLOS [15] and fat-trees [11] both use tree topologies to facilitate bi-directional routing; in [11] a switch is explicitly defined that permits this form of routing. In this paper, a generalization of bi-directional routing that considers all network topologies is presented, and routing properties, in particular, initial results about permutation routing, are shown.

Least Common Ancestor Networks (LCANs) [1, 4] comprise levels of switches that are capable of connecting bi-directional links in a permutation pattern permitting bi-directional routing. LCANs exploit locality of communication through the communication hierarchy imposed by the actual implementation of massively parallel computers. Traditional implementation of MINs does not take advantage of this 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 PEs [3]. 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.

Typical usage of a MIN for interconnection forces *every* permutation to require backplane communication since each source-destination pair utilizes a switch in every stage. LCANs force off-chip and off-board communication only when necessary. As an extreme example, consider the identity permutation: why should each PE need to find a path through the entire network to its destination? Using an LCAN, each source-destination pair of the identity permutation need only route to the first level of the network and back - only on-chip communication is necessary.

LCANs are a generalization of important pioneering work done by Leiserson [11], Lipovski and Malek [14], Wu and Feng [21], and Menezes and Jenevein [15]. Their networks can be classified as particular instances of LCANs. The work in this paper presents a general framework for researching bi-directional routing and investigates permutation routing properties. Prior work is briefly reviewed with discussion relevant to the LCAN generalization in the following:

[1] Bi-directional routing was termed *full communication* in [21].

Fat-trees reduce root contention by increasing the "capacity" of the links between switches (*i.e.* each link has more bandwidth) closer to the root switch in a binary tree [11]. However, they require switches that proportionally increase in size; for a large number of PEs this can be impractical. Fat-trees are instances of LCANs with switch sizes that vary by level.

SW-banyans are MINs described in [14]. Bi-directional routing procedures for SW-banyans are given without an explicit switch definition; SW-banyans are instances of LCANs. Later in this paper, after considering LCANs in general, SW-banyans are shown to be an interesting subclass of LCANs, and their permutation routing properties are considered.

In [21], bi-directional routing is called *full communication*, a network's ability to connect one terminal to any terminal on either side of the network. A routing strategy for full communication on a baseline network is described; with these bi-directional routing capabilities, the baseline network is an instance of an LCAN. A baseline network is also an instance of a SW-banyan.

KYKLOS [15] comprises replications of k-ary trees; each replication is a particular instance of an LCAN. 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 PEs. However, trees are unappealing because of the high degree of contention near the root of the tree. In KYKLOS, the isomorphic replications provide alternate paths, thus reducing the average interprocessor distance. KYKLOS performance was analyzed in a MIMD context.

Three computers take advantage of locality of communication: the Cm* [8], the TRAC [14] and the CM-5 [12]. In all three computers, communication between closer PEs leads to more available communication bandwidth. The networks and routing algorithms in the TRAC and the CM-5 are similar; the networks are instances of LCANs.

The Cm*, developed at Carnegie Mellon University, provides two levels of locality [8]. PEs within a cluster communicate using a "map" bus, and PEs in different clusters use "intercluster" busses. Thus, if more communication is intra-cluster, *i.e.* local, then there is less contention for the intercluster busses. The TRAC 2.0 is an MSIMD computer developed at the University of Texas at Austin [14]. It uses an SW-banyan network and bi-directional routing. Thinking Machines Corporation's CM-5 is an MIMD computer that uses a router network called a "hyper-tree" [18]. In [12], the network is described as a "4-ary fat tree." The routing of their packets is bi-directional, and the switches have buffers. The exact configuration in terms of LCAN parameters is described later in this paper.

The aims of this paper are to present a generalized framework for prior work and other bi-directional routing networks, and to investigate the usefulness of bi-directional routing for permutation routing. Section 2 presents the LCAN generalization and properties of important subclasses. Section 3 covers routing, and Section 4 concludes.

2 Characterization

An LCAN comprises switches with bi-directional connectors that facilitate communication up and down the network (see Figure 3). The least common ancestor of two nodes in a tree is the node at greatest depth which counts both nodes among its descendants [7, 19]. Analogously, two PEs communicating using an LCAN need only utilize switches as high as one of their least common ancestor switches.

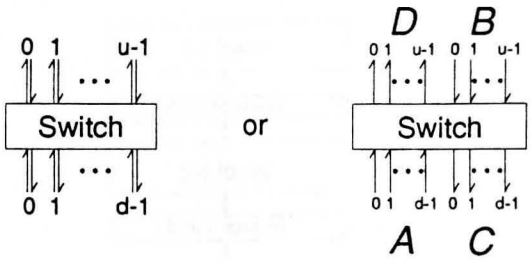

Figure 1: LCAN switch

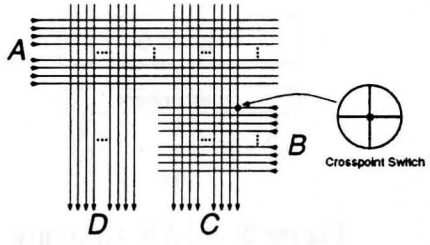

Figure 2: LCAN switch showing crosspoints

In this section, first the LCAN switch is defined, then the network parameters are described. Two significant subclasses of LCANs are identified and some of their properties are stated. Lastly, the import of the level interconnect on routing is discussed.

2.1 Switch description

Each switch has d bi-directional links (labeled $0, 1, \cdots, d-1$) called *downers* that connect to switches in the next lower level and u bi-directional links $(0, 1, \cdots, u-1)$ called *uppers* that connect to switches in the next higher level (see Figure 1). The term *connectors* refers to both uppers and downers.

A bi-directional link actually represents two links, each link carrying information in opposite directions. Let the uppers be partitioned into two groups, B and D, that represent links carrying information to and from the switch, respectively, and the downers also be partitioned into two groups, A and C, that represent links carrying information to and from the switch. In order to achieve connections between links in A, B, C and D, they need to be connected by crosspoint switches detailed in Figure 2. In Figure 2 each intersection of two lines represents a crosspoint.

The switch operates in two modes. In the first mode, data from B and A are used to set the internal crosspoints of the switch. The interpretation of the data by the switch to setup the internal crosspoints is dependent on the routing strategy. In the second mode, information is passed through the switch using the internal crosspoint setup.

2.2 Network parameters

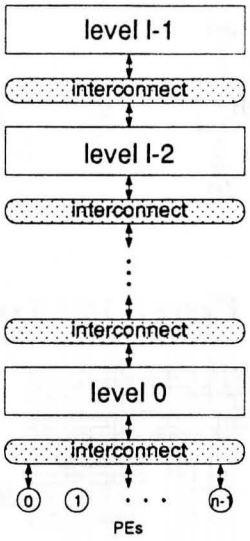

Figure 3: LCAN structure

LCANs are parameterized by (d, u, l, Π)-tuples for N PEs. For purposes of analysis in this section, only LCANs with identical switches are considered (to parameterize LCANs with switch sizes that vary by level, u and d need to be vectors of size N). There are l levels of switches in the network, labelled 0 through $l-1$.

A switch in level i has d bi-directional downers that connect to switches in the next lower level, level $i-1$, and u bi-directional uppers that connect to switches in the next higher level, level $i+1$ (see Figures 1 and 3). Thus, switches in level i are only connected to switches in levels $i-1$ and $i+1$. The exceptions are level $l-1$ (the highest level in the network), in which case the uppers do not connect to anything, and level 0 (the lowest level in the network), in which case the downers connect to the PEs. When $d > u$, the number of switches per level decreases as the number of levels grows; $d = u$, the number of switches per level is constant; $d < u$, the number of switches per level increases.

Let U be the set of all uppers of the switches in some level i, and let D be the set of all downers of the switches in level $i+1$. Then, Π describes the mapping of D to U; the mapping is one-to-one and onto. Note that $|D| = |U|$. Π is called the *level interconnect.*

Let S_i be the number of switches in level i. $S_0 = \frac{N}{d}$ since each switch has d connectors connecting to lower levels. The $\frac{N}{d}$ switches in level 0 each have u uppers connecting to higher levels, thus there are $\frac{uN}{d}$ connectors going up from level 0. Repeating the same reasoning, $S_1 = uN/d^2$.

In general,

$$S_{i+1} = \frac{u}{d} S_i, \text{ where } S_0 = N/d.$$

Recursively substituting,

$$S_i = \frac{N}{d}(\frac{u}{d})^i. \tag{1}$$

Definition 1 *Let* **LCAN(d,u,l,Π)** *denote a least common ancestor network comprising l levels of switches, labelled* 0 *through* $l-1$*, with interconnectivity defined by* Π*; a switch in level i has d downward connectors to switches in level* $i-1$ *and u upward connectors to switches in level* $i+1$*. It is assumed there are N processors connected to the switches in level* 0*, one per downward connector and there exists at least one possible path in the network between any two processors. It is also assumed there are also no unused connectors between consecutive levels.*

The following theorem holds for all LCANs:

Theorem 1 *Given an LCAN(d,u,l,Π), l must be less than or equal to a, where a is the smallest integer such that* $\frac{N}{d}(\frac{u}{d})^{a-1}$ *is an integer and* $\frac{N}{d}(\frac{u}{d})^{a}$ *is not an integer.*

Proof:

Assume $l > a$. By Definition 1, there exist levels $a-1$ and a. Using Equation 1, level $a-1$ has $S_{a-1} = \frac{N}{d}(\frac{u}{d})^{a-1}$ switches, and uS_{a-1} upward connectors to level a. To fully use every one of these connectors, there need be exactly $\frac{u}{d}S_{a-1}$ switches in level a. However, $\frac{u}{d}S_{a-1} = \frac{N}{d}(\frac{u}{d})^{a}$, which is not an integer by assumption. It follows that there must be unused downward connectors in level a, contradicting Definition 1. $\Box Q.E.D.$

Now two subclasses of LCANs are defined in terms of their Π:

Definition 2 *Given an LCAN(d,u,l,Π), let* Π = **(d/u)-ary tree** *iff the switches are arranged in a (d/u)-ary tree fashion where each edge of the tree represents u connectors. Clearly, this is only possible when d is a multiple of u.*

Definition 3 *Given an LCAN(d,u,l,Π), let* Π = *complete bipartite if the interconnectivity is as follows:*

Label the PEs consecutively using $\log_d N$ *digits base d:* $< p_{\log_d N-1} p_{\log_d N-2} \cdots p_0 >$*, and label the switches in level 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:* $< w_{\log_d N-2} w_{\log_d N-3} \cdots w_i w_{i-1} \cdots w_0 >$.

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.*

To connect switches in level i labelled $< w_{\log_d n-2} w_{\log_d n-3} \cdots w_{i+1} j w_{i-1} \cdots w_0 >$*, where* $j = 0, 1, ..., d-1$*, to switches in level* $i+1$*, remove the rightmost base d digit (j) and append a base u digit to the right of the switch label, call it k. The d switches in level i connect to u switches in level* $i+1$ *- those labelled* $< w_{\log_d n-2} w_{\log_d n-3} \cdots w_{i+1} w_{i-1} \cdots w_0 k >$*, where* $k = 0, 1, ..., u-1$*, via the* $k+1$*st upper of the switch in level i and the* $j+1$*st downer of the switch in level* $i+1$*.*

LCANs with Π = *complete bipartite* are topologically the same as SW-banyans. Definition 3 is more rigorous than the definition in [14], and leads to the following lemma:

Lemma 1 *A processor can connect to* u^i *unique switches in level i.*

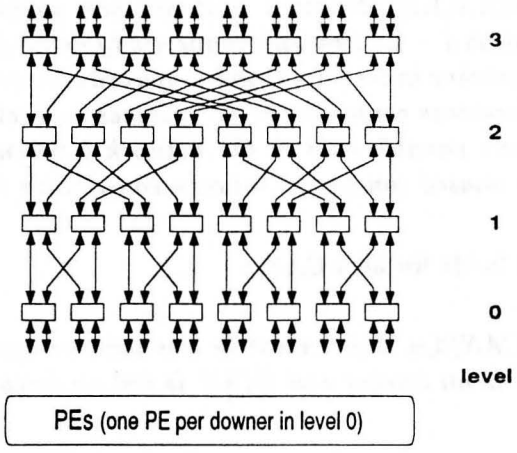

Figure 4: A CB-LCAN(2,2)

Proof: Observe that the u uppers of a switch in level 0 connect to u different switches in level 1 because single base u digit in the switch label represents u distinct values. Likewise, the u level 1 switches connect to u^2 different switches in level 2 because the two base u digits represent u^2 distinct values. It follows that a processor can connect to u^i distinct switches in level i. $\square Q.E.D.$

With the definition of these two Πs, l can be defined as a function of N, d and u for both subclasses of LCANs:

Corollary 1 *(to Theorem 1) Given an LCAN(d,u,l,complete bipartite), $N = d^l$, or $l = log_d N$.*

Proof: By Lemma 1, the number of switches in level $l-1$ that a processor can reach is u^{l-1}, the total number of switches in that level. Equation 1 states that $S_{l-1} = \frac{N}{d}(\frac{u}{d})^{l-1}$. Equating the two,

$$u^{l-1} = \frac{N}{d}(\frac{u}{d})^{l-1}.$$

Reducing, we obtain $N = d^l$, or $l = log_d N$. $\square Q.E.D.$

Definition 4 *Let an LCAN(d,u,l,complete bipartite) be denoted* **CB-LCAN(d,u).**

Figure 4 is a CB-LCAN(2,2).

Corollary 2 *(to Theorem 1) Given an LCAN(d,u,l,(d/u)-ary tree), $N = \frac{d^l}{u^{l-1}}$, or $l = log_{\frac{d}{u}}(\frac{N}{u})$.*

Proof: By Definition 2, $S_{l-1} = 1$; by Equation 1, $S_{l-1} = \frac{N}{d}(\frac{u}{d})^{l-1}$. Equating the two,

$$\frac{N}{d}(\frac{u}{d})^{l-1} = 1.$$

Therefore, $N = \frac{d^l}{u^{l-1}}$, and it follows that $l = log_{\frac{d}{u}}\frac{N}{u}$. $\square Q.E.D.$

Definition 5 *Let an LCAN(d,u,l,(d/u)-ary tree) be denoted* **T-LCAN(d,u).**

Note that a T-LCAN where $u = 1$ is a limiting case of a CB-LCAN. *I.e.* T-LCAN(d,1) is equivalent to CB-LCAN(d,1).

Given these properties and definitions, networks from prior research can classified by LCAN parameters:

- A baseline network with full communication capability is equivalent to CB-LCAN(2,2).
- A (f,s,$log_d N$) SW-banyan is equivalent to a CB-LCAN(f,s).
- The TRAC network is a CB-LCAN(3,2).
- A fat-tree has the form of a binary tree, but the switch size varies by level; they are similar to T-LCANs. An exact description requires a vector of d's and u's, a vector element per level.
- The CM-5 router network is composed of a CB-LCAN(4,4) and many CB-LCAN(4,2)s.

2.3 Level interconnectivity

In this section, initially some definitions are given, then the importance of Π with respect to routing is discussed.

Definition 6 *A* **least common ancestor (LCA) switch** *of two processors is a switch in level i, such that:*

1) it can be connected to both processors using only switches in levels 0 *through* $i - 1$, *and*

2) there is no switch in a level j, $j < i$, that satisfies property 1).

Note that two PEs may have more than one LCA switch.

Definition 7 *The* **LCA level** *of two processors is the level to which their LCA switch(es) belong.*

Lemma 2 *In a CB-LCAN, two processors whose LCA level is i have exactly u^i LCA switches.*

Proof: Assuming the LCA level of two PEs, a and b, is i, it follows from Lemma 1 that a is connectable to u^i switches in level i, and b is connectable to u^i switches in level i. Due to the recursive nature of the interconnect, the two sets of switches are either disjoint or identical. If the former, then there are no switches in level i that can connect a and b, and thus the LCA level could not be i. Therefore, it follows from the latter that the number of LCA switches is u^i. $\square Q.E.D.$

Lemma 3 *In a T-LCAN, any two processors have exactly one LCA switch.*

Proof: In a k-ary tree, given any two leaves (PEs) there is exactly one least common ancestor (LCA switch), the root of the smallest subtree containing both leaves. $\square Q.E.D.$

Definition 8 *A* **path** *between two processors consists of an alternating series: connector, switch, connector, switch,$\cdots$, connector, such that two processors may communicate in a circuit-switched fashion.*

A *non-redundant switch* path is a path that does not pass through any switch more than once.

Theorem 2 *The number of non-redundant switch paths between any two processors in a CB-LCAN is exactly the number of LCA switches of the two processors.*

Proof: Each LCA switch is the root of a subtree of switches, the leaves are PEs. The subtrees share the same leaves due to the recursive nature of the CB-LCAN interconnect. In a tree, there is only one path between any two nodes. Utilizing switches in levels above an LCA switch does not add additional non-redundant switch paths because the LCA switch lies within a tree containing those switches and the path taken upwards must be the same path as that taken back down (thus the LCA switch is passed through twice). Therefore, the number of non-redundant switch paths between any pair of leaves is exactly the number of trees, or in this case, the number of LCA switches. $\Box Q.E.D.$

Because of these LCA switches, all LCANs exhibit the following feature: given a source-destination pair, there are one or more LCA switches, and communication need progress upwards only to an LCA switch. 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* level of the network; whereas with the typical usage of MINs all source-destination pairs *must* route through every stage.

This LCAN feature is important because it complements the physical partitioning of the network into chips and boards. Permutations requiring only on-chip and on-board communication are performed quickly. Determination of these permutations depends on a given partitioning and quantitatively relates the physical construction of the network to the expected permutation traffic.

In order to exploit this routing feature, network parameters that permit routing to and from an LCA switch need to be identified. Ideally, this routing would be based solely on local knowledge (source and destination labels).

Certain MINs, *delta networks* [17] are capable of "self-routing," *i.e.* require only a destination address to route, uni-directionally across the network. A subclass of delta networks, *bidelta networks* [9] are also self-routing if the network is reversed. CB-LCANs are topologically similar to delta and bidelta networks. However, because of the added switch functionality, CB-LCANs additionally allow bi-directional routing using only source and destination labels; T-LCANs also allow bi-directional routing. Currently, no routing algorithms that only use local knowledge have been identified for other LCANs.

One focus of any comparison between CB-LCANs and T-LCANs should be their routing capabilities. One metric that affects routing is *spreadout*. The *spreadout* of Π in an LCAN(d,u,l,Π) is the number of different switches in level $i+1$ to which the uppers of a switch in level i connect. Π = *complete bipartite* affords maximal spreadout, *i.e.* u, and $\Pi = (d/u) - ary\ tree$ yields minimal spreadout, *i.e.* one. These two Πs are regular in the sense that every switch has the same spreadout.

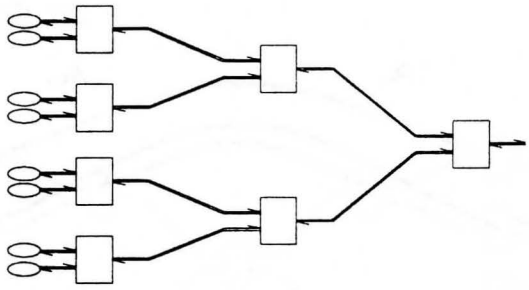

Figure 5: LCAN labelled A

Given a CB-LCAN and a T-LCAN with the same N, d and u, observe that l in the CB-LCAN is always less than or equal than l in the T-LCAN. This follows from Corollaries 1 and 2. In the limiting case, $u = 1$ and the networks are identical; assume $u > 1$. Intuitively, the maximal spreadout of the CB-LCAN allows a processor to communicate with more processors "faster," *i.e.* with less levels. From Equation 1, since N, d and u are the same for both networks, the total switch cost for the CB-LCAN is less than that of the T-LCAN.

3 Routing

Attention is focused on CB-LCANs in this section because routing on T-LCANs is essentially a limiting case of routing on CB-LCANs. LCAN switches are assumed to have no buffers, thus the results in this section apply to circuit-switched packet routing.

In this section, uni-directional and bi-directional routing methods are restated from prior work and an interesting perspective relating bi-directional routing to uni-directional routing is presented. Then, routing on CB-LCANs given global knowledge is discussed. Finally, permutation routing is explored.

3.1 PE-to-PE routing

CB-LCANs facilitate uni-directional routing upwards using routing tags in base u and downwards with routing tags in base d. The switches in a given level use a unique digit of the destination tag to determine which link to use.

Lipovski and Malek describe a procedure for bi-directional routing [14]; for the specific case of $d = u = 2$, it is the same as the procedure in [21]. The procedure is summarized here: Bi-directional routing uses an LCA level computation. A source-destination pair needs to route to its LCA level. 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 recursive nature of the Π = *complete bipartite* interconnect. The routing procedure provides some degree of freedom: randomly route upwards anyway possible to the LCA level, then deterministically route downwards to the destination PE using part of the destination label[2]. The LCA level is the place of the most significant digit where the source and destination labels differ.

[2]The CM-5 router network uses this routing algorithm.

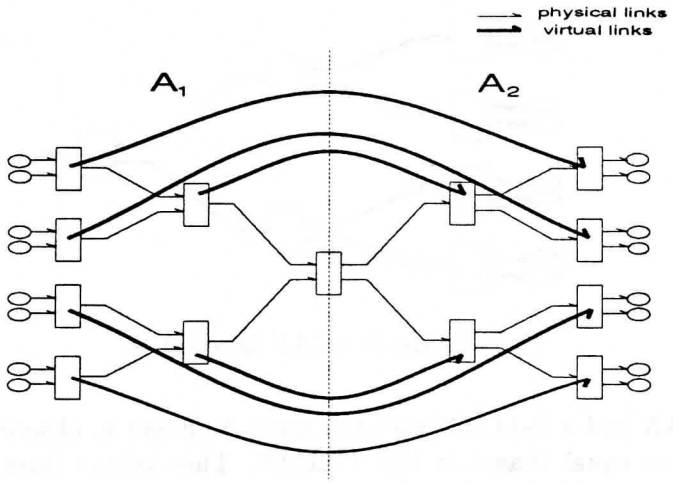

Figure 6: A unfolded into A_1 and A_2 showing virtual links

Bi-directional routing on LCANs can also be viewed from another perspective. Let A be an LCAN (see Figure 5); let A_1 comprise the PEs, the switches and the upward-going wires of the bi-directional links. Let A_2 comprise the downward links of the bi-directional connectors, and duplicates of the PEs and the switches (except those in level 0). Unfold A_2 in a mirror-like fashion (see Figure 6). Each switch in A_1 effectively has an extra link to its corresponding mirrored switch in A_2 (represented by the thick edges in Figure 6). These are virtual links, and are costless to traverse. Communication in the unfolded LCAN occurs uni-directionally; however, in the folded LCAN A, communication moves forward then backward.

The added functionality provided by the virtual links allows certain connections to be made without utilizing a switch in every level, *i.e.* some middle levels need not be traversed. Given that the switch that is the least common ancestor of two communicating PEs belongs to some level k, to connect the PEs, $1 + 2(k-1)$ levels in A can be skipped by utilizing the virtual links between A_1 and A_2.

3.2 Global routing knowledge

To route a set of packets, possibly a permutation, with the minimal amount of network cycles, global knowledge is usually required. Either all PEs know all source-destination pairs and act in unison, or a central controller knows all source-destination pairs and directs the PEs. Leiserson developed excellent off-line routing results for fat-trees in [11]. In this section, routing with global knowledge is considered on CB-LCANs by examining their "equivalent" fat-trees. First, Leiserson's results are restated (see [11] for further details).

Recall that a fat-tree takes the form of a binary tree and the switch size can vary by level. The set of connectors between two switches is called a *channel.* Given a message set M, the *load factor*, $\lambda(M, c)$, of a channel c due to M is

$$\lambda(M, c) = \frac{load(M, c)}{cap(c)}.$$

Theorem 3 *A lower bound on the number of network cycles required to route M is $d \geq \lambda(M)$,*

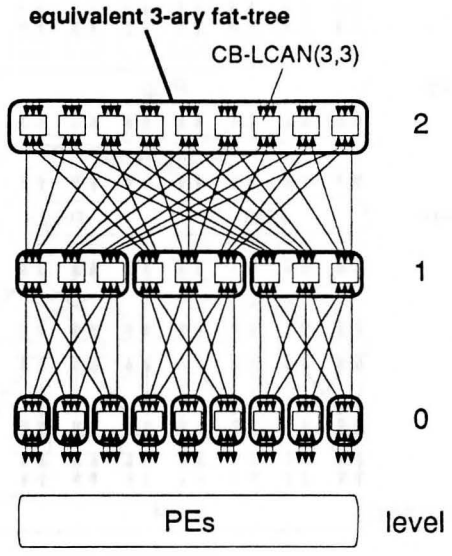

Figure 7: A CB-LCAN(3,3) and its equivalent 3-ary fat-tree

where $\lambda(M)$ is the maximum $\lambda(M,c)$ over all channels.

Theorem 4 *On a fat tree, for any message set M with $\lambda(M) > 1$, there is an off-line schedule of d network cycles such that $d = O(\lambda(M)\ log\ N)$.*

Leiserson provides a method for computing this schedule. To compare CB-LCANs with fat-trees, the definition of a fat-tree is extended:

Definition 9 *Let* **k-ary fat-trees** *be LCANs such that if there are S_i switches in stage i, then $S_{i+1} = kS_i$.*

Thus, 2-ary fat-trees are equivalent to Leiserson's fat-trees. It is conjectured that a schedule of $O(\lambda(M)\ log\ N)$ cycles can be scheduled off-line for a message set M on a k-ary fat-tree.

Definition 10 *Let the* **equivalent d-ary fat-tree** *of a CB-LCAN(d,u) be recursively constructed using a* combine *operation that replaces a group of smaller switches with one large switch; connector number and connectivity is maintained - combining m LCAN switches with d downers and u uppers yields an LCAN switch with (md) downers and (mu) uppers. The construction is as follows: combine the u^{l-1} switches in level $l-1$, then divide the switches in level $l-2$ into d consecutive evenly sized groups. Apply the combine operation recursively to each group (see Figure 7).*

Given two processors in a CB-LCAN there can be many LCA switches; in the equivalent d-ary fat-tree there is only one bigger LCA switch (as a result of the combine operation).

It is conjectured that given global knowledge about a message set M, the minimum number of network cycles necessary to route M in a CB-LCAN(d,u) is the exactly the same as the minimum number in its equivalent d-ary fat-tree. If the two conjectures in this section prove correct, then Leiserson's off-line results can be directly applied to CB-LCANs.

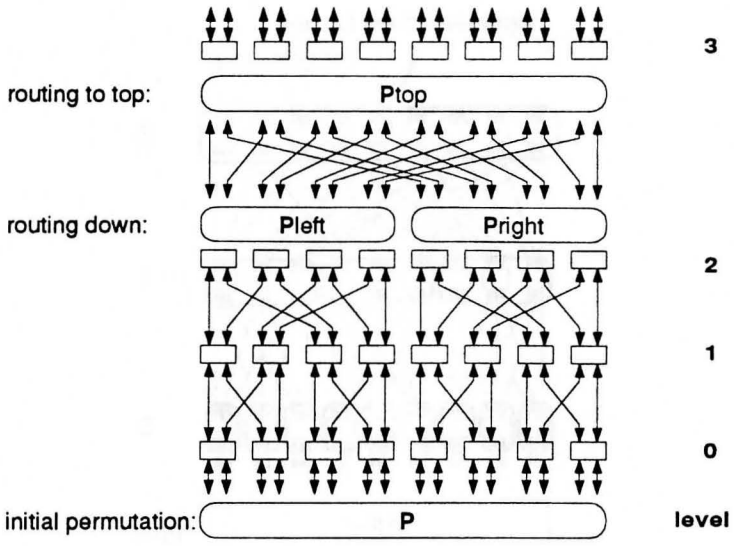

Figure 8: Routing a "worst-case" permutation in a CB-LCAN(2,2)

3.3 Permutation routing

Routing the identity permutation takes only one network cycle in any LCAN since all source-destination pairs need only route to level 0. To compare this permutation to other permutations based on network cycles, the *average* number of cycles necessary to route a permutation must be used since the routing algorithm is random (see Section 3.1).

In this section, permutations with source-destination pairs that all have LCA level $l-1$ are considered on CB-LCANs where $d = u$ (the performance of permutation routing on CB-LCANs where $d \neq u$ is an open question). This set is hereafter referred to as the "worst-case" set. It is conjectured that the average number of cycles to route the "worst-case" set is greater than the average number of cycles for any other set.

The routing of the "worst-case" set is illustrated with CB-LCAN(2,2) (see Figure 8). Choose any permutation from the "worst-case" set, call it P. Let A be the set of PEs labelled 0 to (N/2-1), and let B be the set of PEs (N/2) to (N-1). The destinations of A comprise a permutation of the labels of B, and vice versa; otherwise not all source-destination pairs require routing to the top level. After routing P to the top level, call the resultant permutation P_{top}, each switch contains two packets - one packet must have come from A and be destined for B and vice versa for the other packet.

Next, after routing back down to level $l-2$, the uppers of the switches in the left half of level $l-2$ have a permutation of the labels of A, call it P_{left}, and likewise for the right half and B, call that permutation P_{right}. Observe that P_{left} and P_{right} are random permutations, since given any P, each possible setting of the switches in the network leads to a unique P_{top}. After the first cycle, P_{left} and P_{right} will be partial permutations of A and B respectively. Now the question is how many cycles does it take to route a random permutation uni-directionally through the network?

Routing in a butterfly is the same as routing uni-directionally in a CB-LCAN(2,2). Routing the

transpose permutation in a butterfly takes $\sqrt{N}$ cycles (shown in Section 3.4.9 of [9]). In a butterfly, the transpose permutation is one of relatively few worst-case permutations. So if P_{left} and P_{right} are transpose permutations of the labels of A and B respectively, routing P could take $O(log\ N)$ cycles in the worst case. However, given that P is in the "worst-case" set, the likelihood of P_{left} and P_{right} being transpose permutations is $(\frac{1}{N!})^2$. The likelihood of this occurring $\sqrt{N}$ times is even smaller. Thus it is *extremely* unlikely that routing P will take $\sqrt{N}$ cycles, even if P is in the "worst-case" set. The above reasoning extends to all networks where $d = u$. However, keep in mind that the average case, not the worst case, is of prime interest since P_{left} and P_{right} are random permutations.

In a uni-directional network, the random routing problem is highly similar to permutation routing. Random routing is where every processor randomly chooses a destination from a uniform distribution; a processor could have multiple packets destined for it. It is conjectured that any bound on the number of passes for random routing applys to permutation routing as well.

Problem 3.285 in [9] involves proving that random routing on a butterfly ($d = u = 2$) takes $O(log\ N)$ cycles with high probability; the proof exists [10]. It is easy to see that this extends to uni-directional routing on CB-LCANs with $d = u = 2^k$, where k is an integer: replace each switch with an equivalent CB-LCAN comprising $d = u = 2$ switches; the larger switch is certainly at least as powerful than the equivalent CB-LCAN. It is unclear whether the proof of problem 3.285 extends to all switch sizes where $d = u$.

If the uni-directional random routing bound is an upper bound on uni-directional permutation routing, then routing the "worst-case" set of permutations takes $O(log\ N)$ cycles with high probability. Note that this bound applies to on-line routing, no central controller or global knowledge is necessary.

4 Conclusion

In this paper, LCANs were characterized - they were shown to be a generalization of many previously developed networks. Important classes of LCANs and their relationships were defined. The LCAN switch facilitated the bi-directional routing characteristics of these networks. It was shown that due to the nature of the routing and typical network implementation, locality of communication can be exploited to decrease routing time.

Initial performance results for permutation routing were shown for CB-LCANs where $d = u$. If bounds on the number of network cycles necessary for the similar problem of random routing apply to permutation routing, then a schedule of $O(log\ N)$ cycles with high probability can be achieved on-line for a "worst-case" set of permutations.

References

[1] B.D. Alleyne, C.K. Chien and I.D. Scherson, *Lowest common ancestor interconnection networks,* Technical Report #92-19, University of California, Irvine, February 14, 1992.

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

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

[4] C.K. Chien and I.D. Scherson, *Self-routing lowest common ancestor networks,* to appear as a poster paper, Frontiers of Massively Parallel Processing, 1992.

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

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

[7] D. Harel and R.E. Tarjan, *Fast algorithms for finding nearest common ancestors,* SIAM J. Comput., Vol. 13, No. 2, May 1984, pp. 338-355.

[8] K. Hwang and F. Briggs, *Computer architecture and parallel processing,* McGraw-Hill, 1984.

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

[10] F.T. Leighton, *Personal communication,* 1992.

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

[12] C. Leiserson, et al, *The network architecture of the connection machine CM-5,* SPAA '92, June 1992, pp. 272-285.

[13] G.F. Lev, N. Pippenger and L.G. Valiant, *A fast parallel algorithm for routing in permutation networks,* IEEE Trans. Comput., Vol. C-30, No. 2, February 1981, pp. 93-100.

[14] G.J. Lipovski and M. Malek *Parallel computing,* Wiley & Sons, 1987.

[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] D. Nassimi and S. Sahni, *Parallel algorithms to set up the Benes permutation network,* IEEE Trans. Comput., Vol. C-31, No. 2, February 1982, pp. 148-154.

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

[18] J. Richardson, *Personal communication,* Thinking Machines Corporation, 1992.

[19] B. Schieber and U. Vishkin, *On finding lowest common ancestors: simplification and parallelization,* SIAM J. Comput., Vol. 17, No. 6, December 1988, pp. 1253-1262.

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

[21] 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.