The Kyklos (n Mod r = 0) Routing Algorithm

F. F. Haddix, A. G. Dale,
R. M. Jenevein, and C. B. Walton

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-88-27 July 1988

Abstract

Tree network routing algorithms and node labeling sequences can simultaneously achieve O(1/N) processor loads among interior node processors, level processor loads among leaf node processors and $O(1/N^{1-(1/r)})$ maximum traffic per link, where N is the number of I/O processors located at the leaf nodes, and r is the number of tree replications. Proposed routing algorithms are an r-tree analog to the 2-tree H-2 strategy [Jene86], and the Y-2 strategy, which improves network performance on significant traffic metrics relative to H-2. If m is the tree branching factor, and $n = \log_m N$ is the number of levels in each tree, the routing algorithms work for any m, n and r; however, they are most effective for m = 2 and n mod r = 0.

I. Introduction.

The KYKLOS topology [Mene85] is a multicomputer network based on a multiple tree interconnection topology. In a KYKLOS-x<m,r,n> network, there are N (= m^n) leaf node processors (L-nodes) and B {= [r / (m - 1)](m^n - 1)} interior branch node processors (IB-nodes). The L-nodes are I/O processors, while the IB-nodes have no secondary storage. x refers to the interconnection schema. In KYKLOS-I, a two-tree network replicates the IB-nodes of the first (upper) tree in the second (lower) tree. In KYKLOS-II, the second tree is "shuffled", improving connectivity and reducing certain traffic characteristics.

The number of nonequivalent interconnection strategies in the KYKLOS topology increases with m and n, where the actual number for KYKLOS-x<m,2,n> is given by $N!/(m!^{[(m^{**}n-1)/(m-1)]}$ [Mene86]. With such a large number of choices, the choice of interconnection strategy is important. The objective of this study is to identify an interconnection topology which supports a general approach to routing, minimization of maximum link traffic, and equal loading of processors.

The motivation of this study is the problem of computing a relational join on a partitioned data base. Using the KYKLOS topology with the database distributed across N L-nodes, there will be N^2 partial joins. Two relations, R and S, are joined on a set of common attributes which become the join attributes. In the general case, the N R fragments must be merged with the N S fragments in order for the join to be executed. The investigation is limited to operations on two partitioned relations, R (the smaller in pages) and S (the larger in pages).

The notation used to describe a KYKLOS network refers to trees, links and nodes. The r trees are designated 0, 1, ..., r-1; the N L-nodes are named 0 through N-1. Relations R and S are partitioned horizontally and allocated equally to the L-nodes. Each IB-node has a three-part name, <tvo> or <t,v,o>, where t is the tree name (0 through r-1), v is the level name ($0 < v <= n$, $n = \log_m N$), and o is the node specifier ($0 <= o < 2^{n-v}$).

A consistent routing algorithm for all nodes is necessary for an approach to be generalizable. One way of assuring that this will happen is to have symmetric rules for assigning processing to nodes. In turn, this should yield symmetric (and simple) routing algorithms.

The following discussion assumes that partial joins will take place at IB-nodes. The site of the join is a function of the source nodes of the R and S fragments. Ideally, the node at which a particular partial join is performed is on the path between the R-fragment originating node and the S-fragment originating node. Otherwise, unnecessary traffic would be generated in transferring join candidates to the join site IB-node.

If the joining node is selected so that it is closer to the S-fragment originator, additional savings in traffic result from moving the larger S fragments less and the smaller R fragments farther. If this approach is used in concert with level processor loading, the S fragment is moved only in the tree of its partial join, and only toward the root of that tree, thereby producing the minimum traffic for S fragments, since the S fragment must be moved at least to the level of its partial join. These two properties imply the additional property of nearly equal link traffic balance with respect to S fragments, since each succeeding level v will have $1 / m$ (m^{n-v} / m^{n-v+1}) links (1 /2, if $m = 2$) and will have approximately $1 / m \{(m^{n-v} - 1) / (m^{n-v+1} - 1)\}$ fragments to transfer.

Note that the above relationship holds with respect to the results of the partial joins if the results are stored at the originating node of the S fragment (or anywhere else in the subtree of the partial join site node).

Using the above philosophy, the principal remaining problems to be solved are the following:

a. Determining in which tree a particular partial join will occur (objective: equal processing and traffic in all trees);

b. Determining at which level a particular partial join will occur (objective: equal processor loading between nodes of different levels); and

c. Determining the routing path for the R fragments (objective: minimizing maximum link traffic.)

Note that if the tree and level of a particular partial join have been determined, at that level in that tree there will be only one ancestor IB-node of the S-fragment originating L-node, and this will be the join site. This defines the routing of the S-fragment, as being toward the root of that tree only as far as that level.

II. Dividing Processing Load Between L-nodes and IB-Nodes.

In the KYKLOS architecture configuration, data is stored at the L-nodes while the IB-nodes have no secondary storage capacity. Due to this and the I/O and other processing load on L-node processors, it is desirable to have different loads on L-node and IB-node processors, but equal, or level loads among all L-node processors and among all IB-node processors.

One reliable rule for dividing the work load that will always work is the following:

For a particular partial join $<R_i, S_j>$,

If i = j, join at L-node i;

else join at an IB-node.

This means that each L-node will be the site for exactly one partial join, regardless of n, thus leaving $(N^2 - N)$ partial joins to be performed at the B $\{= [r / (m - 1)](m^n - 1)\}$ IB-nodes. Thus,

Average joins per IB-node $= (N^2 - N) / \{[r / (m - 1)](m^n - 1)\}$

$= (mN - N) / r$

Since the number of IB-nodes at level v is $r*m^v$ and the number of leaves is N, the number of S fragment assignments per level per L-node is the following:

S fragment assignments per level per L-node $= [(mN - N) / r] * (r * m^v / N)$

$= [(m - 1) * m^v]$

III. A Conceptual View of the Routing Problem

Going beyond SISD solutions to multiple computing problems requires that individual processors be distinguishable, and have a method of unique access. The most convenient solution would be to assign each computer a unique name, which not only gives its location, but how to reach it.

There exists a relationship between the binary digits of a number and a binary tree, well-documented in the literature [Lee59], [Bene65], [Sing81], [Horo81], [Mene88]. If the L-node processors are enumerated from 0 to N - 1, left to right, the binary representation of the processor name (number), will give the Huffman code routing from the root to the processor.

The same routing methodology can be used for trees with branching factor (m) greater than 2. For these, the address should be converted to a base m representation. The routing is then based on an increasing left-to-right correspondence, that is, if m = 3, 0 would refer to the left branch, 1 to the middle and 2 to the right.

Routing from the root to the L-node is helpful; however, of greater value is routing between L-nodes. A simple approach, would be to start from the first L-node, traverse to the root, and then use the foregoing routing to traverse to the destination L-node. This routing specifies going to the root on every traversal between two leaves, which is clearly unnecessary in many cases.

One way of conceptualizing this problem is to say that each L-node processor occupies a unique position in an n-dimensional space, where n is the length of the address, and each digit of the address gives the coordinate for the corresponding dimension. Each level of a tree corresponds to a dimension in this space, the correspondence being that at that level and only that level within the subject tree can the spatial coordinate for that dimension be changed.[1] A traversal from L-node i to L-node j is a journey to the appropriate levels in order to execute the appropriate changes of coordinate.

An efficient routing within a tree will only go as far toward the root as the highest level for which a dimensional change of coordinate is necessary. A message going from L-node i to L-node j starts with physical coordinates corresponding to the address of i. Its target coordinates correspond to the address of j. When the physical coordinates of the message equal its target coordinates, it has arrived at its destination. If i <> j, it will have traversed one or more IB-nodes and if r > 1, it may have traversed IB-nodes of more than one tree and L-nodes intermediate (in some sense) to i and j. To the extent that coordinates (for corresponding dimensions) within the addresses of i (initial physical) and j (target = final physical) are the same, they represent a level at which no coordinate change is necessary (hence, traversal may not be necessary). However, if the coordinates are different, then the level corresponding to this dimension (digit in m-ary representation of the address) must be traversed.

---

[1] If there are r trees in the network there are r levels at which the change could take place, one per tree. One of these levels would occur in each of the following ranges: between levels 1 and h; h+1, 2h; ...; h(r-1)+1, rh.

Some of the levels for which no change of coordinate is necessary must be traversed in order to reach a level for which a change of coordinate is necessary, except in the case of the HyperKYKLOS [Mene87a], where the number of trees is equal to the number of levels (= dimensions = $\log_m N$, where N is the number of L-node processors),

The above described changes in dimensional coordinates will be referred to hereinafter as transforms. The corresponding level traversals are independently necessary if a transform is needed at that level and dependently necessary if a transform is not needed at that level, but the traversal is part of the routing to a level of independent necessity. Hereinafter, an independently necessary level traversal will be referred to as an 1-transform, a dependently necessary level traversal will be referred to as a 01-transform, and a possible transform not used in a specific routing will be referred to as a 00-transform. Thus, 00- and 01- transforms form the set of 0-transforms, those transforms which may be of use in the instant routing for traversing but not for transforming.

Within a particular network topology, the number of 1-transforms for a particular routing is constant, as is the number of 0-transforms; however, the numbers of 00- and 01-transforms are functions of routing strategy.

Once a routing strategy has been determined in a KYKLOS topology, an instance of routing will require that a specific level in a specific tree must be attained. The message routed must traverse the tree toward the root until it reaches that level. This level will be a 1-transform, since it must be necessary and a 0-transform would not be necessary. Once this level is attained, the message goes down the tree. For each IB-node, there are m branches, each having a different orientation. For example, if m = 3, each IB-node would have one link each of left, middle and right orientation. If the link taken going away from the root is the same as the link taken when going toward the root, it is a 0-transform; if the two links are different, it is a 1-transform.

The XOR (exclusive OR) operator classifies potential link traversals as being 0-transforms (value 0) or 1-transforms (value 1). However, it does not discriminate between necessary and unnecessary level traversals where no transform is required (01-transforms versus 00-transforms). Recall the operation $X = XOR_m (i, j)$. If we subscript the digits of base m representation of i and j from right to left as 0, 1, ..., n - 1, then the following holds:

$i = i_{n-1} * m^{n-1} + i_{n-2} * m^{n-2} + ... + i_2 * m^2 + i_1 * m + i_0$, which can be represented as

$i = i_{n-1}i_{n-2}...i_2i_1i_0$

$j = j_{n-1} * m^{n-1} + j_{n-2} * m^{n-2} + ... + j_2 * m^2 + j_1 * m + j_0$, which can be represented as

$j = j_{n-1}j_{n-2}...j_2j_1j_0$

If we similarly subscript the bits (base 2 representation) of X:

$X = X_{n-1} * 2^{n-1} + X_{n-2} * 2^{n-2} + ... + X_2 * 2^2 + X_1 * 2 + X_0$, which can be represented as $X = X_{n-1}X_{n-2}...X_2X_1X_0$,

where $X = XOR_m (i, j) \Rightarrow X_g = 0$ if $i_g = j_g$

$X_g = 1$ if $i_g <> j_g$, $0 <= g < n$,

IV. Describing the Interconnection Network

If processing is allocated equally among trees using a symmetric basis for allocation, and if symmetric routes are selected, then both total traffic load and maximum link traffic will be balanced among the trees. The interconnection network (ICN) must be specified before loads can be allocated among regions (trees, levels). The KYKLOS-II topology has many benefits [Mene88] and is used herein.

Once the tree topology has been selected, the next choice is of a logical labeling sequence. The (n mod r = 0) shuffle is amenable to the development of comprehensible routing algorithms. For a KYKLOS-II<m,n,r> network, n is the number of levels. n is also the number of base m digits in the address of a L-node processor. Thus, if (n mod r = 0), we can allocate the n base m digits equally among the r trees.

In distinguishing among regular shuffles, where regular means that one simple rule is applied at each level (but not necessarily all levels within a tree), a useful discriminator between levels is span. Span for a given level is defined as the difference between the lowest[2] L-node number within each successive subtree of a child of an IB-node of that level. Figure 1 illustrates this concept. This is expressed as $span_{tv} = m^{**}(z_{tv} - 1)$, where a unique z_{tv} is associated with each level v within a tree t, ($0 < v$, $z_{tv} <= n$), and each tree t, ($0 <= t <= r$), has a complete set of z_{tv}, defined as above. Note that this z_{tv} corresponds to dimension number in the discussion of transforms in Section III. In other words z_{tv} indicates which dimension's coordinates can be changed when level v of tree t is traversed. When a series of spans or dimensions is given to describe the topology of a tree, the order is from level 1 to the level of the root.

For a KYKLOS-I<m,r,n> tree, where v = 1, 2, ..., n and z_{tv} = 1, 2, ..., n. Thus as we go up a KYKLOS-I<2,r,n> tree, our spans ($span_{tv}$) would be 1, 2, 4, 8, ..., indicating, for example, that the 0 node and the 1 node would be children of the same level 1 parent. In turn, the 0 node and 1 node would have the same level 2 ancestor as nodes 2 and 3 (difference 2). Likewise, nodes 0, 1, 2, 3 would have the same level 3 ancestor as nodes 4, 5, 6, 7 (difference 4).

Similarly, for the lower tree of a KYKLOS-II<m,r,n> interconnection network, z_{tv} = n, n-1, n-2, ..., 1, 0. Thus, in a KYKLOS-II<2,r,4> lower tree, going to higher levels of this tree, our spans would be 8, 4, 2, 1. This tells us that nodes 0 and 8 would have the same level 1 parent; nodes 0 and 8 would have the same level 2 ancestor as nodes 4 and 12; and nodes 0, 4, 8, 12 would have the same level 3 ancestor as nodes 2, 6, 10, 14. The fourth level is the root. An example is shown in Figure 2.

---

[2] Note that using highest, median or mean L-node name will give the same result as using the lowest L-node name.

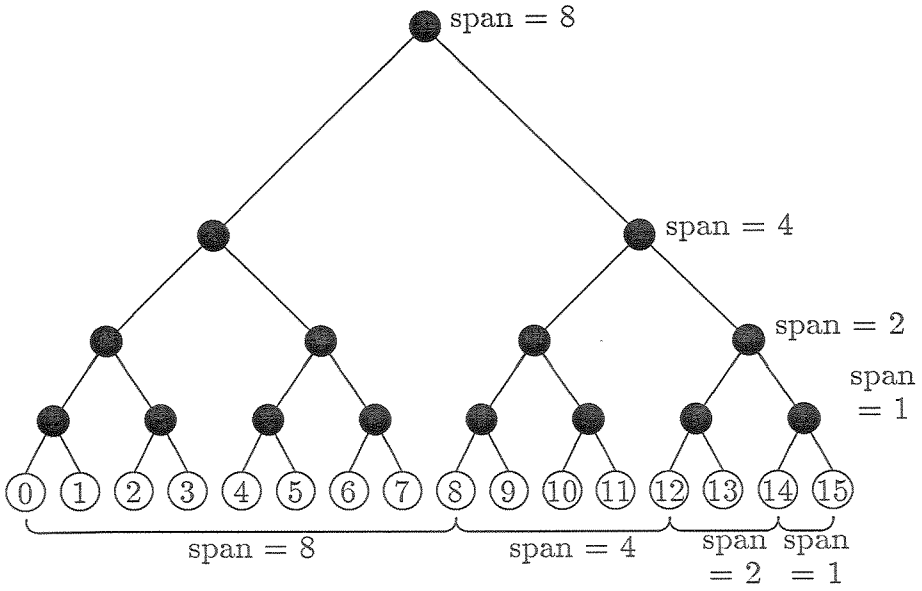

Figure 1: Original KYKLOS-II topology illustrating span: N = 16, r = 1.

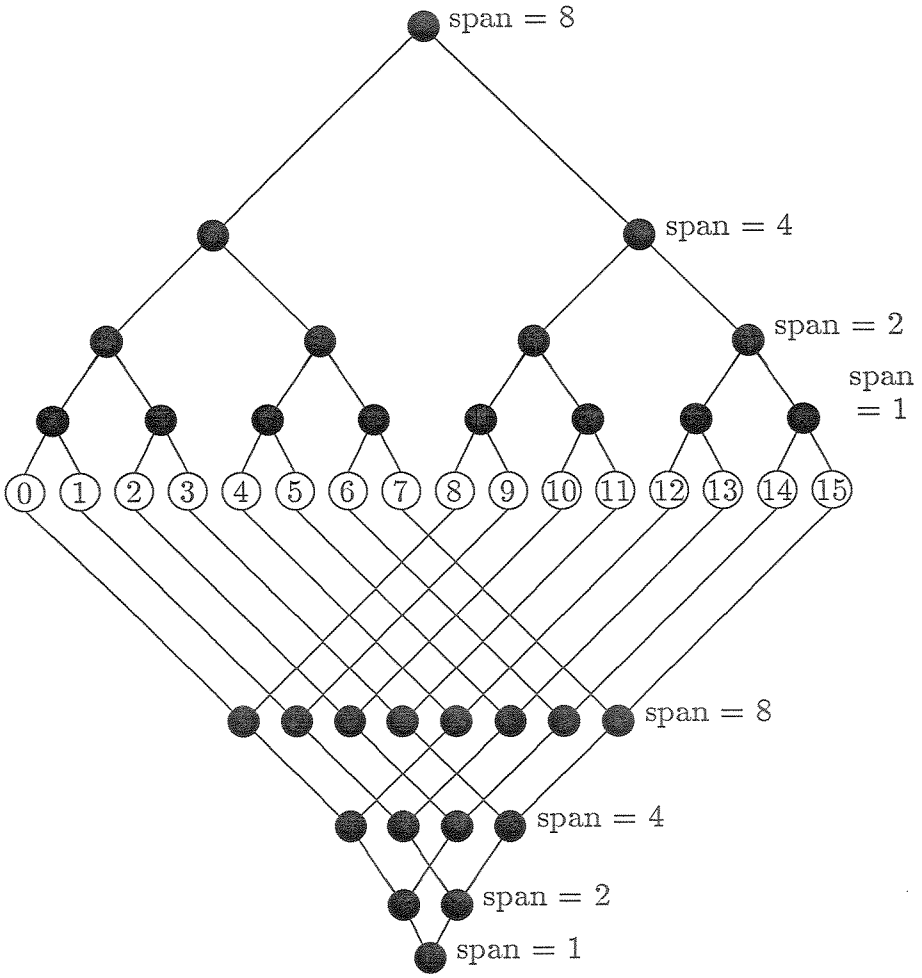

Figure 2: Original KYKLOS-II topology, where N = 16, r = 2.

The objective stated in Section I. was to identify an interconnection topology which supports a general approach to routing, minimization of maximum link traffic, and equal loading of processors. The topology proposed to meet these objectives assigns spans by the following formulation:

t = tree ($0 <= t < r$);

v = level ($0 <= v < n$);

m = branching factor ($0 < m$);

$h = n / r$; [3]

$z_{tv} = h * t + v - 1$

if($v > h$)

$z_{tv} = z_{tv} + h + 1 - 2 * ((v - 1) \bmod h + 1)$

$z_{tv} = z_{tv} \bmod n$

where

mod refers to the integer modulo function,

div refers to integer division (floor function understood), and

/ also refers to integer division, but is used only in places where the floor property is not needed.

$span_{tv} = m^{z_{tv}}$

Examples: n=6, m=2, r=3:

$h = 6 / 3 = 2$

t=0, v=2:

$z_{02} = 2 * 0 + 2 - 1 = 1$
$z_{02} = 1 \bmod 6 = 1$
$\Rightarrow span_{03} = m^1 = 2$

[3] Note that ($n \bmod r = 0$) implies that h is an integer.

t=0, v=3:

$z_{03} = 2 * 0 + 3 - 1 = 2$
$z_{03} = 2 + 2 + 1 - 2 * ((3 - 1) \bmod 2 + 1) = 3$
$z_{03} = 3 \bmod 6 = 3$
$\Rightarrow span_{03} = m^3 = 8$

t=1, v=3:

$z_{13} = 2 * 1 + 3 - 1 = 4$
$z_{13} = 4 + 2 + 1 - 2 * ((3 - 1) \bmod 2 + 1) = 5$
$z_{13} = 5 \bmod 6 = 5$
$\Rightarrow span_{13} = m^5 = 32$

t=2, v=6:

$z_{26} = 2 * 2 + 6 - 1 = 9$
$z_{26} = 9 + 2 + 1 - 2 * ((6 - 1) \bmod 2 + 1) = 8$
$z_{26} = 8 \bmod 6 = 2$
$\Rightarrow span_{26} = m^2 = 4$

The entire set of spans for two KYKLOS-II<m,r,n> networks with the proposed topology are given in Table 1.

Proposed KYKLOS-II<2,3,6>

	z_{tv}			$span_{tv}$		
	tree	tree	tree	tree	tree	tree
level	0	1	2	0	1	2
1	0	2	4	1	4	16
2	1	3	5	2	8	32
3	3	5	1	8	32	2
4	2	4	0	4	16	1
5	5	1	3	32	2	8
6	4	0	2	16	1	4

Proposed KYKLOS-II<2,2,6>

	z_{tv}		$span_{tv}$	
	tree	tree	tree	tree
level	0	1	0	1
1	0	3	1	8
2	1	4	2	16
3	2	5	4	32
4	5	2	32	4
5	4	1	16	2
6	3	0	8	1

Table 1. Spans ($span_{tv}$) and dimensions (z_{tv}) for each level (v) and tree (t) in proposed KYKLOS-II<2,3,6> and proposed KYKLOS-II<2,2,6> configurations.

Labeling sequence refers to the order of nodes in a tree if the tree is rearranged so that no links cross. The characteristic labeling sequence of KYKLOS-II is the Gamma(m,N) sequence [Mene85]. It is not immediately clear that the proposed topology is KYKLOS-II, particularly because both upper and lower trees in the r = 2 version have crossing links. (See Figure 3 for N = 16 example.)

The KYKLOS-II Labeling Sequence for the upper tree:

U: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

Gamma Sequence describing lower tree [Gamma(2,16)]:

L: {0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15}

If the upper and lower trees of the original KYKLOS-II shown in Figure 3 are redrawn so that no links cross, the following labeling sequences result:

U': {0, 1, 2, 3, 8, 9, 10, 11, 4, 5, 6, 7, 12, 13, 14, 15}

L': {0, 4, 8, 12, 2, 6, 10, 14, 1, 5, 9, 13, 3, 7, 11, 15}

Without loss of generality (since the U' names are artifacts of routing strategy but not topology), the nodes in U' and L' can be renamed such that U' = U. Referring to the renamed sequences as U* and L*, a comparison of U' and U* defines the conversions to be made to L' in order to derive L*. See Figure 4 for example of mapping of the conversions.

Since L* is equal to Gamma(2,16), we assert that the proposed topology is an instance of KYKLOS-II. Figure 5 shows an intermediate step in the conversion from the proposed topology to the original. The labeling sequence for IB-nodes is given in Appendix B.

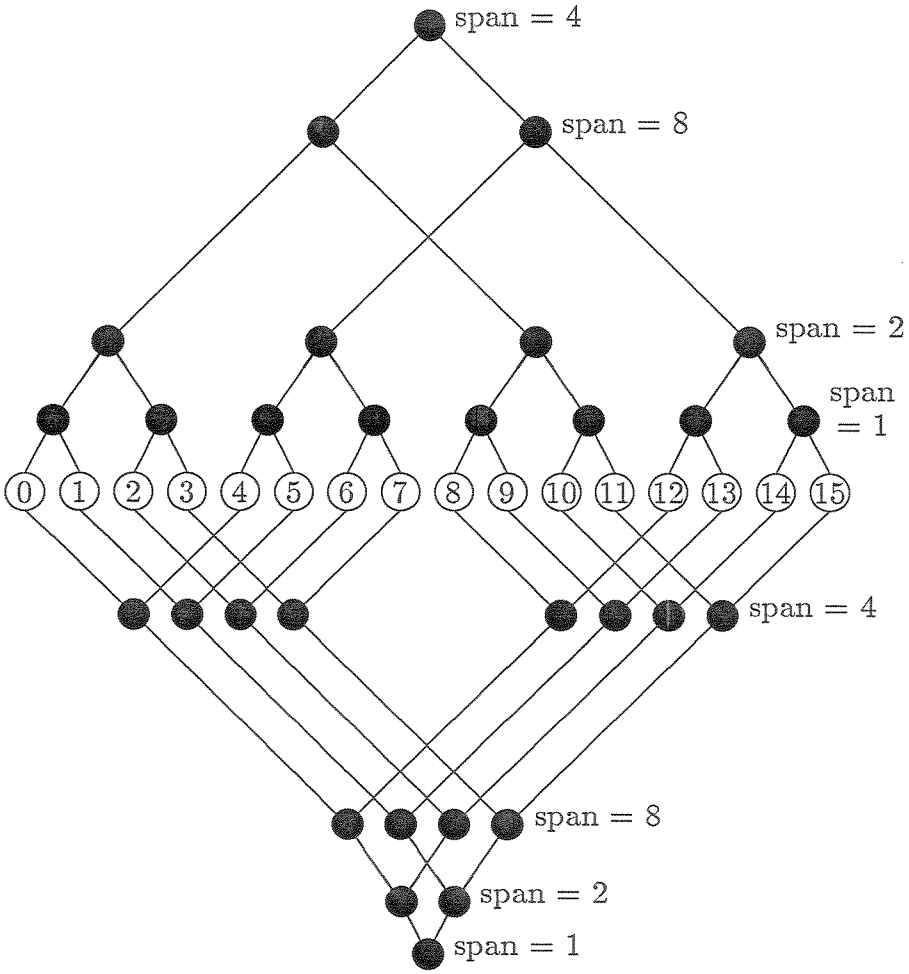

Figure 3: Proposed KYKLOS-II topology, where N = 16, r = 2.

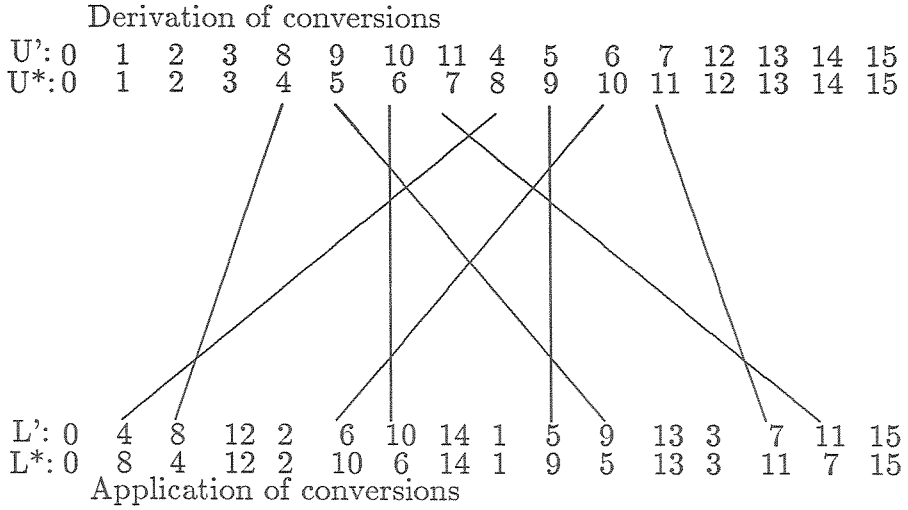

Figure 4: Illustration of conversions between ' and * sequences

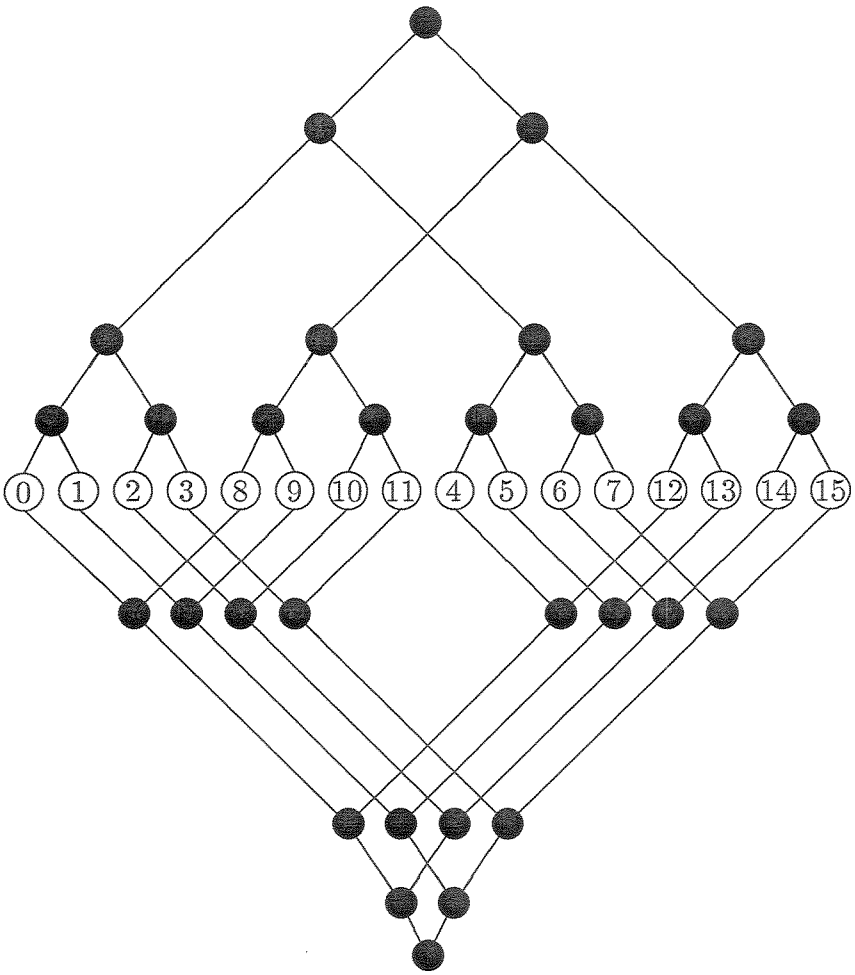

Figure 5: Proposed KYKLOS-II network of Figure 3, relabeled in accordance with Figure 4. If it is twisted to remove the cross-over from the upper tree, this network becomes equivalent to the original KYKLOS-II network shown in Figure 2.

V. Routing in Multiple Tree Interconnection Networks

One solution to routing with more than one tree is the binary solution of KYKLOS-II [Mene87b]. This uses X = XOR(S, D), where S is the source node name and D is the destination node name. In order to make the routing handle more than one tree, the node name is mapped into the tree topology for both trees. This is achieved by letting the increasing order of significance direction within the node name represent the upper tree, and the opposite order, for the lower tree. If routing was between node 15 and node 37 in a KYKLOS-II<2,2,6> interconnection network, the following would result:

15: 001111

37: 100101

X: 101010

With the transform concept of routing, there are three 1-transforms to be made in the two trees. All the transforms must be made, but they can be made in either tree. One possibility would be to traverse only the upper tree. Since this traversal must go all the way to the root, the total traversal is length 12.

Another possibility is to traverse only the lower tree. Routing here is exactly the same as in the upper tree, except that we start from the other end. This is because the spans in the lower tree are 32, 16, 8, 4, 2, 1, the reverse of the order in the upper tree; however, the total traversal here is of length 10, because the root node need not be traversed.

This paradigm is also useful for describing routes that use both trees. In the above example, there are six possible transforms, three that are independently necessary (1-transforms) and three that are dependently necessary or unnecessary (0-transforms). Finding the minimum path requires minimizing the number of dependently necessary transforms (01-transforms). The lower tree routing above was shorter because it considered the transform corresponding to the span = 1 level to be unnecessary (00-transform), rather than a 01-transform, as in the upper tree. Any routing which considers one 0-transform to be a 00-transform will be of length 10 (and in this case, minimal). Two-tree traversals which are minimal, avoid the unnecessary transforms for the span = 4 and span = 16 levels. There are four such routings:

Avoiding the span = 4 transform: Traversing three levels of the lower tree, then two levels of the upper tree; and traversing two levels of the upper tree, then three levels of the lower tree.

Avoiding the span = 16 transform: Traversing one level of the lower tree, then four levels of the upper tree; and traversing four levels of the upper tree, then one level of the lower tree.

In this example, there are five minimal paths. One can be selected on the basis of other criteria, such as minimizing maximum traffic density. As in the one tree case, the X string only indicates how far up to go in a particular tree, i.e., which levels are to be traversed. The specific coordinate change to be made at a level is derived from the base-m representation of the destination node name.

Finding the minimum path is described [Mene88] as a process of finding the largest sequence of consecutive 0's (representing potentially unnecessary transforms) in the X string, and eliminating them from the routing. For the $r > 2$ condition, an analogous process exists.

In the one tree ($r = 1$) case, the minimum path is found by eliminating the upper (toward the root) unnecessary traversals.

In the two tree case, the largest group of consecutive unnecessary traversals can be removed. This is possible because of the redundancy between the two trees. If the ordering of the spans is inverted between the two trees any 1-transform can be performed in either tree. Thus, 1-transforms for levels above the selected group of consecutive 0-transforms can be moved to the second tree[4].

In the r tree case, a useful analytic concept is the slice previously defined for the two-tree case [Jene86]. For r > 2, the slice becomes a subgraph of the interconnection network such that the levels represented constitute a n-member set containing one and only one level corresponding to each of the n implicit dimensions of the network, and for each tree t, if level v is included in the subgraph, then levels 1 to v - 1 are also included. An equi-slice, corresponding to the mid-slice previously defined [Jene86], can be defined as a slice which contains an equal number of levels from each tree. For more than two trees, no simple path optimizing rule analogous to the rule removing the largest groups of consecutive 0-transforms from a routing in the two-tree case. If the metric of interest is path minimization, the largest group of 0-transforms can be removed from any pair of consecutive trees; however, this may foreclose the use of the second tree in a pairing with its successor, and the use of the first tree with its predecessor. A different metric, such as minimizing maximum link traffic may add to the problem's complexity.

In the proposed routing, the mapping from the m-ary representation of a node name applies the general principles enunciated above. If the number of levels (n) is divided by the number of trees (r), h is obtained, where h is the height of an equi-slice and is the length of the routing string corresponding to the lower h levels of one tree.[5] Note that h is the height of a mid-slice in the two-tree case. Before the above can be directly applied to the proposed topology, the base-m representation of the node name and X = XOR(S, D) must be recast as follows.

---

[4] In this view, one tree traversals are degenerate cases in which all or none of the 1-transforms are moved to the second tree.

[5] If only one transform is required, and it falls in a tree's lower h-levels, then the shortest path will traverse only that tree.

r segments of X (X(r-1), X(r-2), ..., X2, X1 X0) can be defined such that the H-2 routing string Xt, associated with tree t, is of length h (= n/r) and is derived from X as follows:

$Xt_g = X_{g+th}$

so that

$X0 = X_{h-1}...X_g...X_1X_0$ and $X0_g = X_g$

$X1 = X_{2h-1}...X_{h+g}...X_{h+1}X_h$ and $X1_g = X_{h+g}$

...

$Xt = X_{(t+1)h-1}...X_{th+g}...X_{th+1}X_{th}$ and $Xt_g = X_{th+g}$

...

$X(r-1) = X_{rh-1}...X_{(r-1)h+g}...X_{(r-1)h+1}X_{(r-1)h}$ and $X(r-1)_g = X_{(r-1)h+g}$

The Y-2 routing string for tree t is then constructed as follows[6]:

Yt = X((t-1)mod r)' cnc ... cnc X((t+2)mod r)' cnc X((t+1)mod r)' cnc Xt

where cnc indicates the concatenation operator--adding the second string onto the end of the first, and ' indicates that the operation of inverting the bit positions has been performed (first becomes last, next-to-first becomes next-to-last, and so forth.

In the routing between two L-nodes, i and j, X = XOR(i,j), and Yt = f(X), as described above. An analogous conversion is performed on the D (destination node name) string in order to determine which branches to use in the routing. If starting from i, D = j and vice versa. Note that the Xt strings determine the depth of traversal for H-2 routing, the Yt strings will be used to determine the depth of traversal for the Y-2 routing and the Dt[7] strings give the choice of branch during descent of a tree (as discussed in Section III.).

---

[6] Note that inverted ordering is a characteristic of both the original and proposed KYKLOS-II topologies.

[7] The Dt = f(i) or Dt = f(j) string would be defined somewhat differently for H-2 and Y-2 routing.

Example 1:

Traversal from node 24 to node 62, (N = 64, n = 6, r = 2, m = 2)

S: 24: 011000

D: 62: 111110

X: 100110 => X0 = 110; X1 = 100; X0' = 011; X1' = 001.

Y0: 001110 (001 cnc 110)

Y1: 011100 (011 cnc 100)

D0: 111110

D1: 011111

The above indicates a path length of 4 * 2 = 8 in tree 0, and a path length of 5 * 2 = 10 in tree 1. This routing is not amenable to pass-through. If tree 0 were traversed, the routing would go up four levels from node 24, then descend to node 62 taking successive branches right, right, right, left. If tree 1 were traversed, the routing would ascend five levels from node 24, then descend to node 62 taking the right link at each of the five IB-nodes.

Example 2:

Traversal from node 60 to node 14, (N = 64, n = 6, r = 2, m = 2)

S: 20: 010100

D: 14: 001110

X: 011010 => X0 = 010; X1 = 011; X0' = 010; X1' = 110.

Y0: 110010 (011 cnc 010)

Y1: 010011 (010 cnc 110)

D0: 100110

D1: 011001

The above indicates a path length of 6 * 2 = 12 in tree 0, and a path length of 5 * 2 = 10 in tree 1. Inspecting X0 and X1 indicates the longest group of 0's is 2, indicating a shortest path length of (6 - 2) * 2 = 8. This corresponds to a two-level deep traversals of both trees. One of the routings would be to go up two levels in tree 0, coming down with a right and then a left; then going up two levels in tree 1, coming down with a left and then a right. The second routing reverses the order of tree traversal. If our traversal was from 14 to 20, the directions would be different but the paths would be the same.

VI. Discussion of Routing Strategies

Several routing strategies applicable to KYKLOS-II topologies have been previously proposed, including the M-2 [Jene86], H-2 [Jene86], P-2 [Mene87b] and modified P-2 [Mene88].

The M-2 routing strategy selects the minimum path within one tree. Its principal virtue is its simplicity. The M-r strategy would be to select the shortest path within one of the r trees within a network.

The H-2 routing strategy uses the concept of mid-slice, a routing being limited to the lower half of each tree. The H-r routing strategy uses the concept of equi-slices of height h, routing being limited to the lower h levels of each tree. This means that a particular 1-transform is assigned to a specific tree, thereby limiting the number of alternative routings.

The P-2 routing strategy is a minimum path strategy, using the non-pass-through routing if pass-through and non-pass-through routings are of equal length. Pass-through routing refers to a traversal of an L-node other than the source and destination nodes. The P-r routing strategy is a minimum path strategy, using minimum number of pass-throughs to arbitrate between paths of equal length.

The modified P-2 routing strategy is a minimum path strategy, using the path which is closest to the L-nodes if two or more paths are of equal length. The modified P-r routing strategy is a minimum path strategy, using the path closest to the L-nodes if two or more paths are of equal length.

The strategies described thus far are adequate to support the ($r > 2$) correlates of the M-2, H-2, P-2 and modified P-2 routing strategies. Applying these routing strategies to networks with more than two trees ($r > 2$) suggests several generalizations of theorems and definitions previously stated for KYKLOS-II<m,2,n> and KYKLOS-II<2,2,n>, for cases where $r > 2$:

Definition: A slice in KYKLOS-II<m,r,n> is the subgraph composed of the nodes from level s_t and below in tree t, such that

$$\sum_{j=0,1,\ldots,r-1} s_j = n$$

Definition: An equi-slice, or h-slice, in KYKLOS-II<m,r,n> is the slice in which

$$h = s_j,\ 0 <= j < r,\ (h = n/r)$$

Theorem: The H-2 Routing Strategy describes a unique traversal of each tree.

Theorem: If g is the number of trees with non-trivial traversals, then there are g! possible routings with H-2 Routing, all of which have the same path length. (A method for selecting the first tree to be traversed in given in Appendix A.)

Theorem: The traffic at level v using the H-2 Routing Strategy is as follows:

$0 < v <= h$: $2 * [1 - (1/m)^{h-v+1}] * N^2$;

$h < v <= n$: 0.

Theorem: The traffic per link at level v using the H-2 Routing Strategy is as follows:

$0 < v <= h$: $\{2 * [1 - (1/m)^{h-v+1}] * N^2\}/(r * f_{n-v+1})$

$= 2 * m^v * N * (m * m^h - m^v) / (r * m^{h+2})$;

$h < v <= n$: 0.

Theorem: The maximum traffic density per link using H-2 Routing occurs at level h and is the following:

$$[2 * m^h * N * (m * m^h - m^h)] / (r * m^h * m^2)$$

$$= \{[2 * (m - 1)] / (r * m^2)\} * N^{1+(1/r)}, (m^h = N^{(1/r)})$$

$$= O(N^{1+(1/r)})$$

Unexpected is the failure of the P-2 and the modified P-2 [Mene88] routing strategies to dominate the less-sophisticated H-2 routing strategy with respect to the metric of maximum traffic density per link ($T_{h2}(v,n)$). Since the P-2 and the modified P-2 strategies minimize traffic, it is intuitively appealing to believe that they would dominate the H-2 strategy. They fail to do this because both are extreme strategies. Comparing them to the H-2 strategy demonstrates this. It is possible to start with the H-2 strategy and convert its routings to P-2 or modified P-2.

A difficulty with the P-2 Routing Strategy is that it may take two H-2 traversals to level h - 1 and convert them to an h + 2 and an h - 5 traversal. This reduces the path length and total traffic, and yet it increases link traffic at level h, relative to that of H-2.

Although the modified P-2 strategy does reduce traffic at level h from P-2 in cases where P-2 might randomly select a routing with higher density at level h. Conversely, in other cases, the P-2 routing creates lower density, such as the case where the X strings for two adjoining trees are all 1's. P-2 would traverse 2h levels of one tree, modified P-2 would traverse h levels of both trees. Both P-2 and modified P-2 will increase density at level h if it reduces total traffic.

The Y-2 routing strategy starts with the routing defined by the H-2 strategy and then moves transforms from the equi-slice area of a tree to higher levels only if doing so will reduce maximum link traffic density. In order to significantly reduce maximum link traffic, it is necessary to take advantage of some of the potential for total traffic reduction. The Y-2 strategy has lower maximum traffic density for two reasons, traffic at the level of maximum traffic density (h) is moved toward the root, and total traffic is reduced. Indeed, the total traffic reduction approaches that of the P-2 or modified P-2 strategies. The criteria used to fine tune the Y-2 strategy are the primary objective of minimizing maximum link traffic and the secondary goal of reducing total traffic.

Y-2 (Hybrid) Routing Strategy (for m = 2)[8]

The Y-2 routing strategy is based on the single slice traversals of the H-2 routing strategy. In attempting to minimize maximum link traffic, it is significant that in the H-2 strategy maximum link traffic occurs at level h, the highest level in the lowest equi-slice.[9] Traffic at level h is easily reduced by moving traversals from level h to the upper levels of the tree--the challenge is to reduce traffic at level h without creating higher traffic levels at the upper levels. The Y-2 strategy achieves this in the following way:

> The H-2 routing is used for the first tree unless the last bit in the first slice and the first bit in the second slice are both on. In this case, the transforms of the first slice of tree t + 1 are moved to the second slice of tree t. In this way, 25 % of the traffic at level h (in H-2) is moved to level h + 1. However, to avoid overloading at the higher levels, the traffic beyond the first 0 above level h + 1 is moved back to tree t + 1. If there are no zeros in the second slice (and there is a third slice), the traversal may continue on into subsequent slices. The process is repeated for subsequent trees. Step-by-step procedures to implement this strategy follow:

Step 1. {Preliminaries--Setting up H-2 routing.}
Recall that H-2 Routing is contained in the r strings Xt (0 <= t < r) of length h. Under H-2 Routing, there will be r or fewer tree traversals, each traversal being of h or fewer levels.[10] Define the gth bit in each H-2 routing string Xt as Xt_{g-1}. Select the start tree. (A method for selecting the start tree to be traversed in given in Appendix B.) Let the start tree be s, the current tree be t, and the current slice in the current tree be k. Set t = s, k = 0, and lcv (loop control variable) = 0.
Note that whenever 1 or k is added to t, the mod r function should be applied. For brevity and clarity these () mod r's have been omitted.

If ($Y_s > 0$)
While (number of trees processed < r)

Step 2. {Determine highest level traversed for tree t in accordance with H-2 strategy.}
Find highest on (= 1) bit in Xt. If highest on bit is bit g, then levels (t) = g + 1.

---

[8] Demonstrated in Appendix C to give a 25 % reduction in maximum traffic density per link over H-2 Routing, if m = 2 and r = 2.
[9] Demonstrated in Appendix C.
[10] Each traversal going to the highest on (= 1) bit in the Xt routing string.

If ((levels (t) < h) or ($X(t+1)_{h-1}$ = 0))

Step 3. {If tree t traversal includes transforms from only one equi-slice, prepare to process tree t + 1.}
Increment t and lcv.

Else if (lcv + 1 < r)

Step 4. {If tree t traversal is of transforms of more than one equi-slice, determine the number to be included.}
Increment k and lcv.
While ((X(t+k) = 2^h - 1) and (lcv + 1 < r))
Calculate how many additional equi-slices are to be included in routing for tree t. Each increment in k indicates another tree which the current routing does not traverse.
Increment k and lcv.

If ($X(t+k)_h$ = 1)
Step 5. {Current tree traversal ends within next equi-slice.}
First 0 bit in X(t+k)' is g. Number of successive 0's following and including g is f.
Levels (t) = k * h + g.
Levels (t+k) = h - g - f.
{Prepare to process next tree}
t = t + k + 1
lcv = lcv + 1
k = 0

Else
Step 6. {Current tree traversal stops with last transform (inverted order) of current equi-slice.}
levels(t) = h * k
t = t + k
k = 0

End {While (lcv < r)}

If ((k > 0) and ($X(t+k)_h$ = 1) and (levels (s) = h))

Step 7. {Allow slice s to be included in the routing if tree s is not part of a multi-slice routing.}
First 0 bit in X(s)' is g. Number of successive 0's following and including g is f.
Levels (t) = k * h + g.
Levels (s) = h - g - f.

{At this point, all trees to be traversed in the routing are known, and the number of levels to be traversed in each tree is also known.}

Step 8. {Routing for the S (larger relation) fragment.}
As discussed in Section I., the routing for the S fragment to be joined is to move up the start tree (s) to the level of the join site. Be definition, this is the site of the partial join.

Step 9. {Determine the routing for the R (smaller relation) fragment}
$t = t_j$ {The initial reference point is the tree containing the partial join site, although this will be the last tree to be traversed}.
entry = j {The entry point into the first tree to be traversed is the L-node containing the fragment of relation R.}
dest = i {The routing is calculated as though going all the way to i, although it stops at the partial join site.}

```
While (lcv < r)
        t = t - 1
        If (levels(t) > 0)
                Dt = f(i)
                {The i string may be treated analogously to the X string,
                that is we may think of it is being composed of r substrings i0, i1,
                ..., i(r-1), of length h, and we may also denote the inverses of the
                substrings by '. Then, Dt = i(t-1)' cnc i(t-2)' ... cnc i(t+2)' cnc
                i(t+1)' cnc it, where cnc is the string concatenation operator.}
                traverse up the tree levels(t) levels
                {traversals up the tree are performed so that entry =
                parent(entry) as described in Appendix B.}
                If (t <> tj)
                        traverse down the tree levels(t) levels
                Else
                        traverse down the tree (levels(t) - v) levels
                        {where v is the level of the partial join site}
                {traversals down the tree are performed so that entry =
                child(entry) as described in Appendix B, and the branch taken
                (child selected) at level g is determined by examining bit g-1 in Dt,
                as discussed in Section III.}
```

{At the end of each tree traversal, entry is the name of the L-node used for pass-through from the tree just traversed to the next tree.}

End of Y-2 Routing Strategy

VII. Level Processor Loading for IB-Nodes (m = 2).

Recall that the allocation of processor loading discussed earlier gives a load per IB-node of (m - 1) * N / r = N / r. Given a routing between two L-nodes containing fragments to be joined of relations R(smaller) and S(larger), how is the join site <t, v, o> specified, where the join site is the IB-node specified by tree t, level v, node o, so that processor loading of the order of N / r will be obtained. Tree t is selected in accordance with Appendix A. IB-node o will be the ancestor of the L-node containing the S fragment in tree t.

Earlier we discussed the concept of dimensionality as applied to levels. Here we apply dimensionality to trees, using the (Y-2 tree) routing string Ys of tree s to define the points within the space of each dimension. Then the problem of applying equal loads becomes one of partitioning an r-dimensional space. The permutations of techniques for achieving this allow a wide range of choices; however, the additional constraint that the routing of a join must pass through the specified level-tree combination reduces the number of possibilities dramatically.

For the two-tree case, we consider two dimensions, represented by the (H-2 slice) routing strings of length h = n / 2, Xs and Xt, where s is the start tree and t is the other tree. {t = (s + 1) mod r} Consider the grid representing the set of points representing all the possible XOR combinations of Source and Destination nodes. This must be allocated meeting the following requirements: Balance between levels within a particular tree, balance between a particular level in different trees, and the join locations must be on the routing between the two nodes. Table 2 gives an example of one such allocation consistent with Y2 routing.

Xs	Xt': 000	001	010	011	100	101	110	111
000	<0,0>	<1,t>	<1,t>	<1,t>	<1,t>	<1,t>	<1,t>	<1,t>
001	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>
010	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>
011	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>	<2,s>
100	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>
101	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>	<1,s>
110	<3,s>	<4,s>	<3,s>	<5,s>	<3,s>	<4,s>	<3,s>	<1,s>
111	<3,s>	<4,s>	<3,s>	<5,s>	<3,s>	<4,s>	<3,s>	<6,s>

s -- Tree is start tree according to Appendix A.
t -- Tree is the other tree (not the start tree).
Xs -- H-2 routing string for tree s.
Xt'-- Inverse of H-2 routing string for tree t. If Xt' is concatenated onto the left hand side of Xs, the result is the Y-2 routing string Ys for tree s.

Table 2. <level, tree> pairs applicable to specific routing (XOR) strings, for a KYKLOS-II<2, 2, 6> network.

There are two significant attributes to the above allocation. The first is that it is consistent with the processing load division described in Section II, that is, one partial join per L-node source is allocated to the L-node, 32 partial joins are allocated to level one join sites, and each higher level has half the partial joins allocated to the next lower level. By level, the potential still exists to have N / r partial joins per IB-node. The second attribute is that the algorithm to implement it is simple.

The first step is the allocation of three special cases.

i. If Ys = 0, assign the partial join to level 0 (Assign the partial join to L-node, if both relation fragments are stored there).

ii. If Ys = N - 2, assign the partial join to level 1.

iii. If Ys = N - 1, assign the partial join to level n (The general case portion of the algorithm does not assign partial join sites above level n - 1).

The second step assigns the remaining partial joins to appropriate levels. Note that for half the routing strings should have a 0 in any bit position. However, we should assign partial join sites that are the Y-2 routing. If we assign all partial joins with routing strings that have a 0 in bit 1 (the second bit from the right) to level one, we have achieved this end[11], subject to the special cases above[12]. We then can assign half the remaining partial joins to each successive level, using a 0 in the $v+1^{th}$ bit location as the discriminator for level v.The following Level Assignment Algorithm (find v) corresponds to the above description.

---

11 Note that although this holds for the H-2 and Y-2 routing strategies it would not hold for a minimum path (or global traffic minimization) algorithm.

12 Special case i. handles the situation in which the routing will not go into the trees, thereby reducing the N / 2 assignment by 1. Special case ii. handles two difficulties: the reduction in level 1 loading due to special case i., and with special case iii., handles the remnant in the in the assignments which the general algorithm would assign based on (non-existent) bit n and following.

```
Y = Ys
v = 1
if(Y == 0) then v = 0
if(Y == N - 2) then Y = 0
if(Y == N - 1) then begin
        v = n;
        Y = 0
endif
Y = Y / m
while (Y > 0) begin
        if (Y % m) then v = v + 1
        else Y = 0
        Y = Y / m
endwhile
return v
```

This algorithm achieves processor loading of N / 2 (for m = 2 and r = 2), the processor loading objective stated above in Section II and can be made suitable for H2 routing or the HyperKYKLOS topology with minor modification. Theoretically achievable loadings are the following:

H-2 Routing:

v = 0:	1
v = 1:	$N / 2 + N^{1/2} - 1$
1 < v <= h:	N / 2
h < v <= n:	0

Y-2 Routing:

v = 0:	1
0 < v <= n:	N / 2

Partial join assignments to IB-nodes were simulated for a number of topologies with results within 10% of predicted loading for two-tree standard topologies and somewhat larger deviations for other topologies. Work continues in this area as the actualization of KYKLOS approaches.

VIII. Comparison of Various Routing Strategy Metrics

The following table compares Processor Load and Link Traffic metrics for the above algorithms and network with earlier results, including an earlier KYKLOS-II<2,2,n> [Mene87b].

Maximum Link Traffic Density

N	K-1	M-2	H-2	P-2	Y-2
4	4	3	4	3	3
16	64	36	32	26	24
64	1,024	576	256	196	192
256	16,384	9,216	2,048	1,616	1,536
1024	262,144	147,456	16,384	15,808	12,288
4096	4,194,304	2,359,296	131,072	173,568	98,304

Maximum IB-node Processing Load

N	K-1	P-2	H-2	Y-2
4	4	2	3	2
16	64	12	11	8
64	1,024	88	39	32
256	16,384	928	143	128
1024	262,144	9728	543	512

K-1 = KYKLOS-I<2,2,n> using Mid-Point Join Strategy
M-2 = KYKLOS-II<2,2,n> using M-2 Routing
H-2 = KYKLOS-II<2,2,n> using H-2 Routing and (Quasi-) Level Processor Loading
P-2 = KYKLOS-II<2,2,n> using P-2 Routing and Mid-Point Join Strategy
Y-2 = KYKLOS-II<2,2,n> using Y-2 Routing and Level Processor Loading

Table 3. Comparison of Various Routing Strategies

IX. Conclusions

The number h (= n / r) defines the equi-slice for a given (n Mod r = 0) KYKLOS topology. The KYKLOS topology defined by m = 2 and the shuffles defined by the proposed series of spans supports the semantics of H-2 and Y-2 routing strategies for all r. The H-2 routing strategy is defined by the equi-slice for all r. The Y-2 routing strategy is derivable from the H-2 routing strategy and dominates it on the metrics of total traffic, maximum link traffic and equal IB-node processor loading. The Y-2 routing strategy balances processor loading and total traffic among all trees, balances processor loading between IB-nodes of the same tree, produces maximum link traffic and maximum IB-node partial join loads less than previously described routing strategies and addresses the preceding problems in KYKLOS networks with more than two trees.

REFERENCES

[Bene65] V. E. Benes, Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, 1965.

[Horo81] Ellis Horowitz and Alessandro Zorat, "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI", *IEEE Transactions on Computers*, 1981.

[Jene86] Roy M. Jenevein and Bernard L. Menezes, "KYKLOS: Low Tide High Flow," *Proceedings of the Sixth International Conference on Distributed Computing*, May, 1986.

[Lee59] C. Y. Lee, "Representation of Switching Circuits by Binary Decision Programs", *Bell Systems Technical Journal*, 1959.

[Mene85] B.L.Menezes and R.M.Jenevein, "KYKLOS: A Linear growth Fault-tolerant Interconnection Network," *Proceedings of the International Conference on Parallel Processing*, August, 1985, pp. 498-502.

[Mene86] Bernard L. Menezes and Roy Jenevein, "Managing Combinatorial Explosions in a Family of Interconnection Networks", *Proceedings of International Computer Symposium*, December, 1986.

[Mene87a] B.L.Menezes, K.Thadani, A.G.Dale, R.Jenevein, "Design of a HyperKYKLOS-Based Multiprocessor Architecture for High-Performance Join Operations", Dept. of Computer Sciences, Technical Report TR-87-18, the University of Texas at Austin, Austin, Texas, May, 1987.

[Mene87b] B.Menezes, D.Brant, D.Loewi, A.G.Dale, R.Jenevein, "An Interconnection Network Supporting Relational Join Operations", Dept. of Computer Sciences, Technical Report TR-87-25, the University of Texas at Austin, Austin, Texas, June, 1987.

[Mene88] Bernard L. Menezes, "The KYKLOS Multicomputer Network: Interconnection Strategies, Topological Properties, Applications", Ph.D. Dissertation, Department of Electrical and Computer Engineering, the University of Texas at Austin, Austin, Texas, May, 1988.

[Sing81] A. D. Singh, F. G. Gray, and J. R. Armstrong, "Tree Structured Sequential Multi-Valued Logic Design from Universal Modules", *IEEE Transactions on Computers*, 1981.

Appendix A. Partial Join Assignment to Tree.

Fragment Xt is used to determine which fragments are assigned to tree t. The tree assignment algorithm is the following:

1. Consider a fragment S_{ij}, originating at L-node i, and to be joined with fragment R_{ji};
2. Determine $X = XOR_m (i, j)$;
3. The start tree is

$$s = (n * N - \text{bitcount}(j) - j / r) \bmod r;$$

4. Assign the S_{ij}-R_{ji} partial join to the tree t

a. If X = 0 (all Xt's = 0), the partial join is assigned to the L-node, (which belongs to all trees).

b. If Xs > 0, t = s.

c. If Xs = 0 and X > 0, assign the partial join to the first tree greater than s such that Xt > 0. (For all t such that Xt > 0, t = (s + k_t) mod r, and k_t > 0, select the t with the smallest k_t.)

5. Step 7. of the Y-2 Routing Strategy may require that the partial join site tree be changed, although the level of the partial join site would not be changed.

Appendix B. Connection Sequences for L-nodes and IB-nodes.

The labeling sequences for L-node and IB-nodes are to assign increasing integers from left to right. Thus, the IB-node ancestors of L-node 0 will be <t,1,0>, <t,2,0>, . . ., <t,v,0>, . . ., <t,n,0>.

The concept of $span_{tv}$ was described in Section IV. In describing connections of nodes, a useful related concept is real span, referred to hereinafter as r-$span_{tv}$. This refers to the span of a IB-node based on its children rather than the lowest L-nodes in the subtrees of its children. While the span concept is more useful in developing a taxonomy of interconnection networks, r-span is more useful in considering problems of construction. This is because r-span allows consideration of connections between nodes of successive levels (level of L-nodes = 0), without reference to the L-nodes (except at the lowest IB-node level, 1).

The following Table B-1 sets forth real spans for several KYKLOS networks. The lesser real spans in the proposed KYKLOS-II suggests that if construction difficulty is related to number of link crossings, the proposed KYKLOS-II may be more easily constructed than the original topology.

d_{tv}

	Original KYKLOS-II <2, 2, 6>		Proposed KYKLOS-II <2, 2, 6>		Proposed KYKLOS-II <2, 3, 6>		
tree	0	1	0	1	0	1	2
level							
6	0	0	0	0	0	0	0
5	0	1	1	1	1	1	1
4	0	2	2	2	0	2	0
3	0	3	0	3	1	3	1
2	0	4	0	3	0	2	4
1	0	5	0	3	0	2	4

$r\text{-}span_{tv} - 1$ = maximum number of link crossings

6	0	0	0	0	0	0	0
5	0	1	1	1	1	1	1
4	0	3	3	3	0	3	0
3	0	7	0	7	1	7	1
2	0	15	0	7	0	3	15
1	0	31	0	7	0	3	15

Total number of link crossings[13]

6	0	0	0	0	0	0	0
5	0	1	1	1	1	1	1
4	0	6	6	6	0	6	0
3	0	28	0	28	4	28	4
2	0	120	0	56	0	24	120
1	0	496	0	112	0	48	240
Tree totals:	0	651	7	203	5	107	365
Network totals:		651		210			477

Table B-1. Comparison of link crossing metrics in selected networks.

[13] Total link crossings at level v = $2^{n-v-d_{tv}} * (r\text{-}span_{tv} - 1)$

Real span (r-span$_{tv}$) can be computed for level v in tree t by the following formulation:

1. Level v in tree t has associated with it an implicit dimension, z_{tv}, and a nominal span, span$_{tv}$. These may be computed as described in Section IV.
2. Compute d_{tv}, where d_{tv} is the number of dimensions whose associated level is greater than v, and whose dimension is less than z_{tv}.

   Thus, $d_{tv} = \underset{gl = v+1, \ldots, n}{\text{count}} (z_{tg} < z_{tv})$
3. Compute r-span$_{tv} = m^{d_{tv}}$

r-span$_{tv}$ can be used either to determine the parent of a child, or the children of a parent:

Computing the parent of child <t, v, o>:

1. u = r-span$_{t(v+1)}$
2. $w = o \bmod u + u * [o \operatorname{div} (m * u)]$
3. parent <t, v+1, w>

Computing the child of parent <t, v, o>:

1. u = r-span$_{tv}$
2. $w_0 = o \bmod u + m * u * [o \operatorname{div} u]$
3. $w_1 = w_0 + u$

   . . .

   $w_c = w_0 + c * u$

   . . .

   $w_{m-1} = w_0 + (m - 1) * u$
4. The m children of <t, v, o> are <t, v-1, w_0>, <t, v-1, w_1>, . . ., <t, v-1, w_c>, . . ., <t, v-1, w_{m-1}>.

Appendix C. Maximum Link Traffic Density using Y-2 Routing.

A simple way of looking at link traffic density in H-2 routing is to look at the r routing strings of length h from the top down. For a given string, $((m - 1) / m = 1 - 1 / m)$ is the probability of reaching level h. Similarly for level h - 1, $1 - (1 / m)^2$, the probability of an on bit in the h - 1 or h - 2 positions. Thus, the probability of reaching level v in any tree t is $1 - (1 / m)^{h-v+1}$ $(= 1 - m^v / (m * m^h))$

The number of links in any tree at level $v = m * m^{n-v}$, while the number of joins to be performed in the network $= N^2$. The foregoing can be combined for the following results:

Traffic Density per Link using H-2 Routing:

$0 < v <= h$:

$$T_{h2}(v,n) = \{N^2 * [1 - m^v / (m * m^h)]\} / (m * m^{n-v})$$

$$T_{h2}(v,n) = 2^v * N * (2 * 2^h - 2^v) / 2^{h+2}$$

$h = v$:

$$T_{h2}(h,n) = 2^h * N * (2 * 2^h - 2^h) / 2^{h+2}$$

$$T_{h2}(h,n) = N * 2^h / 4$$

$h < v <= n$:

$$T_{h2}(v,n) = 0$$

Maximum Traffic Density per Link:

$$T_{h2}(v,n) = 2^v * N * (2 * 2^h - 2^v) / 2^{h+1},\ 0 < v <= h$$

Let $v = h - d,\ 0 <= d < h$

$$T_{h2}(d,n) = 2^{h-d} * N * (2 * 2^h - 2^{h-d}) / 2^{h+1},\ 0 <= d < h$$

$$T_{h2}(d,n) = (N / 2) * 2^h * 2^{-d} * (2 - 2^{-d}),\ 0 <= d < h$$

The first two terms are unrelated to d, the last two terms are largest when d is smallest. Thus, this function is maximized when $d = 0 => v = h$.

The Y-2 Routing Strategy is based on reducing traffic at level h. The reduction achieved is at least 25 % if $m = 2$. This is demonstrated below for the most extreme case, $r = 2$.

Traffic Density per Link after Y-2 Step 4:

Step 4 removes 1 / 4 of the traffic at level h in the first two trees. For r > 2, additional amount are removed based on trees other than the start tree, so that the total reduction is between 1 / 4 and 1 / 3. The following is based on the r = 2 case as it is the most restricted:

$0 < v < h$:

$$T_{y2}(v,n) = 2^v * N * (2 * 2^h - 2^v) / 2^{h+2} - (N * 2^v) / 16$$

$$= (2^v * N) * [(2^h - 2^{v-1}) / 2^{h+1} - 1 / 16]$$

$v = h$:

$$T_{y2}(v,n) = (3 * N * 2^h) / 16$$

$h < v <= 2h = n$:

$$T_{y2}(v,n) = (N * 2^v) / 16$$

Traffic Density per Link after Y-2 Steps 5 and 6.:

This sub-step removes the transforms of a routing at levels greater than h+g from the current tree if the level h+g transform is a 0-transform. As this 0-transform is then removed from the routing, this step removes traffic on the upper part of the tree, by transferring it to the lower tree (at levels below h), or eliminating it from the routing. Half the traffic moved to the level h + 1 is removed by one of these actions. Likewise, for three-fourths of the traffic at level h + 2, and similarly, for the higher levels.

$0 < v < h - 1$:

$$T_{y2}(v,n) <= [(2^h * N) * 25] / 256$$

$v = h - 1$:

$$T_{y2}(v,n) = (N * 2^h * 5) / 32$$

$v = h$:

$$T_{y2}(v,n) = (N * 2^h * 3) / 16$$

$h < v <= 2h$:

$$T_{y2}(v,n) = (N * 2^h) / 8$$