Pergamon

Mathl. Comput. Modelling Vol. 21, No. 11, pp. 23–33, 1995

0895-7177/95 $9.50 + 0.00

0895-7177(95)00076-3

Topological Properties of the Recursive Petersen Architecture

A. A. NANAVATI AND S. S. IYENGAR
Department of Computer Science, Louisiana State University
Baton Rouge, LA 70803, U.S.A.

A. EL-AMAWY
Department of Electrical Engineering, Louisiana State University
Baton Rouge, LA 70803, U.S.A.

(Received January 1995; accepted March 1995)

Abstract—The Petersen graph is gaining popularity as an interconnection network because of its several interesting properties. The recursive Petersen architecture is very compact and has some very interesting topological properties. In this paper, we study its topological properties in detail. Two labeling schemes are suggested. Basic routing and broadcasting have been discussed. The most attractive features are its logarithmic (to the base 10) diameter and high symmetry.

Keywords—Parallel processing, Distributed systems, Interconnection networks, Graph theory, Parallel architectures.

1. INTRODUCTION

The search for an ideal network topology for parallel processing has yielded several architectures. As is common with such problems, no architecture can claim to be the best possible one for all applications. Among the suggested architectures, the Hypercube (HC) is quite popular because of its high symmetry, good embedding properties and a logarithmic diameter. This has resulted in the evolution of several cube-oriented networks which incorporate the Hypercube within them. Examples of such networks are the Folded Hypercube [1], Cube-Connected Cycles [2], Banyan Hypercube [3], Bridged Hypercube [4], and the Hypertree [5]. Popular examples of non-cube-oriented architectures are the De Bruijn Multiprocessor network [6] and Shuffle Exchange [7,8].

These extensive studies have isolated properties that are considered desirable for architectures to have. It is well understood that the suggested topology should have a small diameter to economise communication; it should be easily extensible so that processors can be easily added to enhance the size of the network while preserving some of the fundamental properties of the network of the smaller size; it should have good embedding properties to be able to simulate well various architectures that are known to be good at particular applications, allowing it to be useful for more number of applications. In addition, every node should have a small degree, and the network should be highly connected. The Recursive Petersen architecture [9], a non-cube-oriented network and the object of this study, has several attractive properties.

We are grateful to S. Kundu of the Department of Computer Science and B. Oporowski of the Department of Mathematics, Louisiana State University, for several fruitful discussions and insightful comments. We also thank J. Barhen (ORNL) for his suggestions regarding the presentation of the material.

2. THE PETERSEN GRAPH

In this section, we introduce some graph-theoretic terminology and present some properties of the Petersen graph.

2.1. Terminology

We define some graph theoretic terms [10,11] that are relevant to subsequent discussions, following which, we examine some interesting properties of the Petersen graph.

A graph $G = (V, E)$ consists of a set of objects $V = \{v_1, v_2, \ldots\}$ called *vertices*, and another set $E = \{e_1, e_2, \ldots\}$, whose elements are called *edges*, such that each edge e_k is identified with an unordered pair (v_i, v_j) of vertices.

When a vertex v_i is an end vertex of some edge e_j, v_i and e_j are said to be *incident* with each other. The number of edges incident on a vertex v_i is called the *degree* of the vertex v_i. A graph in which all vertices are of equal degree is called a *regular graph.*

A graph in which there exists an edge between every pair of vertices is called a *complete graph.* K_n denotes the complete graph on n vertices.

The *diameter* of a graph is the largest distance between two vertices in the graph. The *girth* of a graph G, is the length of a shortest cycle (if any) in G. A graph is *connected* if every pair of points are joined by a path.

A maximal connected subgraph of G is called a *connected component* or simply a *component* of G. A *bridge* is an edge whose removal increases the number of components.

A graph g is said to be a *subgraph* of a graph G if all the vertices and all the edges of g are in G, and each edge of g has the same end vertices in g as in G. A *spanning subgraph* is a subgraph containing all the points of G.

A *factor* of a graph G is a spanning subgraph of G which is not totally disconnected. An *n-factor* is regular of degree n. If G is the sum of n-factors, their union is called an *n-factorisation* and G itself is *n-factorisable.*

An *n-cage*, $n \geq 3$, is a cubic graph (regular with degree 3) of girth n with the minimum possible number of vertices.

2.2. Properties of the Petersen Graph

The Petersen graph, henceforth known as P, gets its name after its discoverer, Julius Petersen, a Danish mathematician, who used it to show that not every 3-regular bridgeless graph is 1-factorable. Petersen also proved that every 3-regular graph (with at most two bridges) can be factored into a 2-factor and a 1-factor. P is a bridgeless, cubic, regular graph of 10 vertices and 15 edges, connected as shown (Figure 1). The graph has diameter 2. This is the best that can be achieved to accommodate 10 vertices, each of degree 3. This advantage stems from the fact that the pentagram has been connected to the pentagon. Should two pentagons be connected (one inside the other and corresponding vertices adjacent), the diameter of the graph would be 3.

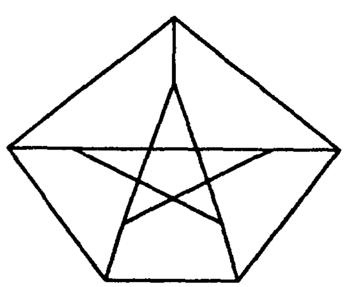


Figure 1. The Petersen graph.

2.3. Topological Properties

P has several interesting features, some of which are listed here, without proof. The proofs can be found in any standard text on graph theory:

1. P is the largest graph which is 3-regular and has diameter 2. In other words, it is not possible to connect more vertices with these constraints. It is the most compact.
2. P is the only 3-regular graph (apart from K_4), in which any two nonadjacent vertices are mutually adjacent to just one other vertex.
3. P is the smallest graph with the property that, given three distinct vertices u, v and w, there is a fourth vertex adjacent to u, but not to v or w.
4. P is vertex transitive, that is, it has automorphisms mapping any given vertex to any other. It is also distance transitive, in that, whenever the distance from v to w is the same as from v' to w', there is an automorphism taking v to v' and w to w'. Actually, distance transitivity implies vertex transitivity.
5. P is strongly regular, which means that the number of vertices mutually adjacent to any pair of adjacent vertices is constant and the number of vertices mutually adjacent to any pair of nonadjacent vertices is also constant.
6. P is the smallest graph that contains a minor of K_5 as well as a subdivision of $K_{3,3}$, and is therefore nonplanar.
7. P is the only 5-cage, the only 3-regular graph with girth 5.
8. P is a bridgeless, cubic graph and a sum of a 1-factor and a 2-factor. The pentagon and the pentagram together constitute a 2-factor, and the five lines joining the pentagon with the pentagram form a 1-factor (Figure 2).
9. P is non-Hamiltonian.

Some interesting properties are listed in [12]. There are several other features which may be of more interest to graph theorists. Some other properties of P are presented with proofs.

PROPOSITION 2.1. *Any two adjacent vertices A and B of a Petersen graph are such that the vertices adjacent to A and those adjacent to B are not adjacent.*

PROOF. We can prove the proposition by contradiction. Let us assume that there exist a vertex A' adjacent to A and a vertex B' adjacent to B. Furthermore, let us assume that A' and B' are adjacent. In such a case, $ABB'A'$ form a cycle of length four. But the girth of the Petersen graph is five, and so it does not contain a cycle of length four.

PROPOSITION 2.2. *The vertex connectivity and the edge connectivity of the Petersen graph is 3.*

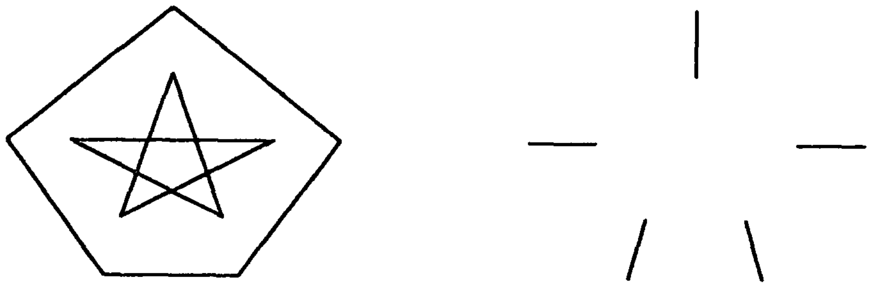
PROOF. Since the Petersen graph is a regular graph with each vertex of degree three, it follows, from Menger's theorem, that the vertex connectivity as well as the edge connectivity of the graph is 3.

PROPOSITION 2.3. *There exist 3-vertex disjoint paths between every pair of distinct vertices in the Petersen graph.*

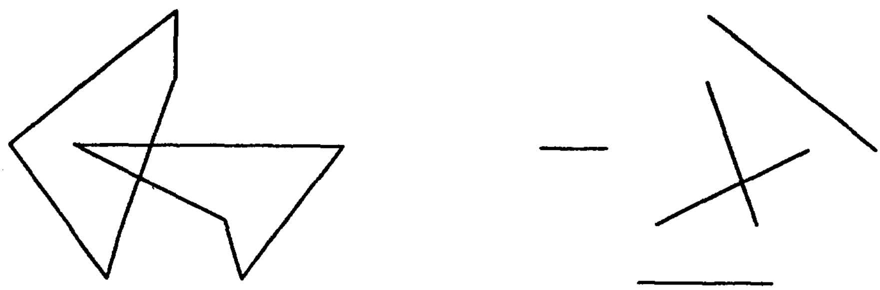
PROOF. The proof follows immediately from Proposition 2.2.

PROPOSITION 2.4. *The Petersen graph can be expressed as the sum of a 2-factor and a 1-factor in two different ways as shown (Figures 2(a) and 2(b)).*

PROPOSITION 2.5. *Every pair of distinct vertices in a Petersen graph is included in a cycle of length five.*



(a) The 2-factor and the 1-factor of the Petersen graph.



(b) Another 2-factor and the 1-factor of the Petersen graph.

Figure 2.

PROOF. Consider a pair of vertices (x, y). Two cases, upto isomorphism, arise:

1. Both of them belong to the pentagon or the pentagram in the 2-factor (Figure 2(a)).
2. If they do not, then they belong to a pentagon in the 2-factor (Figure 2(b)).

There is no other situation.

2.4. Labeling

For any architecture, the underlying graph structure has to be labeled because every vertex in the graph corresponds to a processor in the network and has an address. The labeling scheme should be aimed at optimising communication costs. At the same time, it should allow for simplicity in communication algorithms. We propose two labeling schemes (Figure 3). It is impossible to label P such that the addresses of neighbouring vertices differ in exactly one bit, because the graph contains cycles of odd length. Consider the factors of the Petersen graph as shown (Figure 2(a)). The set of vertices V can be partitioned into two disjoint subsets: V_o (outer), the vertices of the pentagon, and V_i (inner), the vertices of the pentagram.

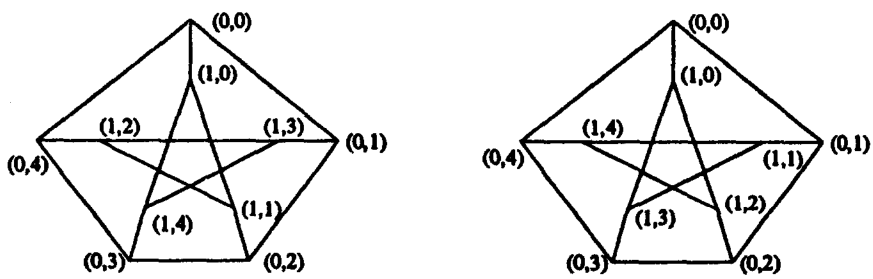


Figure 3. The two labelings (1,2) of the Petersen graph.

Each vertex, v, is identified by a 2-tuple (s, n), where

$$s = \begin{cases} 0, & \text{if } v \in V_o, \\ 1, & \text{if } v \in V_i, \end{cases}$$

and $0 \leq n \leq 4$. Also, $V_i \cup V_o = V$ and $V_i \cap V_o = \phi$, where $V = \{i \mid 0 \leq i \leq 9\}$.

Two possible labeling schemes, 1 and 2, are shown (Figure 3). Each set of vertices, V_i and V_o, form a cycle of length five as can be seen from the two factor.

LEMMA 2.1. *The Petersen graph can be labeled in 20 different ways according to the above mentioned labeling method (Figure 3).*

PROOF. Consider the labels given (Figure 3). Without loss of generality, consider the vertex labeled $(0,0)$. Any one of the set V_o could be labeled $(0,0)$, which gives five possibilities. The labeling can be done counterclockwise (instead of clockwise as shown), which would again give a graph isomorphic to the situation shown. That leads to two possibilities. The set V_i could be switched with the set V_o, which again doubles the number of possible labels. Therefore, the total number of ways in which the Petersen graph can be labeled is $5 \times 2 \times 2 = 20$.

2.5. Distances

The distance $d(v_i, v_j)$ between any two vertices in an undirected graph is a metric, and therefore, $d(v_i, v_j) = d(vj, vi)$. In the Petersen graph, the distance has to be calculated from the labels (s_i, n_i) and (s_j, n_j) of the two vertices, so we define a commutative operator, $\odot$, as follows:

$$n_i \odot n_j = n_j \odot n_i = \min\left(n_i \circ n_j, n_j \circ n_i\right), \tag{1}$$

where min denotes the minimum of the two values and $\circ$ denotes subtraction modulo 5. Since min is commutative, so is $\odot$. We first look at the adjacency condition between any two vertices; as from these, the distance formula can be easily derived. This is so because the distance between any two vertices in the Petersen graph cannot exceed 2. Therefore, just three conditions arise:

1. The two vertices are not distinct (distance is 0).
2. The two vertices are adjacent (distance is 1); and
3. The vertices are nonadjacent (distance is 2).

THEOREM 2.1. *In labeling scheme 1, two vertices $v_i(s_i, n_i)$ and $v_j(s_j, n_j)$ are adjacent iff*

$$adj\,(v_i, v_j) = \begin{cases} n_i \odot n_j = 1, & \text{if } s_i = s_j, \\ 2n_i \odot n_j = 0, & \text{if } s_i = 0,\ s_j = 1. \end{cases}$$

PROOF. The proof is clear from the labeling scheme.

THEOREM 2.2. *The distance $d(v_i, v_j)$ between any two vertices v_i and v_j, in labeling 1, is given by*

$$d(v_i, v_j) = d(v_j, v_i) = \begin{cases} n_i \odot n_j, & \text{if } s_i = s_j, \\ 2n_i \odot n_j + 1, & \text{if } s_i = 0,\ s_j = 1. \end{cases}$$

PROOF. The proof is clear from the labeling scheme.

THEOREM 2.3. *According to labeling 2, two vertices $v_i = (s_i, n_i)$ and $v_j = (s_j, n_j)$ are adjacent iff*

$$adj\,(v_i, v_j) = \begin{cases} n_i \odot n_j = 1 + s_i, & \text{if } s_i = s_j, \\ n_i = n_j, & \text{otherwise}. \end{cases}$$

PROOF. The proof is clear from the labeling scheme.

THEOREM 2.4. *The distance $d(v_i, v_j)$ between any two vertices v_i and v_j, in labeling 2, is given by*

$$d(v_i, v_j) = d\,(v_j, v_i) = \begin{cases} n_i \odot n_j, & \text{if } s_i = s_j = 0, \\ 3 - (n_i \odot n_j), & \text{if } s_i = s_j = 1, \\ n_i \odot n_j + 1, & \text{otherwise}. \end{cases}$$

PROOF. The diameter of the graph is 2. Hence, $d(v_i, v_j) = d(v_j, vi) \leq 2$. If $si = s_j = 0$, then $v_i, v_j \in V_o$; and both lie on the outer cycle (pentagon). The operator $\odot$ gives the distance between the two on the cycle. When $s_i = s_j = 1$, then $v_i, v_j \in V_i$; and both lie on the inner cycle (pentagram). Since adjacent labels differ in their second tuple by 2 in the pentagram and nonadjacents differ by 1, the difference given by the operator is 3-complemented. When the vertices belong to different groups, if they have the same second tuple, they are at distance 1 (adjacent), otherwise at distance 2.

Either of the two labelings described above can be used. In the rest of the paper, we will follow labeling 2. Without affecting significant features, labeling 1 could have been used instead.

3. TOPOLOGICAL PROPERTIES OF THE RECURSIVE PETERSEN GRAPH

1. Number of Vertices(P): The total number of vertices in an RP of dimension p is 10^p.
2. Number of Edges(E): The total number of edges in an RP of dimension p is
$$\frac{10^p \times 3p}{2}.$$
3. Degree(K): The degree of each vertex in a p-dimensional RP is $3p$.
4. Diameter(D): The diameter of a p-dimensional RP is $2p$.
5. Connectivity(V): The node connectivity of a p-RP is $3p$.
6. Cost(C): The cost is estimated by the product of the degree of any vertex and the diameter of the graph. For a p-RP, it is $3p \times 2p$.
7. Average Distance(d_{ave}): To calculate the average distance, we need to calculate the total distance (σ_d) of all distances from any particular vertex. From [9, Theorem 2], we know that
$$\sigma_d = n_1 \times \sigma_{d2} + n_2 \times \sigma_{d1},$$
where n_1 and n_2 represent the number of vertices, e_1 and e_2, the number of edges and σ_{d1} and σ_{d2} represent the total distances in the graphs G_1 and G_2, respectively, and σ_d denotes the total distance in $G_1 \times G_2$. The number of edges in $G_1 \times G_2$ is given by
$$e = n_1 \times e_2 + n_2 \times e_1.$$
Since, in the Petersen graph, $\sigma_d = e$, it follows from the above statements that for higher dimensions too, this property is preserved [9] (see message traffic density next). The average distance is given by
$$\frac{\sigma_d}{n} = \frac{e}{n} = \frac{10p \times 3p/2}{10p} = \frac{3p}{2}.$$
8. Message Traffic Density(M): The message traffic density of the p-RP is
$$\frac{(e_p/(P-1)) \times P}{e_p} \approx 1.$$
The message traffic density (or the average message passing density) is optimal in the following sense. A value of $M > 1$ implies that there could be congestion of messages on a link, which would result in inefficiency. A value of $M < 1$ would indicate the presence of too many edges and hence an increase in cost. A value of $M = 1$ signifies that the message traffic density is optimal. The set of graphs with the property $M = 1$ are both economic and efficient in the aforementioned sense and are denoted by $\mathcal{E}^2$. It has also been shown that the set $\mathcal{E}^2$ is closed under the Cartesian product operation and that in the class of Moore graphs K_1, K_2 and P are the only ones which belong to $\mathcal{E}^2$ [9]. It has been shown in [13] that in the class of the generalised Petersen graphs, only the 2-cube and P belong to $\mathcal{E}^2$.

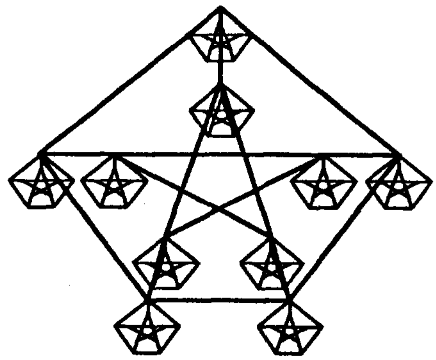


Figure 4. The 2-*RP* (thick lines denote links connecting corresponding pairs of the Petersen graphs).

3.1. Labeling

The *RP* can be labeled in a recursive fashion by extending the strategy of labeling the Petersen graph. A *p-RP* would therefore require p 2-tuples described above. A *p-RP* can be visualised as a Petersen graph with every vertex of the Petersen graph being a $(p-1)$-*RP*. The first 2-tuple isolates a $(p-1)$-*RP*, the second two-tuple isolates a $(p-2)$-*RP* within it, and so on. The p^{th} 2-tuple indicates the vertex. Thus, the address is complete. Figure 4 shows the 2-*RP*.

3.2. Routing and Broadcasting

To do routing and broadcasting efficiently, we first try to find a shortest path algorithm for any pair of vertices in P. Then we modify the algorithm to route and broadcast in any *p-RP*.

3.3. Shortest Path Algorithm

THEOREM 3.1. *There exists a unique shortest path between any pair of vertices in the Petersen graph.*

PROOF. Let us assume that the shortest path between any two vertices is not unique. Since the path length of any path in the graph cannot exceed 2 (since 2 is the diameter of the graph), the paths can be of length 0, 1, or 2. Since there are no self-loops in the Petersen graph, all paths of length 0 are unique. There are no parallel edges either; therefore, all paths of length 1 are unique. A nonunique path of length 2 implies the existence of a cycle of length four in the graph, but the Petersen graph has girth five.

ALGORITHM FOR COMPUTING THE SHORTEST PATH IN THE PETERSEN GRAPH. Let V denote the set of vertices and E the set of edges in the Petersen graph. Consider the graph to be labeled according to labeling 2. The computation of the shortest path can be divided into two cases. Let the source vertex be $v_i(s_i, n_i)$ and the destination vertex, $v_j(s_j, n_j)$. The case where $i = j$ is trivial. In the case of distinct vertices, the following possibilities arise:

1. $s_i = s_j = 0$. If $n_i \odot n_j = n_i \circ n_j$, the shortest distance is from j towards i; therefore, the path taken should be $(s_i, n_i) \rightarrow (s_j, n_i \circ 1) \cdots \rightarrow (s_j, n_j)$. Else, the shortest distance is from i to j, and the shortest path is $(s_i, n_i) \rightarrow (s_j, ni \oplus 1) \cdots \rightarrow (s_j, n_j)$, where $\oplus$ is addition modulo 5.

2. $si = s_j = 1$. If $n_i \odot n_j = n_i \circ n_j$, the shortest distance is from j towards i; therefore, the path taken should be $(s_i, n_i) \rightarrow (s_j, n_i \circ 2) \cdots \rightarrow (s_j, n_j)$. Else, the shortest distance is from i to j, and the shortest path is $(s_i, n_i) \rightarrow (s_j, n_i \oplus 2) \cdots \rightarrow (s_j, n_j)$.

3. $s_i \neq s_j$. We consider three subcases.

(a) $n_i = n_j$. The vertices are adjacent and the shortest path is $(s_i, n_i) \to (s_j, n_j)$.
(b) $s_i = 0, s_j = 1$. If $n_i \odot n_j = 1$, the shortest path is $(s_i, n_i) \to (s_i, n_j) \to (s_j, n_j)$. Else, the shortest path is $(s_i, n_i) \to (s_j, n_i) \to (s_j, n_j)$.
(c) $s_i = 1, s_j = 0$. If $n_i \odot n_j = 2$, the shortest path is $(s_i, n_i) \to (s_i, n_j) \to (s_j, n_j)$. Else, the shortest path is $(s_i, n_i) \to (s_j, n_i) \to (s_j, n_j)$.

PROOF. The case $i = j$ is trivial. In case 1, $n_i \circ n_j$ indicates whether v_j is closer to v_i along the clockwise or the counterclockwise direction; that is, whether a sequence of modulo 5 additions or a sequence of modulo 5 subtractions gives the shortest path. The same is true for case 2, except that in this case, adjacent vertices differ by 2 in their second tuple and not by 1 as in case 1. Case 3(a) and 3(b) differ for exactly the same reason. Case 3(a) is obvious from the labeling.

3.4. Algorithms for Routing and Broadcasting

Owing to the fact that the *RP* is the product of the Petersen graph with itself, it is not surprising that almost everything that is applicable to the Petersen graph can be recursively applied to the *RP*.

An algorithm for routing in the *RP* follows. The idea is based on the following notion. A p-dimensional *RP* can be looked upon as P where every vertex of P actually denotes a $(p-1)$-dimensional *RP*. Formally, the p 2-tuples of the source and the destination are used as follows. Without loss of generality, let the first (most significant) i 2-tuples be identical in the two addresses. This implies that the source and destination belong to the same $(p-i)^{\text{th}}$ *RP*. Using the links of the $(p-i)^{\text{th}}$ *RP*, the procedure *route_petersen* reaches the destination $(p-i-1)^{\text{th}}$ *RP*. The address of the current node differs from the destination in one less tuple than the source and so on.

ALGORITHM FOR ROUTING IN *RP*.

```
route_rp(src,dest)

/*
 *      src : p-2-tuple address of source.
 *      dest : p-2-tuple address of destination.
 */
begin
        while (src -dest)
        begin
              d := the most significant differing 2-tuple in src and dest.
              i := dth 2-tuple of src
              j := dth 2-tuple of dest
              route_petersen (d,i,j)
        end
end

route_petersen (d,i,j)
begin
/* use links of the dth dimension RP, i.e., RP of dimension (d-1) is considered as a vertex */
        if (i= j)
            send message to local processor
        elseif (group(i) = group(j)) /* they both belong to set V_i or V_o */
            calculate modulo difference
            send message over the shorter path
        /* otherwise, the src and the destination belong to different groups */
```

elseif (i and j adjacent)
send message to j
else
find a common neighbour, u
send message through neighbour u
end

PROOF. The correctness of *route_petersen()* comes from Section 3.4.1. The procedure *route_rp()* calls *route_petersen()* for all differing 2-tuples, starting with the most significant, till there are no differing tuples left. After each call to *route_petersen*, it is clear that the distance between the intermediate destination and the final destination decreases by at least 1. So, *route_rp()* will eventually stop and *dest* will be reached.

For the purpose of broadcasting, it is convenient to define a *Petersen tree.*

3.5. Petersen Tree

The Petersen tree is a spanning tree of the Petersen graph. It is ternary. The tree is shown in Figure 5. The Petersen tree (henceforth p-tree) is a complete 3-ary tree where every vertex is of degree 3. The depth of the tree is 2.

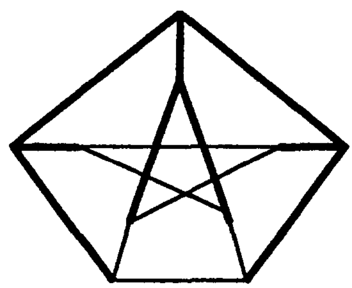


Figure 5. The Petersen tree.

If all ports transmit and receive is allowed, then a message can reach all the vertices in 2 time units. If single port transmit and/or receive is allowed, a message can reach in 5 time units.

Figure 6 shows a p-tree of dimension 2. The p-tree is the broadcasting tree. The depth of the p-tree of dimension 2 is only 4, which means that it takes only 4 units of time to broadcast a message assuming it is possible to simultaneously send a message over all links. The recursive nature of the p-tree is evident. To draw a p-tree of dimension 2, one can draw a p-tree of dimension 1 with all the nodes of the tree being replaced with p-trees of dimension 1. Similarly, a p-tree of dimension 3 can be drawn by replacing each node of a p-tree of dimension 1 with a p-tree of dimension 2 and so on.

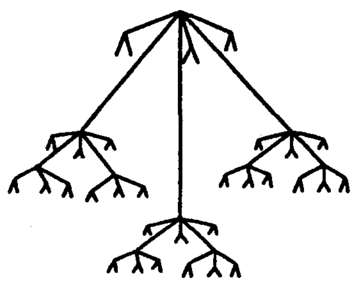


Figure 6. The broadcasting p-tree for the 2-RP.

The basic idea of the broadcasting algorithm is as follows. Each node is a vertex of a Petersen graph in several dimensions. The i^{th} 2-tuple of the p 2-tuple (assuming a p-RP) indicates the

Table 1. Comparison of topological features.

Topology	P	E	K	D	V	C	d
HC	8	12	3	3	3	9	1.5
	64	192	6	6	6	36	3
	2K	11.3E3	11	10	11	121	5.5
	1M	10.5E6	20	20	20	400	10
FHC	8	16	4	2	4	8	1.25
	64	224	7	3	7	21	2.41
	2K	12.3E3	12	6	12	72	4.65
	1M	11.0E6	21	10	21	210	8.65
CCC	8	8	3	4	3	12	2
	64	96	3	9	3	27	4.63
	2K	3.07E3	3	19	3	57	10.55
	1M	1.57E6	3	39	3	117	22.47
RP	10	15	3	2	3	6	1.5
	100	300	6	4	6	24	3
	1000	4500	9	6	9	54	4.5
	1000000	9E6	18	12	18	216	9

position of the node in the i^{th} dimension(level) Petersen graph. To broadcast, any node must send the message to its 3 neighbours at every level. Every receiving node must then send it to its neighbours (at the same neighbour-level as that of the sender) and to all neighbours at lower levels. To prevent the same message from reaching the same node more than once, a table of 10 p-trees (dimension 1) with every node being the root should be stored. From the table, each node can decide its i-level neighbours given the address of the node they receive the message from. For each lower level, the originator decides the tree (1 of 10) according to which messages should be sent. After each iteration, it is assured that subsequent tuples will be serviced and thus eventually all nodes will receive the message.

ALGORITHM FOR BROADCASTING IN THE *RP*.

broadcast()
if originator then send message to all neighbours
else
  i = level of sender address
  send messages to all neighbours at level i from the selected tree
  send messages to all neighbours at lower levels
end

4. CONCLUSIONS

The *RP* compares and competes well with the hypercube. Table 1 summarises the comparisons of the hypercube, some of its variants and the *RP*. The *RP* is very compact and highly symmetric. It has a logarithmic diameter (to the base 10). Due to its structure, for the same degree, it can accommodate many more vertices. A 3-*RP* has 1000 nodes, each of degree 9 and a diameter of only 6! From the table, it is evident that, for nearly the same number of nodes compared to the *HC* and the *FCC*, the *HP* has fewer edges, lesser diameter, optimal average message passing density for the same connectivity. Its cost is significantly lower than the hypercube. Over CCC, it has the advantage that it is for enhancement, and no existing connections have to be altered.

For implementation purposes, the neighbour computation need not be done every time; instead, a table can be stored to save computation. The embedding properties of the *RP* have not been explored in this paper, but the RP appears to be promising by the virtue of its symmetry. Also, it is known that a complete binary tree can be embedded in P with dilation-1, expansion-1. The hope is that the recursive nature of RP should therefore be able to embed a complete binary tree with expansion-1, dilation-1.

From the study of the Generalised Petersen graphs [13] and Moore graphs [9], it appears that K_1, K_2, and the Petersen graph are unique ($P(4,1)$ can be obtained from the Cartesian product of K_2 with itself). The natural and very interesting question that arises is: *Are* K_1, K_2 *and* P, *taken with the cartesian product operation, sufficient to generate all distance degree regular graphs that are optimal in this property?* This question is still open.

REFERENCES

1. A. El-Amawy and S. Latifi, Properties and performance of folded hypercubes, *IEEE Transactions on Parallel and Distributed Systems* **2** (1), 31–42 (1991).
2. F.P. Preparata and J. Vuillemin, The cube-connected cycles, A versatile network for parallel computation, *Communication of the ACM*, 30–39 (May 1981).
3. A.S. Youssef and B. Narhari, The Banyan hypercube networks, *IEEE Transactions on Parallel and Distributed Systems* **1** (2), 300–309 (1990).
4. A. El-Amawy and S. Latifi, Bridged hypercube networks, *Journal of Parallel and Distributed Computing* **10**, 90–95 (1991).
5. J.R. Goodman and C.H. Sequin, Hypertree: A multiprocessor interconnection topology, *IEEE Transactions on Computers* **C-30** (12), 923–933 (1981).
6. M.R. Samatham and D.K. Pradhan, The De Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI, *IEEE Transactions on Computers* **38** (4), 567–580 (1989).
7. H.S. Stone, Parallel processing with the perfect shuffle, *IEEE Transactions on Computers* **C-20** (2), 153–161 (1971).
8. C.L. Wu and T. Feng, The universality of the shuffle exchange, *IEEE Transactions on Computers* **C-30** (5), 324–332 (1981).
9. A.A. Nanavati and S.S. Iyengar, On optimal average message passing density in Moore graphs, *Appl. Math. Lett.* **7** (5), 67–70 (1994).
10. N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice-Hall, New Jersey, (1974).
11. F. Harary, *Graph Theory*, p. 22, Addison-Wesley, (1972).
12. G. Chartrand and R.J. Wilson, The Petersen graph, In *Graphs and Applications, Proc. of the First Colorado Symposium on Graph Theory*, (Edited by F. Harary and J.S. Maybee), pp. 69–100, John Wiley & Sons, New York, (1982).
13. A.A. Nanavati and S.S. Iyengar, On optimal average message passing density in generalized Petersen graphs, *Discrete Applied Mathematics* (submitted).