Networks of Machines for Distributed Recursive Computations

Alain J. Martin
and
Jan L. A. van de Snepscheut

Computer Science
California Institute of Technology

5147:TR:84

NETWORKS OF MACHINES FOR DISTRIBUTED RECURSIVE COMPUTATIONS

Alain J. Martin
and
Jan L. A. van de Snepscheut

Computer Science
California Institute of Technology

5147:TR:84

The research described in this paper was sponsored by
the Defense Advanced Research Projects Agency, ARPA Order No. 3771,
and monitored by the Office of Naval Research
under contract number N00014-79-C-0597

Networks of Machines for Distributed Recursive Computations

Alain J. Martin & Jan L.A. van de Snepscheut

Computer Science
California Institute of Technology
Pasadena, CA 91125

July 1984

Abstract

Distributed computations may be viewed as a set of communicating processes. If such a computation is to be executed by a multi-processor system, the processes have to be distributed over the processors and the communications have to be distributed over a network.

This leads to the questions of load balancing and message routing. In this paper we consider distributed recursive computations and we propose a class of processor networks that admits a homogeneous distribution of processes and trivial routing. Furthermore, we identify a subclass that admits a planar embedding of the network.

1. Introduction

Distributed computations may be viewed as a set of communicating processes. If such a computation is to be executed by a multi-processor system, the processes have to be distributed over the processors and the communications have to be distributed over a network.

In [1], the problem of implementing a distributed computation on a network of machines is viewed as mapping a dynamic "computation graph" on a fixed "implementation graph". The vertices and edges of the computation graph are called "nodes" and "channels", respectively. The vertices and edges of the implementation graph are called "cells" and "links", respectively. Nodes are mapped on cells and channels on links. In general, the number of nodes exceeds the number of cells, and therefore in a mapping a cell in general contains several nodes. Since each channel is mapped on a link, as opposed to a chain of links, we need not worry about message routing. Distributing the nodes homogeneously over the cells, however, is a point of concern.

Due to the importance of recursive computations, we are interested in implementation graphs well suited for mapping binary trees. A first criterion is that the implementation graph admit a mapping such that if the computation tree is a complete binary tree, the number of nodes accommodated per cell differs by at most one for any two cells. In many recursive computations the storage and time requirements differ between nodes of distinct heights and are the same for nodes of equal heights. A second criterion is therefore, that the implementation graph admit a mapping of a complete binary tree such that, for each height, the number of nodes of that height accommodated per cell differs by at most one for any two cells. A mapping that satisfies both criteria is called homogeneous.

2. An implementation graph admitting a homogeneous mapping

In [2], a class of implementation graphs admitting a homogeneous mapping was proposed. One of the authors regrets that these graphs have become known as Sneptrees.

Although channels and links admit communications in two directions, for the purpose of the discussion, we describe the computation tree and Sneptree as being directed. In the computation tree the channels are

directed towards the leaves. The Sneptree and the mapping will be constructed such that each channel is mapped onto a link pointing in the same direction.

A Sneptree is constructed as follows. Consider a complete, directed, binary tree. (The root and leaves of this tree will be called root and leaves of the Sneptree as well.) Let us add two incoming edges to the root, one incoming edge to each interior node, one incoming edge and two outgoing edges to each leaf. We observe that we have added the same number of incoming edges as of outgoing edges. Hence by connecting each dangling outgoing edge to a dangling incoming edge, a graph is obtained in which each vertex has two incoming and two outgoing edges. The graph obtained is a Sneptree.

The mapping in which the root of the computation tree is mapped on the root of the Sneptree, and the left and right successor of a node are mapped on the left and right successor, respectively, of the cell containing that node, is a homogeneous mapping.

Let the computation tree be a complete binary tree. Let the leaves of the Sneptree have height $M - 1$. The Sneptree, therefore, has $2^M - 1$ cells. A cell at height m, $0 \leq m < M$, has two incoming links, one from a cell at height $m - 1$ and one from a cell at height $M - 1$. (The number $m - 1$ is to be read as $M - 1$ if $m = 0$, but we shall not express this in our formulae.) We define $k_{m,n}$ and $s_{m,n}$ such that each cell at height m, $0 \leq m < M$, accommodates $k_{m,n}$ nodes of height n, $n \geq 0$, and $s_{m,n}$ nodes of height n or less. This definition corresponds to

$$
\begin{aligned}
& k_{0,0} = 1 \\
& k_{m,0} = 0 && \text{for all } m \quad : 0 < m < M \\
& k_{m,n} = k_{m-1,n-1} + k_{M-1,n-1} && \text{for all } m,n : 0 \leq m < M \wedge n > 0 \\
& s_{m,n} = (j: 0 \leq j \leq n: k_{m,j}) && \text{for all } m,n : 0 \leq m < M \wedge n \geq 0
\end{aligned}
$$

The homogeneity of the mapping is expressed by the property that for each n, $n \geq 0$, numbers i_n, $0 \leq i_n < M$, and c_n, $c_n \geq 0$, exist such that for all m, $0 \leq m < M$,

$$
\begin{aligned}
k_{m,n} &= c_n && \text{if } m \neq i_n \\
k_{m,n} &= c_n + 1 && \text{if } m = i_n \\
s_{m,n} &= 2.c_n + 1 && \text{if } m \leq i_n \\
s_{m,n} &= 2.c_n && \text{if } m > i_n
\end{aligned}
$$

This property is easily proved by mathematical induction on n and by using

$$
\begin{aligned}
i_n &= n \ \underline{\text{mod}} \ M \\
c_0 &= 0 \\
c_{n+1} &= 2.c_n && \text{if } n \ \underline{\text{mod}} \ M \neq M - 1 \\
c_{n+1} &= 2.c_n + 1 && \text{if } n \ \underline{\text{mod}} \ M = M - 1.
\end{aligned}
$$

3. Two instances of Sneptrees

It is quite frequent that a particular computation gives rise to a computation tree in which only right edges are present (a so-called right string) or only left edges are present (left string). If the Sneptree contains two edge-disjoint Hamiltonian circuits, the same mapping of binary trees as mentioned above is homogeneous for right and left strings. This is easily obtained by labelling all edges of one Hamiltonian circuit right edges and all edges of the other Hamiltonian circuit left edges.

For obvious lay-out reasons, we are also interested in planar Sneptrees, in particular for possible VLSI implementations.

We propose a method for constructing a Sneptree that is planar and admits two edge-disjoint Hamiltonian circuits. (The construction method is also an existence proof of such Sneptrees.) The method is by induction on the height of the Sneptree.

The Sneptree S_1 of height 1, shown in figure 1,

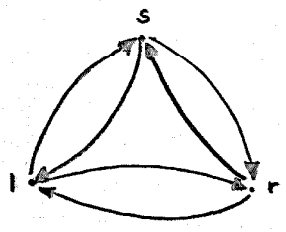


figure 1: Construction of S_1

is obviously planar, and has two edge-disjoint Hamiltonian circuits <s,1,r,s> and <s,r,1,s>. Assume that a Sneptree S_n of height n exists, that satisfies the Property P: Let s,1,r be the root, leftmost leaf, and rightmost leaf of S_n respectively. Then S_n has two edge-disjoint Hamiltonian circuits H1 = <s,D,1,r,s> and H2 = <s,E,r,1,s> where D and E denote two paths.

(Observe that S_1 satisfies P.)

Let R_n be the graph obtained from S_n by removing the directed edges (1,r), (r,s), (r,1), and (1,s). Assume that R_n satisfies the Property Q: R_n admits a planar representation in which s,1,r are "on a perimeter", i.e. can be joined in a cycle such that all edges of R_n are inside the cycle.

(Observe that R_1 obtained from S_1 satisfies Q.)

From two instances R′ and R″ of R_n we construct S_{n+1} as shown in fig.2.

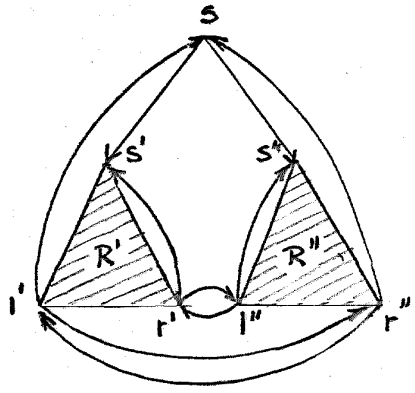


figure 2: Construction of S_{n+1}

In the graph of fig. 2, we identify 1′ with 1, and r" with r. S_{n+1} satisfies P for:

H1 = <s,s",D",1",r′,s′,D′,1,r,s>
H2 = <s,s′,E′,r′,1",s",E",r,1,s>.

The subgraph joining s to s′,1′,r′,s",1",r" in the graph of fig.2 is planar, and since R′ and R" satisfy Q, S_{n+1} is planar. Furthermore, R_{n+1} obviously satisfies Q.

Another interesting class T_n of Sneptrees can be constructed in a similar way. T_1 is identical to S_1. Assume T_n exists that fulfills the Property P′: Let s,1,r be the root, leftmost leaf, and rightmost leaf of T_n respectively. Then T_n has two edge-disjoint Hamiltonian circuits H1 and H2 of the form:
H1 = <s,D,r,s>, H2 = <s,E,1,s>, where D and E denote two paths.

(Again T_1 satisfies P′.)

Let U_n be the graph obtained from T_n by removing the edges (1,s) and (r,s). From two instances U′ and U" of U_n we construct T_{n+1} as follows:

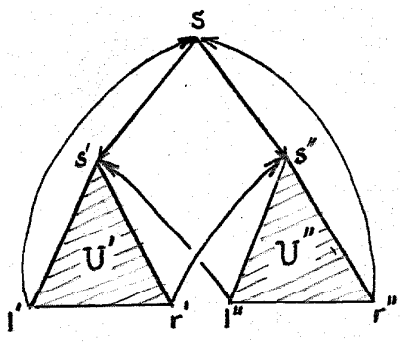


figure 3: Construction of T_{n+1}

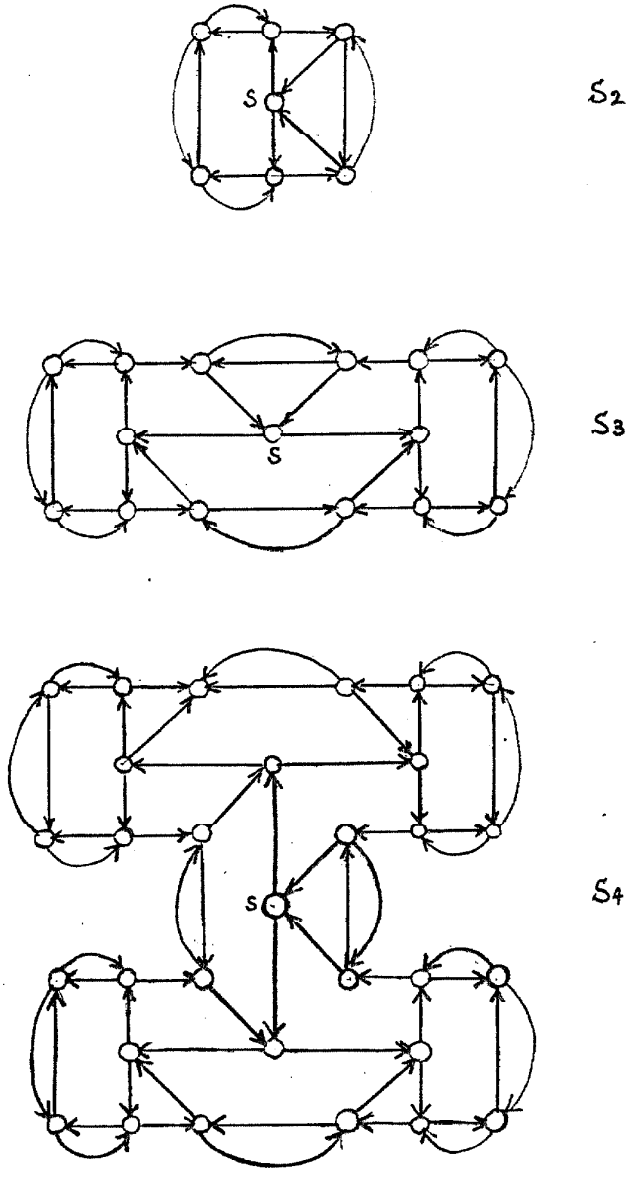
In the graph of fig. 3, we identify 1 with 1′ and r with r". T_{n+1} has two edge-disjoint Hamiltonian circuits satisfying P′, namely:

H1 = <s,s′,D′,r′,s",D",r,s>
H2 = <s,s",E",1",s′,E′,1,s>.

But for $n > 1$, U_n does not satisfy Q. Hence for $n > 2$, T_n has no planar representation.

4. Examples of planar embedding

The following figure shows representations of S_2, S_3, S_4 based on the H-tree representation of binary trees:



References

[1] A.J. Martin: "A distributed implementation method for parallel programming," Proceedings IFIP-Congress, October 1980.

[2] J.L.A. van de Snepscheut: "Mapping a dynamic tree on a fixed graph," JAN 78a, February 1981.