THE CONTRACT NET: A FORMALISM FOR THE CONTROL OF DISTRIBUTED PROBLEM SOLVING*

Reid G. Smith
Heuristic Programming Project,
Department of Computer Science,
Stanford University, Stanford, California, 94305.

Distributed processing offers the potential for high speed, reliable computation, together with a means to handle applications that have a natural spatial distribution. In this paper, distributed processing is defined as processing that is characterized by physical decomposition of the processor into relatively Independent processor nodes. Recent advances in LSI technology, expected to result in single silicon wafers with at least 10^6 active elements by 1981 [Noyce, 1976], indicate that it is reasonable to contemplate designs which incorporate large networks of single chip processor nodes.

In this paper we examine the control of problem solving in such an environment, where most information is local to a node, and relatively little information is shared by the complete network. Individual nodes correspond to "experts" which cooperate to complete a top level task (analogous to the "beings" of [Lenat, 1975]). The distributed processor is thus to be composed of a network of "loosely-coupled", asynchronous nodes, with distributed executive control, a flexible interconnection mechanism, and a minimum of centralized, shared resources. This puts the emphasis on "coarse grain" parallelism, in which individual nodes are primarily involved with computation (large kernel tasks), pausing only occasionally to communicate with other nodes.

The CONTRACT NET represents a formalism in which to express the control of problem solving in a distributed processor architecture. Individual tasks are dealt with as <u>contracts</u>. A node that generates a task broadcasts Its existence to the other nodes in the net as a contract announcement, and acts as the <u>contract manager</u> for the duration of that contract. <u>Bids</u> are then received from potential <u>contractors</u>, which are simply other nodes in the net. An <u>award</u> is made to one node which assumes responsibility for the execution of that contract. <u>Subcontracts</u> may be- let in turn as warranted by <u>task size</u> or a requirement for special expertise or data not in the possession of the contractor. When a contract has been executed, a <u>report</u> is transmitted to the contract manager.

Contracts may be announced via general broadcast, limited broadcast, or point-to-point communications mechanisms, depending on information about relevant contractors available to the contract manager. If, for example, a manager has knowledge about the location of particular data, then its contract announcement will be directed to the node(s) believed to possess that data, so that the complete network is not needlessly involved.

Contracting effectively distributes control throughout the network, thus allowing for flexibility and reliability. Decisions about what to do next are made as a result of relatively local considerations, between pairs of processors, although the nature of the announcement-bid-award sequence maintains an adequate global context; that is, the decision to bid on a particular contract is made on the basis of local knowledge (the task being processed in the node contemplating a bid), and global knowledge (other current contract announcements). The formalism also incorporates two way links between nodes that share responsibility for tasks (managers and contractors). The failure of a contractor is therefore not fatal, since a manager can re-announce a contract and recover from the failure.

A node in the CONTRACT NET is composed of a contract.processor, management processor, communications Interface, and local memory.The contract processor is responsible for the applications-related computation of the node. The management processor is responsible for network communications, contract management, bidding, and the management of the node itself. Individual nodes are not designated a priori as contract managers or contractors. Any node can take on either role, and during the course of problem solving, a particular node normally takes on both roles simultaneously for different contracts.

A contract is represented as a record structure with the following fields: <u>NAME</u> - the name of the contract, <u>NODE</u> - the name of another processor node associated with the contract. <u>PRIORITY</u> - a description of the "importance" of the contract. <u>TASK</u> - a description of the task to be performed,<u>RESULTS</u> - a description of the results obtained/and <u>SUBCONTRACTS</u> - a pointer to the list of subcontracts that have been generated from the contract.

Contracts are divided into two classes in a node: those for which the node acts as contractor, and those for which it acts as manager. The node field of a contract is filled accordingly - with the name of the contract manager in the first case, and with the name of the contractor in the second case. Subcontracts waiting for service are held at the node that generated them, with an empty node field.

The <u>priority description</u> is used by a management processor to establish a partial order over contracts to be announced, and by potential contractors to order contracts for the purpose of bidding. Similar descriptions are also used to order contractors for the purpose of awards. The concept of priority thus must be generalized over simple integer descriptions to include such (layered) descriptions of potentially arbitrary complexity, which include both applications-related and architecture-related information.

A task description also contains two types of information: the local context in which the task is to be executed, and the applications software required to execute it. This information is passed when a contract is awarded. Depending on the task, the required global context may be passed with the award, or further contracts may be let to obtain it. Software passed to a node for execution of a particular contract is retained for future use, and its presence has a favorable effect on the future bids of that node.

A SAIL simulation has been constructed to test the formalism. It accepts simple applications programs. and maps them onto a simulated distributed processor with a variable number of nodes. The simulation Is being used to determine the costs associated with the formalism, in terms of both processor and communications overhead, and the decrease in computation time that can be expected for various applications. Distributed heuristic search is presently being examined in this way, and alternatives in distributed deduction will be examined in the near future.

This work is supported by the Advanced Research Projects Agency under contract DAHC 15-73-C-0435. Computer facilities are provided by the SUMEX-AIM facility under National Institutes of Health grant RR-00785. The author is supported by the Research and Development Branch of the Department of National Defence of Canada. E. A, Felgenbaum, B. G. Buchanan. G. Wiederhold, K. G. Knutsen, and E, J. Gilbert provided a number of useful suggestions. F. K. Buelow **was** a valuable source of LSI technology data.

References

D. B. Lenat. "Beings: Knowledge As Interacting Experts," IJCAIU Proceedings, Tbilisi, USSR, September 1975, pp. 126-133.

R. N. Noyce, "From Relays to MPU's," Computer, Vol. 9, No. 12, December 1976, pp.26-29.