

A FRAMEWORK FOR DISTRIBUTED PROBLEM SOLVING

Reid G. Smith

Defence Research Establishment Atlantic
Box 1012
Dartmouth, Nova Scotia, Canada, B2Y 3Z7

Abstract

The contract net framework for distributed problem solving is discussed. Task distribution is viewed as a mutual selection process, a discussion carried on between a node with a task to be executed and a group of nodes that may be able to execute the task. This leads to the use of a control formalism based on a contract metaphor, in which task distribution corresponds to contract negotiation.

The three primary components of the framework are described: communication, control, and knowledge organization. The use of the framework is illustrated for a distributed sensing system. We then discuss suitable applications and limitations of the framework.

1 Distributed Problem Solving

Distributed Problem Solving is the cooperative solution of problems by a decentralized and loosely coupled collection of knowledge-sources (KSs), each of which may reside in its own distinct processor.¹ Decentralized means that both control and data are logically, and sometimes geographically, distributed--there is neither global control nor global data storage. The KSs cooperate to solve problems by sharing tasks and/or results. Loosely coupled means that individual KSs spend the great percentage of their time in computation as opposed to communication. Such problem solvers offer advantages of speed, reliability, extensibility, the ability to handle applications with a natural spatial distribution, and the ability to tolerate errorful data and knowledge. In addition, as a result of their modularity, they offer conceptual clarity and simplicity of design.

2 The Contract Net Framework

The central result of the research reported here is the contract net framework for problem solving in a distributed processing environment. The framework has three major conceptual components: a control component that specifies the possible modes of interaction between processor nodes; a knowledge organization component that specifies how knowledge is organized in individual nodes and distributed throughout the collection of nodes; and a communication component that provides the basis for node interaction.

2.1 Communication And Control

2.1.1 Connection And Contract Negotiation

In the design of the communication and control components of the framework, our emphasis to date has been placed on facilitation of task-sharing as a means of cooperation for problem solving. The computational load for execution of subtasks of the overall problem is distributed among the nodes. To enable this distribution, there must be a means whereby nodes with tasks to be executed can find other idle nodes with KSs capable of executing those tasks. We call this the connection problem. (In centralized problem solvers it is called the invocation problem; that is, which KS to invoke at any given time for the execution of a task.) In a distributed

¹ This work was supported in part by the Advanced Research Projects Agency under contract MDA 903-77-C-0322, and the National Science Foundation under contract MCS 77-02712. It was carried out on the SUMEX-AIM Computer Facility, supported by the National Institutes of Health under grant RR-00785. This paper is based on work done as part of a Ph.D. dissertation at Stanford University. The contributions of Randall Davis, Bruce Buchanan, and the rest of the staff and students of the Heuristic Programming Project are gratefully acknowledged.

problem solver, both sets of nodes can proceed simultaneously, engaging each other in a process of negotiation to solve the connection problem. Task distribution can thus be considered as a process of contract negotiation, or mutual selection based on a two-way transfer of information. This process gives rise to the name--the contract net framework [Smith, 1978b].

The collection of nodes is referred to as a contract net and the execution of a task is dealt with as a contract between two nodes. Thus, each node in the net takes on one of two roles related to the execution of an individual task: manager or contractor. A manager is responsible for monitoring the execution of a task and processing the results of its execution. A contractor is responsible for the actual execution of the task. Individual nodes are not designated a priori as managers or contractors. These are only roles, and any node can dynamically take on either role. During the course of problem solving, a particular node normally takes on both roles (perhaps even simultaneously for different contracts).

The normal method of negotiating a contract is for a node that generates a task to advertise the existence of that task to other nodes in the net with a task-announcement message. It then acts as the manager of that task for its duration. Many such announcements are made over the course of time as tasks are generated for execution.

Nodes in the net have been listening to the task announcements and have been evaluating their own level of interest in each task with respect to their specialized hardware and software resources. When a task of sufficient interest is found, a node submits a bid. A bid message indicates the capabilities of the bidder that are relevant to execution of the announced task. A manager may receive several such bids in response to a single task announcement; based on the information in the bids, it selects one (or several) node(s) for execution of the task. The selection is communicated to the successful bidder(s) through an award message. These selected nodes thus become contractors for that task.¹

¹ This is a simplified version of the actual process (and does not work, for example, in the case of a task that cannot be executed due to insufficient data). We have also ignored the use of focused addressing and more specialized interactions like directed contracts. See [Smith, 1978b] for the

2.1.2 The Contract Net Protocol

The use of communication protocols in networks of resource-sharing computers, such as the ARPAnet, is by now quite familiar. The primary function of these protocols is to effect reliable and efficient communication between computers. Such low-level protocols are, however, only a start--a prerequisite for distributed problem solving. They enable bit streams to be passed between nodes but do not consider the semantics of the information being passed. We require a high-level protocol, which we call a problem-solving protocol (PSP), to assign problem-solving interpretations to the bit streams--to control the processes used to solve problems as opposed to the processes used to effect communication. This enables the applications programmer to focus on what the nodes must say to each other, as opposed to how to say it.

The contract net protocol is the PSP that governs all problem-solving interactions between nodes in a contract net. Task-dependent information is placed in typed slots in the messages of the protocol and is encoded in a common internode language. Task-independent information is encoded directly in the protocol; that is, in the specification of message types and their associated slots.

A task announcement message, for example, has four slots for task-dependent information. The eligibility specification is a list of criteria that a node must meet to be eligible to submit a bid. It enables a node receiving the message to decide whether or not it is able to execute the task. The task abstraction is a brief description of the task to be executed. It enables a node to rank the announced task relative to other announced tasks. The bid specification is a description of the expected form of a bid. It enables a node to include in a bid only the information about its capabilities that are relevant to the announced task. Finally, the expiration time is a specification of the time period during which the announcement is valid.

The common internode language provides the primitive elements for encoding task-dependent information in the task abstraction, eligibility specification, bid specification, and so on. The implemented language enables

specification of the full set of message types in the protocol and a complete discussion of the processing associated with each of the message types.

2.2 Example

The managers for this signal task attempt to find a set of sensor nodes such that the set has an adequate spatial distribution about the surrounding area and has an adequate distribution of sensor types. Sensor nodes, on the other hand, attempt to find managers for the signal task that are closest to them.

The potential contractors listen to the task announcements from the various managers. They respond to the nearest manager with a bid that contains the information specified in the task announcement. The managers use this information to select a set of bidders and then award signal contracts. The award messages specify the sensors that each contractor is to use to provide raw data to its manager.

To: * <Managers make announcements
From: 25 of this form.>
Type: TASK ANNOUNCEMENT
Contract: 22-3-1
Message:

2.3.1 Partitioning

It is common in AI problem solvers to partition expertise into domain-specific knowledge-sources (KSs), each of which is expert in a particular part of the overall problem. KSs are typically formed empirically, based on examination of different types of knowledge that can be brought to bear on a particular problem. In a speech-understanding problem, for example, knowledge is available from the speech signal itself, from the syntax of the utterances, and from the semantics of the task domain [Erman, 1975]. The decisions about which KSs are to be formed is often made in concert with the formation of a data hierarchy for a problem. KSs are typically chosen to handle data at one level of abstraction or to bridge two levels (see, for example [Erman, 1975] and [Nil, 1978]).

In a distributed processor there is an additional consideration: The KSs themselves will likely be distributed to different nodes, and their interactions will therefore often be more expensive than in a uniprocessor. As a result, the kernel size of the KSs must be chosen carefully, based on the characteristics of the distributed processor. KSs that are too small will result in a large amount of communication between nodes, thus reducing the speedup that can be achieved via a distributed approach. Matching of kernel size to distributed processor characteristics must at present be done via trial and error by the applications programmer.

2.3.2 Indexing

In the contract net framework, the main issue in indexing of knowledge is connecting nodes with tasks to be executed to nodes with KSs that are appropriate to execute those tasks. Two major types of knowledge are recognized for indexing: task-centered knowledge and knowledge-source-centered knowledge (KS-centered knowledge). Task-centered knowledge is useful for finding nodes capable of executing tasks and KS-centered knowledge is useful for finding tasks to be executed by a node. The contract net appears to be the first use of both kinds of knowledge at the same time.

In the DSS, for example, the task-centered knowledge used by the manager for the signal task specifies that contracts should be awarded so that the area surrounding the manager is covered with an adequate distribution of sensor

types. The KS-centered knowledge used by a sensor node specifies that a signal task should be selected from the closest manager that offers such a task.

2.3.3 Distribution Of Knowledge

Distribution also has two aspects: static distribution, how knowledge is initially loaded into the nodes, and dynamic distribution, how knowledge is transferred between nodes as work on the overall problem proceeds.

The criteria for a good static distribution of knowledge are minimization of message traffic and avoidance of critical function nodes that could reduce processing speed and create reliability problems. In the DSS, for example, KSs with expertise at a particular level of the data hierarchy were placed in different nodes. This enabled nodes to carry on simultaneously with computation specific to those levels of the data hierarchy that concerned them (without the need for frequent interactions with nodes operating at different levels of the hierarchy).

In the contract net framework, dynamic distribution of knowledge can be effected in three ways. First, a node can transmit a request directly to another node for the transfer of the required knowledge. The response is the knowledge requested (e.g., the code for a procedure). Second, a node can broadcast a task announcement in which the task is a transfer of knowledge. A bid on the task indicates that another node has the knowledge and is willing to transmit it. Finally, a node can note in its bid on a task that it requires particular knowledge in order to execute the task. The manager can then send the required knowledge in the contract award if the bid is accepted. Each of these interactions is simplified by the use of a common internode language.

Dynamic distribution enables effective use of available computational resources: A node that is standing idle because it lacks information required to execute a previously announced task can acquire that information as indicated above. Dynamic knowledge distribution also facilitates the addition of a new node to an existing net; the node can dynamically acquire the procedures and data necessary to allow it to participate in the operation of the net. This is especially useful in the DSS application.

3 Discussion

There are several reasons for adopting a distributed approach to problem solving. These include speed, reliability, extensibility, and the ability to handle applications that have a natural spatial distribution. The design of the contract net framework has taken into account each of these considerations. We relate here design choices made in the framework to their underlying motivations.

In order to achieve high speed we wish to avoid bottlenecks. Such bottlenecks can arise in two primary ways: by concentrating disproportionate amounts of computation or communication at central resources, and by saturating available communications channels so that nodes must remain idle while messages are transmitted.

To avoid bottlenecks we facilitate the distribution of control and data. This is a major motivation for the use of local contract negotiation to achieve connections and task distribution and to provide a means for dynamic distribution of procedures and other data. System concurrency is also enhanced because both managers and contractors simultaneously seek each other out, finally achieving connections by mutual selection.

To avoid communications channel saturation we attempt to maintain loose-coupling. The framework is well-suited to loosely coupled systems in three respects. First, it provides a very general form of guidance in determining appropriate partitioning of problems: the notion of tasks executed under contracts is appropriate for a kernel size larger than that typically used in parallel systems. Second, the framework attempts to be efficient with respect to its use of communications channels by reducing the number and length of messages. The information in task announcements, for instance, helps minimize the amount of channel capacity consumed by communications overhead. Extraneous traffic is eliminated by the eligibility specifications of the task announcements and the bid messages are reduced in length because of the bid specifications. Finally, the framework enables dynamic distribution of procedures and other data. Thus the option exists to distribute information only when required:

Reliability is also enhanced by the distribution of control and data, together with shared responsibility for tasks (by managers and contractors). The failure of a contractor,

for example, is not fatal, since its manager can re-announce the appropriate contract and recover from the failure. This strategy allows the system to recover from any node failure except that of the node that holds the original top-level problem.¹

Extensibility is facilitated by the use of a common internode language and dynamic distribution of knowledge.

The ability to handle applications with a natural spatial distribution is facilitated by the use of local contract negotiation to achieve connection and task distribution.

4 Suitable Applications

The framework is particularly well-matched to problems that use a hierarchy of tasks and levels of data abstraction. Heuristic search problems are examples of the former, and applications that deal with sensed data (e.g., audio or video signals) are examples of the latter.

The manager-contractor structure provides a natural way to effect hierarchical control and the managers at each level in the hierarchy are an appropriate place for data integration and abstraction. The contract links between nodes assist in coordination of KSs (i.e., a manager can coordinate the actions of contractors working on related subtasks).

It also follows that the framework is primarily applicable to domains where the subtasks are large (in the loose-coupling sense) and where it is worthwhile to expend a potentially nontrivial amount of computation and communication to invoke the best KSs for each subtask. (The same basic mechanism can, however, be used simply as a means of task distribution with distributed control and shared responsibility for tasks to maintain reliability and avoid bottlenecks.)

The framework has also been designed to

¹ At the top level, contracting can distribute control almost completely, hence removing the bottlenecks that centralized controllers create. There still remains, however, the reliability problem inherent in having only a single node responsible for the top-level problem. Since this cannot be handled directly by the manager-contractor links, standard sorts of redundancy are required.

provide a more powerful mechanism for transfer of control than is available in current problem-solving systems (see [Smith, 1978b] for a complete discussion). The announcement-bid-award sequence of contract negotiation enables more information and more complex information to be transferred in both directions (between caller and respondent) before KS-invocation occurs. In addition, information about the complete collection of candidate KSs is available before a final selection is made. The computation devoted to the selection process, based on the information transfer noted above, is more extensive and more complex than that used in traditional approaches, and is local in the sense that selection is associated with and specific to an individual KS (rather than embodied in, say, a global evaluation function). As a result, the framework is most useful when the specific KS to be invoked at any time is not known a priori and when specific expertise is required.

5 Limitations And Extensions

We have proposed a framework that offers some ideas about what information is useful for distributed problem solving and how that information can be organized. There is still a considerable problem involved in instantiating the framework in the context of a specific task domain. The contract net protocol provides a site for embedding particular types of task-dependent information (e.g. an eligibility specification), but does not specify, for a particular problem, the content, nor how to instantiate it in a particular domain. We require more experience with the framework to better understand its utility as a mechanism for helping a user structure and understand distributed problems.

We have emphasized task-sharing as a means of internode cooperation and have attempted to provide some structured mechanisms for the communication required to effect this mode of cooperation. We have, however, not yet adequately studied result-sharing as a means of cooperation; that is, nodes assisting each other through sharing of partial results, based on somewhat different perspectives on the overall problem. Different perspectives arise because the nodes use different KSs (e.g., syntax vs acoustics in the case of a speech understanding system) or different data (e.g., data that is sensed at different locations in the case of a DSS). This type of cooperation appears to be of use in dealing with problems

where errorful data or knowledge lead to conflicting views of the problem at individual nodes (see [Lesser, 1978] for a preliminary discussion). It is our intention to examine the structure of communication for this mode of cooperation with a view to extending the contract net framework so that a synthesis of the two approaches to distributed problem solving can be attempted.

References

[Erman, 1975]

L. D. Erman and V. R. Lesser, A Multi-level Organization For Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge. IJCAI4, 1975, pp. 483-490.

[Lesser, 1978]

V. R. Lesser, Cooperative Distributed Processing. COINS TR 78-7, Dept. of Computer and Information Science, University of Massachusetts, May 1978.

[Nii, 1978]

H. P. Nii and E. A. Feigenbaum, Rule-Based Understanding Of Signals. In D. A. Waterman and F. Hayes-Roth (Eds.), Pattern-Directed Inference Systems. New York: Academic Press, 1978. Pp. 483-501.

[Smith, 1978a]

R. G. Smith and R. Davis, Applications Of The Contract Net Framework: Distributed Sensing. Proceedings of the ARPA Distributed Sensor Net Symposium, Pittsburgh, PA, December 1978, pp. 12-20.

[Smith, 1978b]

R. G. Smith, A Framework For Problem Solving In A Distributed Processing Environment. Ph.D. Dissertation, STAN-CS-78-700 (HPP-78-28) Dept. of Computer Science, Stanford University, December 1978.