Report 78-07
Stanford -- KSL

Scientific DataLink

Distributed Problem Solving: The Contract Net Approach.
Reid G. Smith, Randall Davis,
Sep 1978

Stanford Heuristic Programming Project
Memo HPP-78-7

September 1978

Computer Science Department
Report No. STAN-CS-78-667

DISTRIBUTED PROBLEM SOLVING: THE CONTRACT NET APPROACH

by

Reid G. Smith and Randall Davis

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

DISTRIBUTED PROBLEM SOLVING: THE CONTRACT NET APPROACH

Reid G. Smith and Randall Davis

STAN-CS-78-667
Heuristic Programming Project Memo 78-7

ABSTRACT

We describe a problem solver based on a group of processor nodes which cooperate to solve problems. In a departure from earlier systems, we view task distribution as an interactive 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.

We also consider the kinds of knowledge that are used in such a problem solver, the way that the knowledge is indexed within an individual node, and distributed among the group of nodes. We suggest two primary methods of indexing the knowledge (referred to as "task-centered" and "knowledge-source centered"), and show how both methods can be useful.

We illustrate the kind of information that must be passed between nodes in the distributed processor in order to carry out task and data distribution. We suggest that a common internode language is required, and that task-specific "expertise" required by a processor node can be obtained by internode transfer of procedures and data.

We consider the operation of a distributed sensor net as an instantiation of the issues we raise.

Finally, the approach presented here is compared with those taken by the designers of earlier systems, such as PLANNER, HEARSAY-II, and PUP6.

This is a preprint of a paper to appear at the Second National Conference of the Canadian Society for Computational Studies of Intelligence, Toronto, Canada, July 1978.

KEY WORDS

PROBLEM SOLVING, DISTRIBUTED PROCESSING, CONTRACT NET, ARTIFICIAL INTELLIGENCE.

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2494, Contract No. MDA 903-77-C-0322.

DISTRIBUTED PROBLEM SOLVING: THE CONTRACT NET APPROACH

Reid G. Smith and Randall Davis[1]

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

May 1978

Abstract

We describe a problem solver based on a group of processor nodes which cooperate to solve problems. In a departure from earlier systems, we view task distribution as an interactive 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.

We also consider the kinds of knowledge that are used in such a problem solver, the way that the knowledge is indexed within an individual node, and distributed among the group of nodes. We suggest two primary methods of indexing the knowledge (referred to as "task-centered" and "knowledge-source centered"), and show how both methods can be useful.

We illustrate the kind of information that must be passed between nodes in the distributed processor in order to carry out task and data distribution. We suggest that a common internode language is required, and that task-specific "expertise" required by a processor node can be obtained by internode transfer of procedures and data.

We consider the operation of a distributed sensor net as an instantiation of the issues we raise.

Finally, the approach presented here is compared with those taken by the designers of earlier systems, such as PLANNER, HEARSAY-II, and PUP6.

This is a preprint of a paper to appear at the Second National Conference of the Canadian Society for Computational Studies of Intelligence, Toronto, Canada, July 1978.

---

[1] This work has been 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 has been carried out on the SUMEX-AIM Computer Facility, supported by the National Institutes of Health under grant RR-00785. Reid Smith is supported by the Department of National Defence of Canada, and Randall Davis is supported by the National Science Foundation and a Chaim Weizmann Postdoctoral Grant for Scientific Research.

1 Introduction

The ongoing revolution in LSI technology is drastically reducing the cost of computer components, making multiple processor architectures economically viable. These architectures have the potential to provide several computational advantages over uniprocessor architectures, including speed, reliability, and efficient matching of available processing power to problem complexity [Baer, 1973]. This has led to a search for problem solving methods which can exploit the new technology. In this paper we present one approach to problem solving in such architectures.

We propose a model of a ***distributed problem solver*** which consists of a collection of processors connected with communications and control mechanisms that enable them to operate concurrently, and enable them to cooperate in solving complex problems. We use the term "distributed" rather than "parallel" to emphasize that the individual processors are ***loosely-coupled***; that is, the time a processor node spends in communication is small with respect to the time it spends in computation.

Loosely-coupled systems are desirable for a number of reasons. First, such systems are highly modular, and hence offer considerable conceptual clarity and simplicity in their organization. Second, and equally important from our perspective, systems designed to be loosely-coupled require less communication by an individual node. This is an important practical consideration because a major problem that arises in the design of multiple processor architectures is interconnection of the nodes [Anderson, 1975]. Complete interconnection (so that a node can communicate directly over a private channel to every other node) is extremely expensive because it entails a number of channels proportional to the square of the number of nodes. One way to reduce this expense is to employ a single broadcast communications channel which is shared by all nodes. Unfortunately, such a channel can be a major source of contention and delay when the number of processor nodes is large. The communications medium connecting the nodes is thus a valuable (and limited) resource that must be conserved if a large number of processor nodes is to function together effectively. It is thus desirable to reduce the amount of message traffic, and designing the system so it is loosely-coupled is one way to accomplish this goal. Loose-coupling can in turn be effected by careful partitioning of the top-level problem to insure that individual processor nodes work on tasks that are relatively independent of each other, and that require processing times which are large with respect to the time required for internode communication.[2]

1.1 A Human Model

The operation of a problem solver working in a distributed processor architecture may be likened to the operation of a group of human experts experienced at working together to complete a large task.[3] In such a situation we might see each expert spending most of his

---

[2] Partitioning of this kind is, of course, a well-known problem solving strategy, often referred to as "divide and conquer".

[3] The group of experts model has also been used as a starting point by [Lenat, 1975] and [Hewitt, 1977a], but has resulted in approaches with different characteristics than that considered in this paper. We compare the different approaches in Section 5.

time working alone on various subtasks that have been partitioned from the main task, pausing occasionally to interact with other members of the group in specific, well-defined ways. When he encounters a subtask too large to handle alone, he further partitions it into manageable (sub)subtasks and makes them known to the group. Similarly, if he encounters a subtask for which he has no expertise, he attempts to pass it on to another more appropriate expert. In this case, the expert may know another expert (or several other experts) in the group who have the necessary expertise, and may notify him (them) directly. If the expert does not know anyone in particular who may be able to assist him, or if the new subtask requires no special expertise, then he can simply describe the subtask to the entire group. If some other expert chooses to carry out the subtask, then that expert will request further details from the original expert, and the two may engage in further direct communication for the duration of the subtask. The two experts will have formed their own subgroup, and similar subgroups of variable size will form and break up dynamically during the course of the work on the problem. Subgroups of this type offer two advantages. First, communication among the subgroup members does not needlessly distract the entire group of experts. Such distraction may be a major source of difficulty in large groups (see, for example [Brooks, 1975]). In addition, the subgroup members may be able to communicate with each other in a language that is more efficient for their purposes than the language in use by the group as a whole.

It is worthy of note that among the tasks which can be posed by an expert in the group are those that involve a transfer of "expertise" from one expert to another; that is, one expert may request instruction in the execution of a particular task.

In our human model, no one expert is in control of the others, (although one expert may be ultimately responsible for communicating the solution of the top-level problem to the "customer" outside the group). As a result, one of the major problems facing such a group is integration of information held by the individual members. The group members must find ways to share and build on one another's information, and find ways to examine and resolve differences in order to reach a consensus.

2 Problem Solving Protocols

We now consider the design of a problem solver that can exploit the characteristics of a distributed processor architecture. In doing this, we make a rough correspondence between human experts and individual processor nodes, but our aim is the design of an effective problem solver, not a simulation of human performance. The question is then, "What techniques will supply the requisite communications and control mechanisms?" We will see that one of the necessary mechanisms is a ***problem solving protocol*** designed to enable the individual nodes to communicate for the purpose of cooperative problem solving. It is based on the more traditional notion of ***communications protocol***.

The use of communications protocols in networks of resource-sharing computers, such as the **ARPAnet**, is by now quite familiar [Kahn, 1972]. These protocols have as their primary function reliable and efficient communication between computers. The layers of protocol in the **ARPAnet**, for example, serve to connect IMP's to IMP's (the subnet

communications devices), hosts to hosts (the processor nodes of the network), and processes executing in the various hosts to other such processes [Crocker, 1972].

Communications protocols are, however, only a start - a prerequisite for distributed problem solving. We need to build upon the work of network and communications protocol designers to focus on *what* to say in the context of distributed problem solving, as opposed to *how* to say it. In ARPAnet terms, we must move above the process-to-process protocol to add yet another layer - one concerned with the management of tasks.

2.1 Design Goals

Before presenting the specific protocol to be used throughout the remainder of this paper, we review the general design goals for a problem solving protocol.

First, we are concerned with the communication of messages between the nodes of a distributed problem solver. We must therefore insure that our protocol is sufficiently general that it allows the communication of a broad class of information, and allows interactions capable of supporting complex problem solving behavior.

Second, the protocol must be well-suited to systems that are loosely-coupled. As noted earlier, it is important to minimize communication since communications channel capacity is expensive. While careful task partitioning has the greatest potential impact on the amount of internode communication required, the problem solving protocol also plays a role. Therefore, the protocol should be efficient in terms of its use of communications resources (i.e., terse).[4]

The protocol should also foster distribution of control and data in order to insure that advantage can be taken of potential gains in speed and reliability that may be achieved through the use of multiple processors. Centralized control could create an artificial bottleneck (slowing the system down), and could make it difficult for the system to recover from failure of critical components.

Finally, the protocol should aid in maintaining the ***focus*** of the problem solver, to combat the combinatorial explosion which besets almost all AI programs. For a uniprocessor, focus involves selection at each instant in time of the most appropriate task to be executed [Hayes-Roth, 1977]. For a distributed processor, focus can be reformulated as finding the most appropriate tasks to be executed and matching them with processor nodes appropriate for their execution.

In a uniprocessor, focus of attention is generally handled by a single, global, heuristic evaluation function used to rank order all tasks in the system (see, for example [Lenat, 1976]). In a distributed processor, however, each individual processor node has its own local evaluation function. Task selection and decisions are thus based on local considerations, and this locality gives rise to the problem of inducing global coherence in the

---

[4] Current evidence [Galbraith, 1974] suggests that effective human organizations operate in an analogous manner, minimizing unnecessary communications among the members.

actions of the individual processor nodes. Since this can be a major source of difficulty, the problem solving protocol should also offer some assistance in overcoming it.

2.2 The Contract Net

These considerations lead to the notion of task distribution as an interactive process, one which entails a discussion between a node with a task to be executed and nodes that may possibly be able to execute the task. The contract net approach to distributed problem solving [Smith, 1977] uses an announcement - bid - award sequence of ***contract negotiation*** to effect this matching. We present a simplified description of the approach in this section.

A ***contract net*** is a collection of interconnected processor nodes whose interactions are governed by a problem solving protocol based on the contract metaphor. Each processor node in the net operates asynchronously and with relative autonomy. Instances of the execution of individual tasks are dealt with as ***contracts***. A node that generates a task advertises existence of that task to the other nodes in the net as a ***task announcement***, then acts as the ***manager*** of that task for its duration. In the absence of any information about the specific capabilities of the other nodes in the net, the manager is forced to issue a ***general broadcast*** to all nodes. If, however, the manager possesses some knowledge about which of the other nodes in the net are likely candidates, then it can issue a ***limited broadcast***. Finally, if the manager knows exactly which of the other nodes is appropriate, then it can issue a ***point-to-point*** announcement.[5] As work on the problem progresses, many such task announcements may be made by various managers.

The other 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 is found to be of sufficient interest, a node may submit a ***bid***. Each bid 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 select 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 assume responsibility for execution of the task, and each is called a ***contractor*** for that task.

A contract is thus an *agreement* between a node that generates a task (the manager) and a node that executes the task (the contractor). Note that establishing a contract is a process of mutual selection. Available processor nodes evaluate task announcements made by several managers until they find one of interest; the managers then evaluate the bids received from potential contractors and select one they determine to be most appropriate. Both parties to the agreement have evaluated the information supplied by the other and a mutual decision has been made.

---

[5] Restricting the set of addressees (which we call ***focused addressing***) of an announcement is typically a heuristic process, since the information upon which it is based may not be exact (e.g., it may be inferred from prior responses to announcements).

The contract negotiation process is expedited by three forms of information contained in a task announcement. An ***eligibility specification*** lists the criteria that a node must meet to be eligible to submit a bid. This specification reduces message traffic by pruning nodes whose bids would be clearly unacceptable. A ***task abstraction*** is a brief description of the task to be executed, and allows a potential contractor to evaluate its level of interest in executing this task relative to others that are available. An abstraction is used rather than a complete description in order to reduce message traffic. Finally, a ***bid specification*** details the expected form of a bid for that task. It enables a potential contractor to transmit a bid which contains only a brief specification of its capabilities that are relevant to the task (called a ***node abstraction***), rather than a complete description. This both simplifies the task of the manager in evaluating bids, and further reduces message traffic.[6]

The normal contract negotiation process may be simplified in two instances. First, a ***directed contract*** does away with the announcement and bid, and is awarded directly to a selected node. Second, a request - response sequence is used without further embellishment for tasks which amount to simple requests for information. These two simplifications serve to enhance the efficiency of the protocol.

It is important to note that individual nodes are not designated a priori as 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 (perhaps even simultaneously for different contracts).

In addition to effecting task distribution, a contract between two nodes serves to set the context for their communication. Setting up such a context facilitates their communication. A contract is also of assistance in forming subgroups of nodes. As in the human model discussed above, such subgroups can communicate among themselves without distracting the entire group. Furthermore, an established context permits the use of a specialized language for their communication. This helps to reduce message traffic.

The award message contains a ***task description***, which includes the complete specification of the task to be executed. After the task has been completed, the contractor sends a ***report*** to its manager. This message includes a ***result description***, which contains the results that have been achieved during execution of the task.

The manager may ***terminate*** contracts as necessary, and ***subcontracts*** may be let in turn as required by the size of a contract or by a requirement for special expertise or data that the contractor does not have.

Contracting distributes control throughout the network, helping to create a flexible system; that is, a number of different (potentially dynamic) approaches to problem solving can be implemented. Distributed control and two-way links between managers and contractors also enhance system reliability, in that they enable recovery from individual component failure. 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

---

[6] We discuss the encoding of this information in Section 4.3.

top-level problem.[7]

While the contract net protocol is a general problem solving protocol, it has been designed so it can be pruned to meet the specific requirements of the application at hand, and hence reduce message traffic and message processing overhead. In its simplest form, it reduces to a standard communications protocol, sending messages between specified sources and destinations. At a slightly more general level, broadcasting of tasks and results is possible, thus effecting a more implicit form of addressing. At progressively more general levels, complex bidding and award mechanisms are added. The contract net can thus be a useful approach to distributed problem solving at many different levels of complexity.

We can now consider how well the contract net protocol meets the design goals specified earlier for a problem solving protocol.

The protocol is well suited to loosely-coupled systems in two 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 grain size larger than that typically used in problem solving systems. (Section 5.1 contains further discussion of this issue.) Second, the protocol is efficient with respect to its use of communications channels. The information in task announcements, for instance, helps minimize the amount of channel capacity consumed by communications overhead. Such efficiency helps to preserve whatever loose-coupling character is already present in the system as a result of problem partitioning.

The use of autonomous contract nodes interacting through a process of contract negotiation fosters distribution of control and data throughout the system, thus meeting the third design goal.

Maintenance of focus is perhaps the most difficult of the design goals to meet, and we do not yet have a good understanding of the underlying issues involved. Our approach is to attack the problem explicitly through "appropriate" definition of the functions used to evaluate task announcements and bids. In addition, each node maintains a list of the "best" recent task announcements it has seen - a kind of window on the tasks at hand for the net as a whole. This window enables the evaluation functions to "integrate" the local situation over time to assist in maintenance of focus.

It is of interest to note that the focus problem does not necessarily have to be attacked explicitly. Some problems lend themselves to a relaxation style of problem solving. Low level vision operations, for example, are suitable candidates for this approach [Zucker, 1977]. The nature of the relaxation process itself tends to produce global coherence from the actions of individual processes even though focus is not addressed explicitly as a problem. In this approach, a lack of appropriate global coherence shows up as oscillation in the relaxation process.

---

[7] At the top level, contracting can distribute control "almost" completely, hence removing the bottlenecks that centralized controllers may 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.

3 Example - Distributed Sensing

In this section, we demonstrate the use of the contract net approach in the solution of a problem in area surveillance, such as might be encountered in ship or air traffic control. The example will help to demonstrate the ideas which form the central foci of the remainder of this paper: (a) task distribution as an interactive process, and (b) indexing and distribution of knowledge.

We consider the operation of a network of nodes, each of which may have either sensing or processing capabilities, and which are spread throughout a relatively large geographic area. Such a network is called a ***Distributed Sensing System*** (***DSS***). The primary aim of the system is rapid, reliable, accurate, and low-cost analysis of the traffic in a designated area. This analysis involves detection, classification, and tracking of vehicles; that is, the solution to the problem is a dynamic map of traffic in the area which shows vehicle locations, classifications, courses and speeds. Construction and maintenance of such a map requires integration and interpretation of a large quantity of sensory information received by the collection of sensor elements.

There are a number of tradeoffs involved in the design of a DSS architecture, and we present only one possible approach. The primary intent of the example is to act as a vehicle for demonstration of the contract net approach to distributed problem solving.[8]

The example we present here is a hand simulation, but is based on a working SAIL simulation of the contract net that has been applied to a related distributing sensing problem.

3.1 Hardware

The DSS is organized as a contract net that is monitored by a distinguished processor node called the ***monitor node***. All communication in the net is assumed to take place over a broadcast channel. The nodes are assumed to be in fixed positions known to themselves but not known a priori to the monitor node, and they may have two different kinds of capability: sensing and processing. The sensing capability includes low level signal analysis and feature extraction. We assume that a variety of sensor types exist in the DSS, that the sensors are widely spaced, and that there is some overlap in sensor area coverage.

Nodes with processing capability supply the computational power necessary to effect the high level analysis and control in the net. They are not necessarily near the sensors whose data they process. These nodes are able to acquire (if necessary) the procedures essential to effect any of the information processing functions, by transfer from other nodes.

---

[8] Further discussion of the background issues inherent in DSS design is presented in **[Smith, 1978a]**; a more detailed discussion of this example is presented in **[Smith, 1978b]**, which includes examination of several of the design options and tradeoffs that can only be mentioned briefly here due to space limitations.

3.2 Data And Task Hierarchy

The DSS must integrate a large quantity of signal data, reducing it and transforming it into a symbolic form meaningful and useful to a human observer. We view this process as occurring in several stages, which together form a data hierarchy (Figure 3.1). The hierarchy offers an overview of DSS function and suggests a task partitioning suitable for a contract net approach.

overall area map

area map

vehicle

signal group

signal

Figure 3.1. Data Hierarchy.

For purposes of this example, the only form of signal processing we consider is narrow band spectral analysis, and the ***signal*** has the following features: frequency, time of detection, strength, characteristics (e.g. increasing signal strength), name and position of the detecting node, and the name, type, and orientation of the detecting sensor.

Signals are formed into ***signal groups*** at the second level of the data hierarchy. A signal group is a collection of related signals. For this example, the signal groups have the following features: the fundamental frequency of the group, the time of group formation, and the features of the signals in the group (as above).

The next level of the hierarchy is the ***vehicle***. It has one or more signal groups associated with it, and is described in terms of position, speed, course, and type. Position can be established by triangulation, using matching groups detected by several sensors with different positions and orientations. Speed and course must be established over time by tracking.

The next level of the data hierarchy is the ***area map***. This map incorporates information about the known vehicle traffic in an area. It is an integration of the vehicle level data. There will be several such maps for the DSS, corresponding to areas in the span of coverage of the net.

The final level is the complete or ***overall area map***. In this example, the map is integrated by the monitor node.[9]

---

[9] A DSS may have several functions, and not all of these functions will require integration of overall area data at a single node.

The hierarchy of tasks follows directly from the data hierarchy. The monitor node manages several ***area*** contractors (Figure 3.2). These contractors are responsible for formation of traffic maps in areas defined by the monitor node. Each area contractor in turn manages several ***group*** contractors that provide it with signal groups for its area (Figure 3.3). Each group contractor integrates raw signal data from ***signal*** contractors that have sensing capabilities.

The area contractors also manage several ***vehicle*** contractors that are responsible for integration of information associated with individual vehicles. Each of these contractors manages a ***classification*** contractor that determines vehicle type, a ***localization*** contractor that determines vehicle position, and a ***tracking*** contractor that tracks the vehicle as it passes through the area.[10]

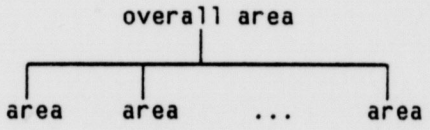

Figure 3.2. Traffic Map Partitioning.

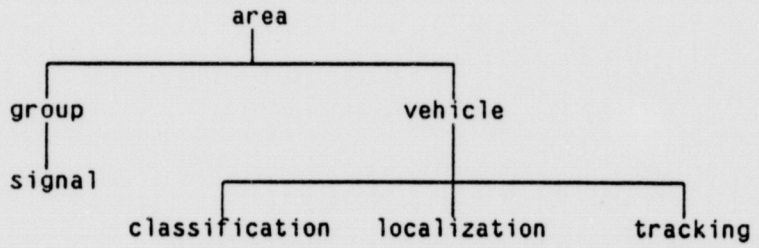

Figure 3.3. Area Task Partitioning.

3.3 DSS Initialization And Operation - The Contract Net Approach

This section reviews in qualitative terms how the DSS problem can be attacked using the contract net approach, and illustrates several of the ideas central to its operation. Appendix A gives specific examples of the message traffic that is described here in more general terms.

---

[10] In a real solution to the DSS problem, it is possible that not all of these tasks would be large enough to justify the overhead of contracting; that is, some of them might be done in a single node. It is also of interest to note that some of the tasks in the hierarchy are *continuing* tasks (e.g., the area task), while others are *one-time* tasks (e.g., the localize task).

3.4 Initialization

The monitor node is responsible for initialization of the DSS and for formation of the overall map. It must first partition the overall span of coverage of the system into areas, and select other nodes to be area contractors. For purposes of illustration we assume that the monitor node knows the names of nodes that are potential area contractors, but must establish their positions in order to partition the overall span of coverage. Hence, it begins by announcing contracts for formation of area maps of the traffic. Because the monitor node knows the names of potential area contractors, it can avoid a general broadcast and use a focused addressing scheme. The announcement contains the three components described in Section 2.2, a task abstraction, eligibility specification, and bid specification. The bid specification is of primary interest for this task. It informs a prospective area contractor to respond with its position. The monitor node uses the positions returned in bids on the task to form appropriate areas and select a subset of the bidders to be area contractors. Each contractor is informed of its area of responsibility in the contract award message.[11]

The area contractors now attempt to solicit other nodes to provide signal group data. In the absence of any information about which nodes might be suitable, each area contractor announces the task using a general broadcast. The eligibility specification in these announcements indicates the area for which the individual area contractor is responsible; that is, a node is only eligible to bid on this task if it is in the same area as the announcing area contractor. Potential group contractors respond with their respective positions, and based on this information, the area contractors award group contracts to nodes in their areas of responsibility.

At this point, the group contractors attempt to find nodes that will provide raw signal data. This is done with signal task announcements. The task abstraction in these announcements indicates both the area of responsibility of an individual group contractor and its position. This position information will assist potential signal contractors in determining the group contractors to which they should respond. The eligibility specification in the announcements ensures that a bidder is located in the same area as the announcer, and that it has sensing capabilities.

The potential signal contractors listen to the task announcements from the various group contractors. They respond to the nearest group contractor with a bid that supplies their position and a description of their sensors. The group contractors use this information to select a set of bidders that covers the vicinity with a suitable variety of sensors, and then award signal contracts on this basis. The awards specify the sensors that each signal contractor is to use to provide raw data to its managing group contractor.

The signal contract is a good example of the contract negotiation process, illustrating that the matching of contractors to managers is an interactive process. It involves a mutual

---

[11] The full announcement - bid - award sequence is necessary (rather than a directed contract) because the monitor node needs to know the positions of all of the potential area contractors in order to partition the overall span of coverage of the DSS into manageable areas. Note that this means that the DSS will automatically adjust to a change in the number or position of potential area contractors.

decision based on local processing by both the group contractors and the potential signal contractors. The potential signal contractors base their decision on a distance metric and respond to the closest manager. The group contractors use the distribution of sensor types and numbers observed in the bids to select a set of signal contractors which ensures that every area is covered by every kind of sensor. Thus each party to the contract evaluates the proposals made by the other, using a different evaluation function, and arriving at a task distribution agreement via mutual selection.

3.5 Operation

We now consider the activities of the system as it commences operation.

When a signal is detected, or a change occurs in the features of a known signal, the detecting signal contractor reports this fact to its manager (a group contractor). This node in turn attempts to integrate the information into an existing signal group or to form a new signal group.

A group contractor reports the existence of a new signal group to its manager (an area contractor) which must then decide what to do with it. Whenever a new group is detected, the managing area contractor attempts to find a node to execute a vehicle contract. The task of a vehicle contractor is to classify, localize, and track the vehicle associated with the signal group. Since a newly detected signal group may be attributable to a known vehicle, the area contractor first requests from the existing collection of vehicle contractors a measure of the confidence that the new group is attributable to one of the known vehicles. Based on the responses, the area contractor either starts up a new vehicle contractor, or awards a contract to (augments the existing contract of) the appropriate existing vehicle contractor, with the task of making certain that the new group corresponds to a known vehicle. This may entail, for example, the gathering of new data via adjustment of sensors, or contracts to new sensor nodes.

The vehicle contractor then makes two task announcements: vehicle classification and vehicle localization. The announcement of the classification task includes an abstraction of the available description of the vehicle (i.e., the currently known information). In this example, the abstraction contains a list of the fundamental frequencies of the signal groups currently associated with the vehicle. This information may help a potential classification contractor select an appropriate task (a contractor may, for example, already be familiar with vehicles that have signal groups with the announced fundamental frequencies). The award includes the complete current description. A classification contractor may be able to classify directly, given the signal group information, or on the other hand it may require more data, in which case it can communicate directly with the appropriate sensor nodes.

A localization task announcement includes data on the positions of the detecting nodes. The bid is simply an affirmative response to the announcement and the contract is awarded to the first bidder, which does the required triangulation to obtain the position of the vehicle.

Once the vehicle has been localized, it must be tracked. We assume that this is handled by the vehicle contractor which enters into follow-up localization contracts from time to time and uses the results to update its vehicle description.

There are a variety of other issues that must be considered in the design and operation of a real distributed sensing system; they are discussed in more detail elsewhere [Smith, 1978b]. In the following sections, we focus on issues of knowledge organization and use in the contract net, and refer back to this example to instantiate the issues raised.

4 Organization Of Knowledge

In this section we consider the contract negotiation process from a different perspective, examining the kinds of knowledge that are used, the way that the knowledge is indexed within an individual node, and distributed among the nodes.

We begin with a few definitions. ***Indexing*** indicates the "handles" placed on knowledge modules so that they can be accessed. In the next section we will see that knowledge in a contract net is indexed according to its utility for selecting suitable knowledge sources (KS's) (i.e., processor nodes) for a particular task, or for selecting suitable tasks for a particular KS. ***Distribution*** indicates where the knowledge resides; that is, in which processor nodes. We can distinguish two aspects of distribution of knowledge in a contract net: ***static*** distribution - dealing with the question of how knowledge is pre-loaded into the net (i.e., the a priori distribution), and ***dynamic*** distribution - how knowledge is acquired by a node as work on the problem progresses.

In the following sections we will concentrate primarily on the issue of knowledge indexing, together with the mechanisms that are necessary to use the knowledge in problem solving. We will also see that these same mechanisms permit interactive transfer of expertise between nodes in much the same way as any other form of information is transferred.

4.1 Indexing Of Knowledge

To consider knowledge indexing, the discussion focuses on (a) the two primary questions that must be answered by a node during the contract negotiation process, and (b) the types of knowledge that are used by the manager and potential contractors to effect this negotiation.

A manager has two questions to answer during the contract negotiation process. First,

(1) ***To whom do I address my task announcement?***

Then, once it has received a number of bids in response to an announcement, the manager must answer the question,

(2) ***How can I select the best candidates from among the potential contractors for my task?***

A node that receives an announcement must also answer two questions during the negotiation process. First,

(3) ***Am I relevant to this task and is it appropriate for me to consider making a bid?***

In addition, a node must also determine,

(4) ***Is this task the one that I want to execute next?***

In order to facilitate the contract negotiation process, we find it convenient to specify the indexing of knowledge as being either ***task-centered*** or ***knowledge-source-centered*** (***KS-centered***).

Task-centered knowledge is indexed from the point of view of a particular task, and provides information about KS's with respect to that task. At least two forms can be imagined:

(a) ***IF I have a task of the form [...] to be executed, THEN KS's of the form [...] are potentially useful.***

or,

(b) ***IF I have a task of the form [...] to be executed, THEN KS's of the form [...] are more useful than KS's of the form [...].***

KS-centered knowledge, on the other hand, is indexed from the point of view of a particular KS, and provides information about tasks with respect to that KS. Again, at least two forms can be imagined:

(c) ***IF my knowledge base contains information of the form [...], THEN tasks of the form [...] are appropriate for me.***

or,

(d) ***IF my knowledge base contains information of the form [...], THEN tasks of the form [...] are more appropriate for me than tasks of the form [...].***

Both kinds of knowledge are used during the contract negotiation process. Task-centered knowledge is used first to determine the subset of nodes to which to address a task announcement (i.e., (a) provides the answer to question (1)). This type of knowledge reduces message traffic and message processing overhead because it enables focused addressing, as in the DSS, for example, where the monitor node uses task-centered knowledge to effect focused addressing in announcing the area tasks.

Task-centered knowledge is also used to determine the best course of action once bids are received (i.e., (b) provides the answer to question (2)), and hence is an effective mechanism for encoding strategies. That is, since bid evaluation functions are used to select the next KS to invoke, they are an appropriate location for strategy information that guides the operation of the problem solver.

KS-centered knowledge is used by a node that receives an announcement, first to determine that it is relevant to the announced task (i.e., (c) provides the answer to question (3)). Associating knowledge with KS's allows enhancement of the concurrency in a

distributed processor because many KS's can simultaneously determine their relevance to a task; that is, each KS carries information allowing it to determine the range of tasks to which it is relevant. KS-centered knowledge of type (c) is used by nodes in the DSS example to determine that they are eligible to bid on signal tasks.

KS-centered knowledge is also used by a node to select the task it wishes to execute next (i.e., (d) provides the answer to question (4)). This type of KS-centered knowledge is another effective mechanism for encoding strategies. In the DSS, for example, the initialization strategy for signal contractors is encoded in this way.

4.2 Distribution Of Knowledge

We noted at the beginning of this section that distribution of knowledge has two aspects - static and dynamic. Static distribution is largely task-specific, and the criteria for a good static distribution of knowledge are similar to those for good problem partitioning. The distribution chosen should minimize message traffic, and should not create any bottlenecks in the system. Dynamic distribution of knowledge is the means by which nodes can acquire and transfer information and expertise as the problem progresses. The ability to effect dynamic knowledge distribution places several constraints on the design of the distributed problem solver.

Dynamic distribution of knowledge enables more effective use of available computational resources: a processor node that is standing idle because it lacks information required to perform a previously announced task can acquire the procedures necessary to execute that task. This also facilitates the task of adding a new node to an existing net, since the node can dynamically acquire the procedures and data necessary to allow it to participate in the operation of the net.

This means that nodes do not have to be functionally defined a priori; that is, any node can acquire the procedures necessary to execute any task that its physical attributes (e.g., memory, peripherals, etc.) will support. The alternative would be either forcing the node to remain idle until it hears a task announced for which it already has the necessary procedures, or pre-loading each node in the net with all the procedures that will ever be used. Neither of these alternatives is very attractive.

Procedures can be transferred between nodes in three ways. First, a node can transmit a request directly to another node for transfer of a procedure. The response to the request is the procedure code. Second, a node can transmit a task announcement in which the task is transfer of a procedure. A bid on the task indicates that another node has the code and is willing to transmit it. Finally, a node can note in its bid on a task that it requires the code for a particular procedure in order to execute the task. This is useful when the managing node already has the relevant code but wants to work on some other aspect of the task.

4.3 Internode Communication

Thus far we have concentrated on the role of the contract net problem solving protocol for interaction between nodes. In this section, we consider the ***common internode language*** which serves as the foundation on which the protocol is based. The language provides the primitive elements with which such items as task abstractions, eligibility specifications, and bid specifications are encoded. It thus provides the medium in which nodes "discuss" tasks and KS's, as well as pose the questions about eligibility to bid on tasks, rank ordering of tasks, and control of task distribution that arise during the contract negotiation process.

A relatively simple language, capable of supporting the DSS communication, has been designed. Sample messages are shown in Appendix A. It is believed that the language will support a range of other applications, but, of course would have to be increased in complexity for behavior significantly more complex than that shown in the DSS example.

The language is organized as a collection of associative triples, and has a set of domain-independent, "core" vocabulary items that can be extended with task-specific items.

The current grammar of the language is relatively simple, with the result that the messages shown in Appendix A are somewhat verbose. These messages have the advantage of being easy to write and understand for a human, but have the disadvantage of being less efficient than they might be in their use of communications resources.

The common internode language permits explicit statements of requirement to be made in messages. It is also useful in that it assists a new node in isolating the information it must acquire to participate in the operation of the net. This isolation is an aid to active distribution of knowledge (discussed in the preceding section). Finally, the language simplifies the use of local processing by a node, for example, to evaluate announcements and bids from its own point of view. A node is able to process the information in these messages because the common internode language affords a uniform interpretation of the vocabulary items by all nodes in the net.

Specialized communication is also possible. Two nodes that are linked via a contract, for example, can adopt a more compact form of communication for their messages, since no other nodes need interpret the messages. This compact form of communication can be viewed as a specialized language that the nodes use to communicate with other nodes that share their expertise. In the DSS, for example, once the area and vehicle contractors have established communication through the contract negotiation process, they might alter the language in which they communicate in order to reduce the length of messages and simplify message processing. This is possible because a context has been established through a contractual relationship.

5 Other Systems

The contract net draws upon a variety of ideas from the AI literature. In this section we relate the approach to those used in other systems.

5.1 PLANNER And Actors

The contract net task announcement is analogous to the PLANNER [Hewitt, 1972] goal specification, and functions similarly in providing a mechanism for advertising a task to a group of KS's, instead of invoking a specific KS by name.

By way of contrast, the contract net allows complex local processing by a node in determining its relevance to a particular task, rather than the pattern-matching that is allowed in PLANNER. In addition, the actor model of computation that succeeded PLANNER is based on the concept of a group of experts that communicate by passing point-to-point messages [Hewitt, 1977a], [Hewitt, 1977b], while there are a variety of addressing modes used in contract net messages (general broadcast, limited broadcast, and point-to-point). These different modes serve to reduce message traffic and message processing overhead. Finally, the contract net assumes a loose-coupling of tasks, whereas the actor model does not. This assumption implies a difference in the grain size of tasks into which a problem is decomposed (large for contractors and small for actors), and results from the different motivations of the designers of the two formalisms: where actors have been used as a means of studying fundamental issues involving the nature of computation, control, and program correctness, the contract net is designed as a mechanism for problem solving, and hence views its primitive operations in terms of comparatively large, domain-specific tasks.

5.2 HEARSAY-II

The concept of a group of cooperating KS's has been used to advantage in the HEARSAY-II speech understanding system [Erman, 1975]. The contract net draws upon this model with respect to the modularity and independence of KS's. Unlike this model, however, the contract net enables focused addressing and doesn't use a blackboard, primarily due to the problems such a global data structure can cause in a distributed environment (e.g., reliability and bottleneck problems).

In addition, KS's in the HEARSAY model were seen primarily as information gathering and dispensing processes [Reddy, 1975], so that hierarchical control was not considered necessary. The contract net, on the other hand, is well-suited to hierarchical control as a result of the manager-contractor structure.

Finally, HEARSAY-II did not preserve state information about a hypothesis. In particular, there was neither a way to specify the processing that had already been applied to a hypothesis, nor the kind of processing that might yet be applied, and this made scheduling difficult [Lesser, 1977]. The contract provides a data structure with which to associate this type of information and is one way of avoiding such problems.

5.3 PUP6

The model of a group of human experts cooperating to solve a large problem was also used effectively in PUP6, a system designed to write programs based on informal specifications [Lenat, 1975]. Where interaction between the modules in PUP6 was

accomplished by pattern-matching, the contract net expands on this through the use of a contract negotiation process, based on a common internode language.

PUP6 had no notion of acquired expertise, since each module in the system had a standard set of parts that did not vary over time. Contract nodes, on the other hand, have a standard core structure but have in addition a common internode language which enables them to acquire expertise via transfer of procedures and data.

5.4 Task Distribution / Transfer Of Control - A Progression

It will be useful at this point to compare the approach to task distribution provided by the contract net framework with that provided by previous problem solving formalisms. This will help make clear the ways in which the contract net view is unique and the advantages that uniqueness offers. We consider as points of comparison the techniques used in subroutine calling, PLANNER, CONNIVER [McDermott, 1974], HEARSAY-II, a hypothetical task agenda system, and the PUP6 system. We show that the contract net presents a view that is a natural successor to previous systems but is unique in several respects.

5.4.1 Terminology

We have used the term "task distribution" throughout the paper as a generalized view of what is more traditionally referred to as ***transfer of control***. That is, in a distributed system, when one processor decomposes a problem it is working on and hands one of the resulting subtasks to another processor, both processors continue working on their respective tasks; hence we refer to it as task distribution. In a uniprocessor, however, problem decomposition involves transfer of control: one process selects another process to work on a selected subtask and yields (perhaps temporary) control.

Since all of the systems we wish to use for comparison were designed for uniprocessors, we will adopt this perspective and make the comparison on the basis of transfer of control. This will provide a familiar basis for comparison without losing sight of any of the important issues. It will also serve to demonstrate that the issues we deal with in this section are fundamental issues of KS invocation and problem solving, independent of distributed processing.

5.4.2 The Basic Questions And Fundamental Differences

To make clear the place of the contract net in the sequence of invocation mechanisms that have been created, we consider the process of transfer of control from the perspective of both the caller and the respondent. We focus in particular on the selection aspects and consider what opportunities a calling process has for selecting an appropriate respondent, and what opportunities a potential respondent has for selecting the task on which to work. In each case we ask two basic questions from the perspective of both the caller and the respondent:

What is the character of the choice available?

On what kind of information is that choice based?

The answers to these questions will demonstrate our claim that the contract net view of control transfer differs with respect to:

(a) information transfer: The announcement-bid-award sequence means that there is more information, and more complex information transferred in both directions (between caller and respondent) before invocation occurs.

(b) local selection: 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).

(c) mutual selection: The local selection process is symmetric, in the sense that the caller evaluates potential respondents from its perspective (via the bid evaluation function), and the respondents evaluate the available tasks from their perspective (via the task evaluation functions).

5.4.3 The Comparison

Subroutine invocation represents a degenerate case, since all the selection is done ahead of time by the programmer and is "hardwired" into the code. As a result there is no non-determinism at runtime and hence no opportunity for choice.

A degree of non-determinism (and hence opportunity for choice) for the caller is evident in traditional production rule systems, since a number of rules may be retrieved at once. A range of selection criteria have been used (see [Davis, 1977]), but these have typically been implemented with a single, syntactic criterion hardwired into the interpreter.

PLANNER's pattern-directed invocation provides a facility at the programming language level for nondeterministic KS retrieval and offers, in the "recommendation list", a specific mechanism for encoding selection information. The THUSE construct provides a way of specifying which KS's (theorems) to try in which order, while the theorem base filter (THTBF) construct offers a way of invoking a predicate function of one argument (the name of the next theorem whose pattern has matched the goal) which can "veto" the use of that theorem.

Note that there is a degree of selection possible here, selection that may involve a considerable amount of computation (by the theorem base filter), and selection that is local in the sense that filters may be specific to a particular goal specification. But the selection is also limited in several ways. First, in the standard invocation mechanism the information available to the caller is at best the name of the next potential respondent; in effect a one-bit answer of the form "yes I match that pattern". The caller does not receive any additional information from the potential respondent (such as, for instance, exactly how it matched the pattern), nor is there any easy way to provide for information transfer in that direction. Second, the choice is, as noted, a simple veto based on just that single KS. That is, since

final judgement is passed on each potential KS in turn, it is not possible for instance to make comparisons between potential KS's, nor to pass judgment on the whole group and choose the one that looks (by some measure) the best. (Both of these shortcomings can be overcome if we are willing to create a superstructure on top of the existing invocation mechanism, but this would be functionally identical to the announcement-bid-award mechanism described above. The point is simply that the standard PLANNER invocation mechanism has no such facility and the built-in depth-first with backtracking makes it expensive to implement.)

CONNIVER represents a useful advance in this respect, since the result of a pattern-directed call is a "possibilities list" containing *all* the KS's that matched the pattern. While there is no explicit mechanism parallel to PLANNER's recommendation list, the possibilities list is accessible as a data structure and can be modified to reflect any judgments the caller might make concerning the relative utility of the KS's retrieved. Also, paired with each KS on the possibilities list is an association-list of pattern variables and bindings, making it possible to determine how the calling pattern was matched by each KS. This mechanism offers the caller some information about each respondent that can be useful in making the judgments noted above. As an indirect mechanism, however, it is less effective for information transfer than, for instance, an explicit bid mechanism.

The HEARSAY-II system illustrates a number of similar facilities in an event-driven system. In particular, the focus of attention mechanism has available a pointer to all the KS's that are ready to be invoked (so it can make comparative decisions), as well as information (in the "response frame") estimating the potential contribution of each of the KS's. The system can effect some degree of selection regarding the KS's ready for invocation and has available to it a body of knowledge about each KS on which to base its selection. The response frame thus provides information transfer from respondent to caller, which, while fixed in format, is more extensive than previous mechanisms. There is also a fair amount of computation devoted to the selection process, but note that the selection is not local, since there is a single, global strategy used for every selection.

There are several things to note about the systems reviewed thus far. First, we see an increase in the amount and variety of information that is transferred (before invocation) from caller to respondent (e.g., from explicit naming in subroutines to patterns in PLANNER) and from respondent to caller (e.g., from no response in subroutines to the response frames of HEARSAY-II). Note, however, that in no case do we have available a general information transmission mechanism. In all cases the mechanisms have been designed to carry one particular sort of information and are not easily modified. Second, we see a progression from the retrieval of a single KS at a time to explicit collection of the entire set of potentially useful KS's, providing the opportunity for more complex varieties of selection. Finally, note that all the selection so far is from one perspective; the selection of respondents by the caller. In none of these systems do the respondents have any choice in the matter.

To illustrate this last point, consider a (hypothetical) task agenda system in which there is a central "task blackboard" which contains an unordered list of tasks that need to be performed. As a KS works on its current task, it may discover new (sub)tasks that require execution, and add them to the blackboard. When a KS finishes its current task, it looks at the blackboard, evaluates the lists of tasks there, and decides which one it wants

to execute. Note that in this system the respondents would have all the selection capability; that is, rather than have a caller announce a task and evaluate the set of KS's that respond, we have the KS's examining the list of tasks and selecting the one they wish to work on.

PUP6 was the first system to view transfer of control as a "discussion" between the caller and potential respondents. If, in response to a task broadcast, a KS receives more than one reply offering to do the task, it may "ask" questions of the respondents to determine which of them ought to be used. While this interchange is highly stylized and not very flexible, it does represent an attempt to build in explicit two-way communication.

The contract net differs from all these in several ways. First, from the point of view of the caller, we have improved the standard task broadcast and response interchange by making possible a more informative response. That is, instead of the traditional tools which allow the caller to receive simply a list of potential respondents, we have available a mechanism which makes it possible for the caller to receive extensive information from each respondent describing potential utility.

Second, the contract net emphasizes the utility of local selection. That is, an explicit place in the framework has been provided for mechanisms with which both the caller (in the bid evaluation function) and the respondents (in the task evaluation function) can invest computational effort in selecting KS's for invocation or selecting tasks to work on, respectively. These selection functions are also "local" in the sense that they are associated with and written from the perspective of the individual KS (as opposed to, say, HEARSAY-II's global focus of attention function). While we have labelled this process "selection", it might more appropriately be labelled "deliberation", to emphasize that its purpose is, for the caller, for example, to decide in general what to do with the bids received, and not merely which of them to accept. Note that one possible decision is that *none* of the bids is adequate, and thus none of the potential respondents will be invoked. (Instead, the task may be reannounced later.) This choice is not typically available in other problem solving systems and hence emphasizes the wider perspective taken by the contract net on the transfer of control issue.

Finally, and perhaps most important, is what appears to be a novel symmetry in transfer of control process. Recall that PLANNER, CONNIVER, and HEARSAY-II all offered the caller some ability to select from among the respondents, while our hypothetical task agenda system allowed the respondents to select from among the tasks. The contract net, however, relies on the notion of contract negotiation as a metaphor, and emphasizes an interactive, ***mutual selection*** process in which task distribution is the result of a discussion between processors. As a result of the information exchanged in this discussion, the caller can select from among potential respondents (with its bid evaluation function), while the KS's can select from among potential tasks (with their task evaluation functions).

6 Limitations And Caveats

There are of course a number of limitations and caveats to consider. First, much of what we have proposed is a framework for problem solving that provides some ideas about what kinds of information are useful and how that information might be organized. There is still a considerable problem involved in instantiating that framework in the context of a specific task domain. Beyond the general guidelines offered earlier, it is not obvious, for instance, exactly what information should be in a task abstraction, bid, or task evaluation function. Yet the successful application of the machinery described above depends strongly on the choices made. In this sense, several of the mechanisms we have proposed are similar in spirit to the concept of the recommendation list in PLANNER: The mechanism provides a site for embedding a certain kind of information, but does not specify for a particular problem what goes in there, nor how to instantiate it in a particular domain. The utility of such mechanisms lies in their ability to help a user structure and understand a problem: We tread the traditional thin line between too much generality that provides too little guidance, and too much structure that overly constrains the user's options. More work on this is forthcoming, as we attempt to specify more detailed guidelines on appropriate use of the framework.

An important caveat in considering use of the contract net framework has been touched on earlier, in the issue of loose-coupling and the grain size of the problems attacked. It is apparent, for instance, that the communication involved in task announcements, bids, awards, etc., and the computation involved in the deliberation phase (the task and bid evaluations) may add up to a non-trivial amount of overhead. The size of the tasks being distributed must be such that it is worth this effort. It would make little sense, of course, to go through an extended mutual selection process to get some simple arithmetic done or to do a simple database access. While we discussed earlier how the full protocol can be abbreviated to an appropriately terse degree of interchange (e.g., directed contacts and the request/response mechanism), many other systems are capable of supporting this variety of behavior. The interesting contribution of our framework lies in applications to problems where the more complex interchange provides an efficient and effective framework for problem solving.

7 Conclusion

We have described the operation of a problem solver that is based on a collection of asynchronous processor nodes that cooperate according to a contract metaphor to solve problems. In this metaphor, task distribution is viewed as an interactive process of contract negotiation.

We have noted the ways in which the contract net protocol helps to reduce message traffic and message processing overhead - through the use of task abstractions, eligibility specifications, and bid specifications in task announcements, through the use of focused addressing, and through the use of specialized interactions like directed contracts and requests.

We have considered the indexing and distribution of knowledge in such a problem

solver. In this context, we have suggested two forms of knowledge indexing - task-centered and knowledge-source-centered - and demonstrated their utility in the context of a distributed sensing example.

We have noted that a common internode language is required to enable effective use of the knowledge in a distributed problem solver, and have sketched a rudimentary design for such a language.

While the ideas which form the basis of this paper have been derived from the point of view of designing a problem solver that can effectively exploit the multiple processor computer architectures that have been made possible by LSI technology, they appear to be more general in scope. Knowledge indexing and distribution, for example, are of interest in the design of future uniprocessor as well as multiple processor problem solvers.

8 Acknowledgements

The authors wish to acknowledge the assistance of Penny Nii in formulating the DSS example. Jan Aikins, Bruce Buchanan, Janet Friendshuh, Tom Mitchell, Earl Sacerdoti, Mark Stefik, and John Treichler provided helpful comments on earlier drafts of the paper.

Appendix A

DSS Sample Messages

This appendix includes abbreviated sample messages for the signal task in the DSS example. For brevity, the messages shown contain only the information mentioned in Section 2.2. Terms written in upper case are included in the core internode language, while terms written in lower case are specific to the DSS application.

For purposes of explanation, pseudo-English equivalents to the messages are also shown. The DSS of course has no human-like language processing capabilities.

Signal Task

Announcement: Needed - signal data for traffic in area A. My position is p. If in possession of sensors and located in area A, respond with position, and type and number of sensors.

Task Abstraction: TASK NAME signal
area name A
NODE POSITION p
Eligibility Specification: MUST HAVE DEVICE TYPE sensor
MUST HAVE OWN NODE POSITION
area name A
Bid Specification: BID OWN NODE POSITION
BID EVERY DEVICE TYPE
sensor type number

Bid: Position - q. Sensors: Type S - 3, Type T - 1.

Node Abstraction: NODE POSITION q
sensor type S number 3
sensor type T number 1

Award: Report signals. Use sensors S1 and S2.

Task Description: sensor name S1
sensor name S2

Report: Detected signal: frequency f0, time of detection t0, strength s0, characteristics (...), detecting-node s1, position p2, sensor A1, orientation a.

Result Description: signal name S1
frequency f0
time-of-detection t0
strength s0
characteristics (...)
detecting-node name s1
position p2
sensor name A1
type A
orientation a

References

[Anderson, 1975]
G. A. Anderson and E. D. Jensen, Computer Interconnection Structures: Taxonomy, Characteristics, and Examples, *Computing Surveys*, Vol. 7, No. 4, December 1975, pp. 197-213.

[Baer, 1973]
J.-L. Baer, A Survey of Some Theoretical Aspects of Multiprocessing, *Computing Surveys*, Vol. 5, No. 1, March, 1973, pp. 31-80.

[Brooks, 1975]
F. P. Brooks, Jr., *The Mythical Man-Month*, Addison-Wesley, Reading, Mass., 1975.

[Crocker, 1972]
S. D. Crocker, J. F. Heafner, R. M. Metcalfe, and J. B. Postel, Function-Oriented Protocols For The ARPA Computer Network, *SJCC Proceedings*, 1972, pp. 271-279.

[Davis, 1977]
R. Davis and J. King, An Overview Of Production Systems, in E. W. Elcock and D. Michie, eds., *Machine Intelligence 8*, Wiley, N. Y., 1977 pp. 300-332.

[Erman, 1975]
L. D. Erman and V. R. Lesser, A Multi-level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge, *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, Tbilisi, USSR, September 1975, pp. 483-490.

[Galbraith, 1974]
J. R. Galbraith, Organizational Design: An Information Processing View, in D. A. Kolb, I. M. Rubin, and J. M. McIntyre, eds., *Organizational Psychology*, 2nd edition, Prentice-Hall, Englewood Cliffs, N. J., 1974, pp. 313-322.

[Hayes-Roth, 1977]
F. Hayes-Roth and V. R. Lesser, Focus Of Attention In The HEARSAY-II Speech Understanding System, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977, pp.27-35.

[Hewitt, 1972]
C. Hewitt, *Description And Theoretical Analysis (Using Schemata) Of PLANNER: A Language For Proving Theorems And Manipulating Models In A Robot*, MIT AI TR 258, April 1972.

[Hewitt, 1977a]
C. Hewitt, Viewing Control Structures As Patterns Of Passing Messages, *Artificial Intelligence*, 8, 1977, pp. 323-364.

[Hewitt, 1977b]
C. Hewitt and H. Baker, Laws For Communicating Parallel Processes, in B. Gilchrist, ed., *Information Processing 77*, North-Holland, 1977, pp. 987-992.

[Kahn, 1972]
R. E. Kahn, Resource-Sharing Computer Communications Networks, *Proc. IEEE*, Vol. 60, No. 11, November 1972, pp. 1397-1407.

[Lenat, 1975]
D. B. Lenat, Beings: Knowledge As Interacting Experts, *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, Tbilisi, USSR, September 1975, pp. 126-133.

[Lenat, 1976]
D. B. Lenat, *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*, STAN-CS-76-570 (HPP-76-8), Department Of Computer Science, Stanford University, July 1976.

[Lesser, 1977]
V. R. Lesser and L. D. Erman, A Retrospective View Of The HEARSAY-II Architecture, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977, pp. 790-800.

[McDermott, 1974]
D. V. McDermott, and G. J. Sussman, *The Conniver Reference Manual*, MIT AI Memo 259a, January 1974.

[Reddy, 1975]
D. R. Reddy and L. D. Erman, Tutorial On System Organization For Speech Understanding, in D. R. Reddy, ed., *Speech Recognition*, Wiley, N. Y., 1975, pp. 457-479.

[Smith, 1977]
R. G. Smith, The CONTRACT NET: A Formalism For The Control Of Distributed Problem Solving, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977, p. 472.

[Smith, 1978a]
R. G. Smith, *Issues In Distributed Sensor Net Design*, HPP-78-2 (Working Paper), Heuristic Programming Project, Stanford University, January 1978.

[Smith, 1978b]
R. G. Smith, *A Framework For Problem Solving In A Distributed Processing Environment*, doctoral dissertation, Stanford University, 1978, (in preparation).

[Zucker, 1977]
S. W. Zucker, *Vertical And Horizontal Processes In Low Level Vision*, Report No. 77-4, Department of Electrical Engineering, McGill University, May 1977.