82

A HIGH-LEVEL SIMULATION TESTBED FOR COOPERATIVE DISTRIBUTED PROBLEM SOLVING

Victor Lesser, Daniel Corkill, Jasmina Pavlin, Larry Lefkowitz,
Eva Hudlicka, Richard Brooks, and Scott Reed

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts, 01003

ABSTRACT

The testbed provides a research tool for empirically evaluating alternative designs for cooperative distributed problem solving systems. The testbed simulates a network of nodes, each of which is an architecturally-complete Hearsay-II-like system, extended to include goal-directed control. The nodes attempt to identify, locate, and track patterns of vehicles moving in a two-dimensional space.

Incorporated into the testbed are capabilities for varying the accuracy of individual knowledge sources. This is accomplished through the use of an oracle which can compare the developing interpretations with the interpretation that would be produced if the system had perfect knowledge. These capabilities permit the study of how different control and communication policies perform under varying distributions of uncertainty and error in the intermediate states of processing. Additionally, both vehicle and sensor characteristics can be varied, permitting control of the spatial distribution of ambiguity and error in the task input data. Node configurations and communication channel characteristics can also be independently varied in this simulated system.

I. INTRODUCTION

We have been exploring a new paradigm for distributed problem solving systems in which the distributed system is able to function effectively even though processing nodes have inconsistent and incomplete views of the data bases necessary for their computations. This paradigm is appropriate for distributed applications in which the data necessary to achieve a solution cannot be partitioned in such a way that a node can complete a subtask without seeing the intermediate state of processing at other nodes.

An example of this type of application is distributed vehicle monitoring. Vehicle monitoring is the task of generating a dynamic, area-wide map of vehicles moving through the monitored area. In distributed vehicle monitoring, processing nodes with their associated acoustic sensors (of limited range and accuracy) are geographically distributed over the area to be monitored [LACO78, SMIT80]. Each processsing node can communicate with other nearby nodes over a packet radio communication network [KAHN78]. Because acoustic sensors characteristically produce a significant amount of error, purely localized processing of sensory data would result in "identification" of non-existent vehicles, missed detection of actual vehicles, and incorrect location and identification of actual vehicles. In this application, the amount of communication required to redistribute the raw sensory data necessary for correct localized processing would be significant.

An alternative approach for resolving these errors is for processing nodes to interact in a highly cooperative way, exchanging tentative and possibly incorrect partial results with one another. For example, each node's tentative vehicle identifications can be used to indicate to other nodes the areas in which vehicles are more likely to be found and the details (vehicle type, rough location, speed, etc.) of probable vehicles. In addition, consistencies between these tentative identifications serve to reinforce confidence in each node's identifications. Such cooperation is not only appropriate for vehicle identification, but also potentially useful in other stages of processing (identification of raw signals, groups of harmonically related signals, patterns of vehicles, etc.).

In order to perform this cooperative style of distributed processing, each node must:

- have goals in common with other nodes;
- be able to work asynchronously with other nodes, attempting to do the best it can with available data;
- make its own control decisions based on its view of the state of network problem solving.

This leads to a view of the distributed system as a society of cooperating, semi-autonomous nodes. We

This research was sponsored by the National Science Foundation under Grant MCS-8006327 and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041.

call this style of distributed processing functionally accurate, cooperative (FA/C) [LESS81a, LESS80a].

In order to evaluate this new approach, we have designed a parameterized simulation testbed. This testbed permits the evaluation of different control/communication strategies under different distributions of problem solving expertise in the system and a wide rage of task characteristics. The testbed simulates a model of a distributed, knowledge-based, problem solving architecture applied to an abstracted version of the distributed vehicle monitoring task. We anticipate, the parameterized task and architecture together will provide a very powerful tool for investigating the utility and limitations of the FA/C approach to the design of distributed problem solving systems.

We next describe the issues we wish to explore using the testbed and then present the simulated vehicle monitoring task, followed by a detailed discussion of the simulated vehicle monitoring system. Later sections describe how we can simulate and evaluate the performance of various system components, overview tools that help an experimenter define experiments and analyze their output, review the current status of the testbed implementation, and outline future research directions.

II. IMPORTANT ISSUES FOR FA/C PROBLEM SOLVING

Strategies for high-level cooperation and control are vitally important in an FA/C system because of the potential for resource overloads, imbalances and lack of coherence when there is no global view or control of the problem solving system. For example, such problems arose in a distributed version of the Hearsay-II speech understanding system (a rudimentary FA/C system) in which nodes interacted through the exchange of a small number of hypotheses and each node determined locally what work it should perform. In experiments with this system, we observed that this data-directed and self-directed control regime can potentially lead to redundant and unnecessary processing [LESS80a]. Situations occurred when a node had obtained a good solution in its area of interest and, having no way to redirect its attention to new problems, simply produced alternative but worse solutions. The problem of enlarging or changing a node's area of interest is nontrivial since it may become necessary later to go back and reevaluate old activities using new information. Another problem occurred when a node had bad data and could not possibly find a good solution without help from other nodes. Similarly, when nodes produce solutions at different rates, nodes with bad solutions will distract other nodes by communicating their results. Thus, it appears that a data-directed and self-directed form of control may not always provide sufficient global coherence among the nodes. We believe that in situations of dynamically changing task characteristics and changing processing capacities, additional control mechanisms will be necessary for maintaining global coherence.

There are two approaches for obtaining increased global coherence. The first is to provide each node with a better view of the state of problem solving in the network so that its data-directed and self-directed control decisions are more informed and consistent. This can be accomplished by having nodes exchange detailed meta-information about the state of their local problem solving and what they have learned about the states of other nodes. Another approach, which is compatible with the first, is to integrate goal-directed and externally-directed forms of control into network problem solving [SMIT81]. These types of control can be used to institute more precise and non-local control over the activities of individual nodes.

We believe that both approaches for obtaining improved global coherence are necessary for effective problem solving in some FA/C distributed systems. As will be discussed later, a node must make more complex local control decisions in order to support both approaches; it should be capable of planning sequences of activities and adapting its plan based on self-awareness of its previous activities, its problem solving role in the network, and the status and role of other nodes in the network. It is through interaction of network communication and control strategies with this more complex local control that improved global coherence can be achieved. This interaction will provide the flexibility to handle control and data uncertainty while still maintaining a sufficient level of global coherence to guarantee that acceptable solutions will be generated within given resource constraints.

Our initial approach to exploring a more sophisticated control regime has been to extend the distributed Hearsay-II architecture by adding a planning module to each node and permit communication of goals [CORK81, CORK82a]; this planner can adapt local node activity to respond to both externaly-directed requests by other nodes (goals) and the self-directed processing needs of the node based on the data it is receiving and the results it has so far produced. Furthermore, the planner can bias scheduling of these activities based on the problem solving relationships of the node with other nodes in the system. In such an approach, the priority given to goals received from other nodes versus local data-directed activity determines the degree of externally-directed versus self-directed control present in the system. Our approach is to permit both types of coordination concurrently, and to develop adaptive mechanisms for the system that dynamically determine an appropriate balance.

One of the ways that this sophisticated and self-aware local node control can be exploited is to split the network-wide control problem into two concurrent activities [CORK80, CORK82b]:

1. construction and maintenance of a network-wide organizational structure;

2. continuous local elaboration of this

structure into precise activites using the local control capabilities of each node.

The organizational structure specifies the information, communication, and control relationships among the nodes in only a very general way. Included in the organizational structure are control decisions that are not quickly outdated and that pertain to a large number of nodes. The organizational structure represents general "ballpark" control decisions (i.e., strategic plan describing and delimiting general responsibilities) which are dynamically tailored by the local node control.

The existence of an organizational structure provides a control framework which reduces the amount of control uncertainty present in a local node (due to the lack of incomplete or errorful local control information) and increases the likelihood that the nodes will be coherent in their behavior. The organizational structuring approach to limiting control uncertainty still preserves a certain level of control flexibility for a node to adapt its local control to changing task and environment conditions. As the nature of the problem solving environment changes, the distributed system may need to change its organizational structure to maintain its effectiveness (this is called organizational self-design). In order to effect such a change, the distributed system must:

- detect the decreased effectiveness of its present organizational structure;
- propose alternative organizational structures;
- evaluate the cost of continuing with its current organizational structure versus reorganizing itself into a more appropriate structure;
- carry out the reorganization if appropriate.

One of the main uses of the testbed is as a realistic environment within which to implement and empirically evaluate such approaches to network coordination.

III. A HIGH-LEVEL TESTBED FOR INTERPRETATION SYSTEMS

The high-level simulation testbed we have developed is based on an abstracted distributed vehicle monitoring task that, as well as being realistic, will allow us to explore empirically a much larger part of the design space than was possible with the distributed speech understanding system experiments. Figure 1 shows a typical layout of a task environment, with sensors, processing nodes, and vehicles. This task is a representative example of <u>distributed interpretation</u> problems, which we feel are appropriate for an FA/C approach.

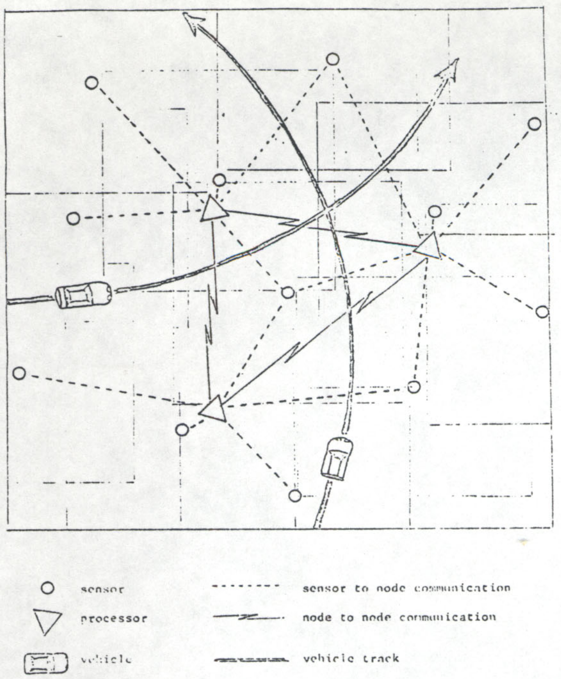

Figure 1: Vehicle Monitoring Task

The vehicle monitoring task has been chosen for several reasons. It is a natural task for a distributed approach, since sensors are in different geographical locations. Also, the task can be easily varied in terms of signal complexity, temporal and spatial constraints on possible interpretations, and configurations of sensors and nodes in the monitoring system. This variability permits us to consider the effects on system behavior of different types, spatial distributions, and degrees of uncertainty in the task. The number and size of the vehicle patterns can also be varied in order to modify the strength of the spatial interdependence of events. Additionally, the type of processing required to accomplish the task leads to a decomposition into disjoint levels of abstraction, with independent processing possible at each level.

In addition to abstracting the task, we have also made simplifying assumptions about the knowledge processing required to solve the problem. We have chosen these abstractions and simplifications because the actual task is highly complex and thus requires much effort to successfully engineer the required knowledge, and excessive processing resources to simulate. Parameterization of the actual task is also much more difficult. We have at the same time chosen not to simulate a highly abstracted version of the task. A more abstracted version of the task, while flexible and efficient to simulate, would not capture all the knowledge- and task-related behavior we want to study. The level of detail we have chosen permits the system to produce actual interpretations, and thus we can be more sure from its results of the nature and interactions of knowledge needed to solve an actual problem.

The next section of the paper describes a node in the system. This includes the basic architecture of a node, the blackboard structure of a node, the details of knowledge source and goal processing in a node and a description of our method of parameterizing and measuring problem solving expertise in the system.

IV. THE PARAMETERIZED DISTRIBUTED INTERPRETATION SYSTEM

The testbed simulates a network of problem solving nodes applied to the vehicle monitoring task. Each node is structured according to a generalized and extended Hearsay-II architecture with task processing modules called knowledge sources (KSs), global data structures for intermodule interaction and control called blackboards, a scheduler, and a planner (Figure 2). Interprocessor communication can be naturally included in the architecture by adding KSs on each blackboard level to send and receive messages (hypotheses and goals).

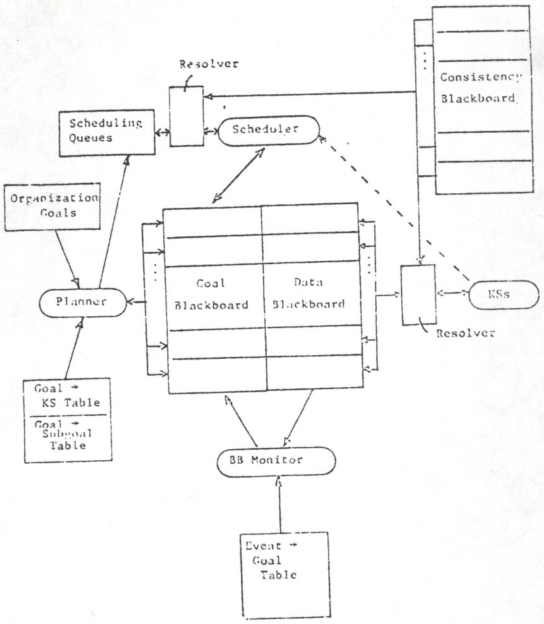

Figure 2: Architecture with resolver and consistency blackboard.

Our simulation model permits arbitrary node-node, node-sensor connectivity and arbitrary distributions of task processing capability across the nodes in the system. This architecture also permits exploration of a wide range of different processing decompositions (flat, hierarchical, matrix, etc.) based on partially configured nodes (those without all necessary KSs and with limited areas of spatial/temporal interest) without modifying the KS modules and local control structures.

4.1 Blackboard Partitions in a Node

There are two parallel blackboard structures in each node of our system: data, and goal.

The data blackboard corresponds to the blackboard of an actual monitoring system. It contains hypotheses each representing a plausible interpretation for a portion of the node's data; the likelihood of this interpretation is indicated by the belief value of the hypothesis. The data blackboard is the only blackboard from which KSs input and is the site of all KS stimulus events.

The goal blackboard holds goals, each representing a request to create a set of hypotheses with specific attributes on the data blackboard in the (corresponding) area covered by the goal. For example, a simple goal would be a request for the creation of a vehicle track hypothesis above a given belief in a specified area of the data blackboard. The goal blackboard is used by the scheduling and planning mechanisms to decide which KSs should be instantiated.

The blackboards of a node can be thought of as having four-dimensions, because hypotheses can be addressed by their level of abstraction, the two spatial (x,y) coordinates of the hypothesized event, and the time of the event. Information classes in the system, in order of increasing abstraction, are signal, group (harmonic set), vehicle and spatial patterns of vehicles. For each information class, there are two blackboard levels, one representing hypotheses about the location of a single event, and the other representing a track made from events which are consecutive in time and space (see Figure 3). Location blackboard levels are: signal location (SL), group location (GL), vehicle location (VL) and pattern location (PL). Track blackboard levels are: signal track (ST), group track (GT), vehicle track (VT) and pattern track (PT). Tracks can be formed from various combinations of locations and tracks of the same information class, or from the tracks of the lower abstraction information class. Locations can be formed from the locations of the lower abstraction information class.

4.2 Details of KS Processing in a Node

The patterns of KS processing arise from the interaction of three distinct types of KSs: synthesizers, mergers, and extenders.

A synthesizer KS creates higher-level hypotheses by abstracting the information contained in a group of lower level hypotheses. There are three types of synthesizer KSs in our system which work from location to location, location to track, and track to track levels on the data blackboard. An example of the first type, S:SL:GL KS, is triggered by the creation or modification of a signal-location (SL) hypothesis, and generates group-location (GL) hypotheses, combining the signals belonging to the same harmonic set. An example of the second type, S:SL:ST KS, is triggered by the creation or modification of a signal-location hypothesis, and generates signal-

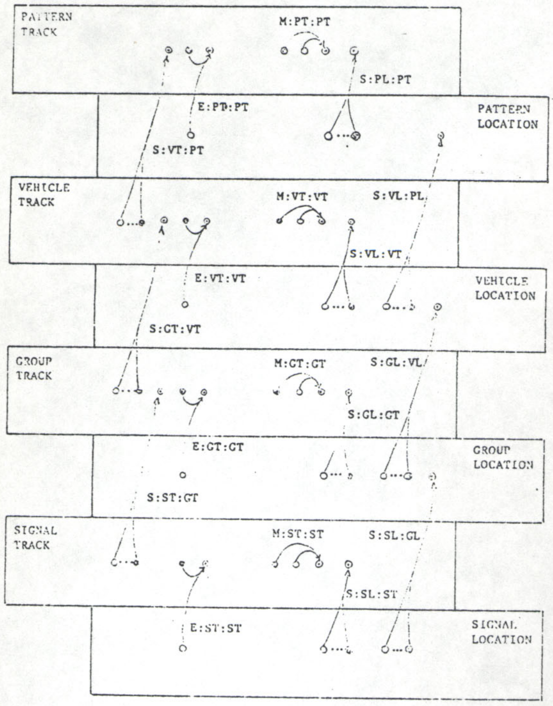

Figure 3: Data blackboard showing levels and KSs.

track (ST) hypotheses combining consecutive signal-location hypotheses of similar frequencies. An example of the third type, S:ST:GT KS, is triggered by the creation or modification of a signal-track hypothesis, and generates group-track (GT) hypotheses combining the tracks of signals in the same harmonic set.

A merge KS is different from synthesis KSs because it does not produce a hypothesis at a higher level of abstraction than its input but rather at the same level. There is a merge KS for each track level on the blackboard. The criteria for merging two tracks of the same event class is that the tracks overlap in time and location. The resulting merged track is supported by the two overlapping tracks. This type of KS is necessary for integrating track hypotheses received from other nodes into the blackboard of the receiving node. This KS is triggered by the creation or modification of a track hypothesis.

An extender KS creates track hypotheses from an existing track hypothesis and location hypotheses on the same abstraction level by extending the track forward or backward in time. These KSs are triggered by the creation or modification of a track hypothesis. There is an extender KS for each level of abstraction.

4.3 Details of Goal Processing in a Node

In order to permit more sophisticated forms of cooperation among nodes in the system, we have integrated goal-directed control into the data-directed control structure of the basic Hearsay-II architecture. This has been accomplished through the addition of a planning module and a goal blackboard. This integrated control framework permits nodes to effect the processing of other nodes not only through the transmission of hypotheses but also through transmission of goals which are requests for the creation of hypotheses with certain attributes.

The integration of data-directed and goal-directed control into a single framework is based on the following observation:

> The stimulation of a KS in the data-directed Hearsay-II architecture not only indicates that it may be possible to execute the KS, but that it may be desirable to do so in order to achieve the goal implicit in the output generated by the KS.

In order to make these implicit goals explicit, we split the event -> KSs mapping into two steps: event -> goals and goals -> KSs. The blackboard monitor watches for the occurrence of a data blackboard event, but instead of placing KS instantiations on the scheduling queue (if the KS preconditions are satisfied), it uses the event -> goals mapping to determine the appropriate goals to generate from the event and inserts them onto the goal blackboard. These goals represent the implicit goals contained in the event -> KSs mapping used by the original blackboard monitor.

A new control component, the planner, is also added to the architecture. The planner responds to the insertion of goals on the goal blackboard by developing plans for their achievement. The goal -> KSs mapping is used by the planner to instantiate one or more KSs which can potentially satisfy the goals. The scheduler then uses the relationships between the KS instantiations and the goals on the goal blackboard as a basis for its scheduling decisions.

The planner also has the responsibility of integrating externally received goals into the node's current goal structure. This integration is accomplished by linking together external and internal goals through goal-subgoal relationships. Local node activity can then be biased so as to satisfy external goals by rating higher those internal goals that help achieve the external goals. The higher the rating attached to these internal goals, the more explicit the control relationship among nodes can be made.

The planner also bases its goal rating decisions on 1) its view of the current state of processing in the node, 2) the organizational roles it is currently adhering to, and 3) the importance attached to the goal by the sender. These organizational roles are stored on an additional blackboard called the organizational blackboard. In our current implementation, the following factors are used to define the organizational roles of a node:

- o the organizational importance of having the node generate hypothesds at particular levels, times, areas, and event-classes;
- o the organizational importance of having the node send to and accept hypotheses and goals at particular levels, times, areas, and event-classes from particular nodes.

The more importance the planner gives to organizational roles the more globally coordinated the system can be made.[1] A more comprehensive description of goal processing and the role of organizational design in FA/C systems is given in CORK80, CORK81, CORK82a, and CORK82b.

We have currently implemented a simple version of the goal send and goal receive KSs based on the following algorithm. A goal becomes a candidate for transmission if its characteristics match those specified by the node's organizational role. The priority of sending this candidate goal is then determined by the following factors:

- o the local rating associated with the goal;
- o the importance that is attached to sending this type of goal as specified by the node's organizational role;
- o whether the goal can be satisfied locally (i.e., there are KSs that could be run which can potentially satisfy this goal);
- o whether the goal is linked to any subgoals which could be potentially satisfied as a result of further node processing.

Also associated with a node's organizational role is a minimal threshold for transmission; if the goal is below this threshold, it is not transmitted. Otherwise, the goal is placed on a priority rated queue and is transmitted when all more highly rated goals have been transmitted. A similar scheme is used to rate hypotheses for transmission, and hypotheses that are candidates for transmission are placed on the same priority rated queue as is used for goals. This use of a single queue for transmission permits the easy specification of mixed initiative transmission policies.

In the case of a receiving node, if an externally received goal is satisfied as a result of local processing, the node can respond in one of the following ways based on the external goal's attributes:

- o it can do nothing;
- o it can transmit an acknowledgement to the sender of the goal that the goal is satisfied;
- o it can transmit the satisfying hypotheses and possibly the acknowledgement of satisfaction to the sender of the goal.

As additional options, all the nodes that have seen this goal can be notified, or just a selected set of nodes (e.g., the original node that generated the goal).

In the situation in which a node cannot satisfy an externally received goal the node may use the model of other nodes processing contained in its organizational roles, to pass on this goal to other nodes in the network for possible satisfaction. In this way a store-and-forward goal processing network can be built-up. If appropriate organizational roles are set-up, this network can be made to simulate the hierarchical flow of goals in the contract-net approach. However, currently there is no bidding protocol, like that in the contract-net approach [SMIT80], for dynamically choosing which nodes should receive a goal. As we increase the sophistication of the send/receive goal KSs and transmit organizational roles among nodes, we hope to be able to simulate more sophisticated coordination and communication policies including the use of negotiation.

V. MODIFYING THE POWER OF KNOWLEDGE SOURCES

An important aspect of our empirical evaluation of alternative FA/C system organizations is the measurement of how their effectiveness changes as the distribution of problem solving expertise is varied. For example, we conjecture that in a system with very reliable KSs and with input data that has low error, organizing the system hierarchically and using an explicit control and communication strategy is more effective. Likewise, it is conjectured that in systems with less reliable KSs and with more errorful input data, more cooperative and implicit control/communication strategies are desirable. We would also like to examine the effects in performance caused by the trade-off between reliability of KSs versus processing time; i.e., fast but less reliable KSs versus highly reliable but slower KSs.

We can simulate KSs of any desired power in the testbed through the use of an oracle which knows how to judge the closeness of a hypothesized interpretation to a consistent interpretation. Each node in the system has access to a hidden data structure called the consistency blackboard, which is precomputed from the simulation input data. This blackboard holds what the interpretation would be at each information level if the system worked with perfect knowledge. This blackboard is not part of the basic problem solving architecture of a node, but rather it is used by the testbed oracle to regulate the power of KSs and the scheduler, and to dynamically measure the problem solving performance of the simulated system with respect to the simulation data.

A KS is simulated in two stages: candidate generator and resolver. The candidate generator stage produces plausible hypotheses for KS output

1. Global coordination is not a positive characteristic if the organizational structure is inappropriate to the problem solving situation.

and assigns initial belief values. It can be implemented either by using a real KS whose performance needs to be upgraded or by constructing a simplified KS incorporating relatively simple domain knowledge. In many cases, it is relatively easy to design a KS which provides moderate accuracy. Most of the effort in knowledge engineering is spent in increasing this accuracy to gain superior performance. The next stage, the resolver, uses information provided by the oracle to minimally alter the initial belief values of the hypotheses output by the candidate generator to achieve, on the average, a desired KS resolving power. With this alteration process, we can simulate the detection of more sophisticated forms of local consistency than is provided by the candidate generator. This technique allows us to simulate, in a real system, either an upgraded version of a KS (where the old KS is used as candidate generator) or, as we have done in the testbed, a totally new KS (with partially developed knowledge), in order to understand the system performance implications of a proposed change [LESS80b, LESS81b]. A similar approach has been used to introduce a parameterized scheduler into the testbed.

VI. FACILITIES FOR EXPERIMENTATION

The testbed kernel is surrounded by a number of other subsystems to facilitate experimentation by making it easy to vary the parameters of an experiment and to analyze the results of an experiment (Figure 4).

The Frontend subsystem is responsible for initializing the testbed with a particular configuration of nodes and sensors and a particular

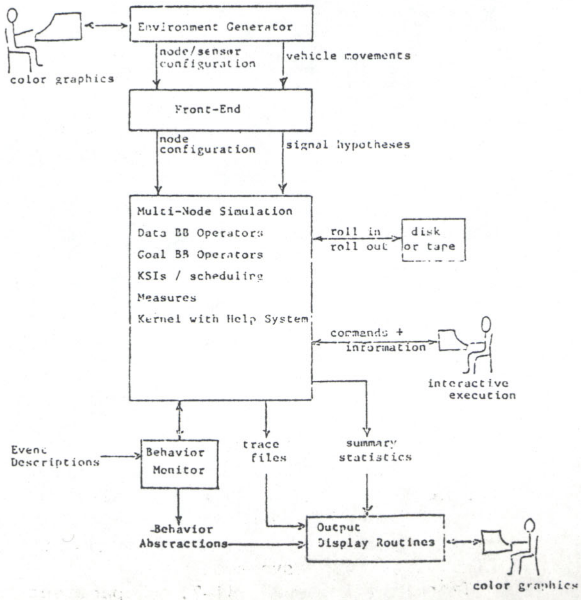

Figure 4: Diagram of testbed Structure and facilities

scenario of patterns and vehicles moving in the environment. This initialization process involves reading an input file to set up local and global data structures, creating the consistency blackboard, and placing at appropriate times the raw signal data on the data blackboards of appropriate nodes. Each configuration-scenario combination is stored compactly in an input file. This includes the information contained in the node and sensor definitions as well as communication (node-node and sensor-node) and environmental data.

By varying the grammar specification to the Frontend, the number of legal patterns of hypotheses can be varied. The most constrained grammar would be one that only allowed the particular scenario for the experiment in question to be recognized. Thus, the nature and the scope of consistency constraints used by KSs to resolve errors can be altered. This ability to modify the grammar combined with the ability to vary the local resolving power of KS provides a powerful tool for varying the knowledge expertise in the simulated system. The Frontend, in the generation of sensor data, can introduce controlled error (noise) to model imperfect sensing. Noise is added to the location and signal class of each true and/or consistent-false signal according to the sensor class and the distance of the signal from the sensor. Frontend processing is also parametized so that either these signals can be introduced into the nodes all at once or at the time they are sensed. The former provision allows exploration of systems in which there are burst receptions of sensor data.

In order to help in the analysis of the results of an experiment, a number of tools have been developed: a selective trace facility, a summary statistics facility, and an interactive, menu-driven debugging facility. In addition to these three fairly common analysis tools, we feel that there is need for tools that permit a more dynamic and high-level view of the distributed and asynchronious activity of the simulated nodes. An event monitoring facility, which has not yet been fully implemented, will permit a user to define and gather statistics on such user-defined events as the average time it takes for a node to receive a message and incorporate the received information into a message to be transmitted to another node. A description of this facility is contained in BATE81. Another facility which is currently operational in a limited form is a color-graphics output facility. The current output display provides dynamic visual representations of the distribution of hypotheses in the x-y space of the nodes in the network during a simulation.

VII. PRELIMINARY EXPERIMENTS

We are currently running a set of experiments in the testbed on a simple scenario comparing the performance of a centralized version, a laterally organized, 4-node system with broadcast communication among nodes at the vehicle track level, and a hierarchical, 5-node system which is similar to the 4-node system except that instead of transmitting vehicle track hypotheses to the other nodes, a fifth node without sensors is inserted which receives all the communications and generates the answer.

By varying the organizational roles of nodes through parameters on their organizational blackboard, a number of different communication/control strategies can be evaluated for each of these different architectures. The communication strategies we have been examining in these preliminary experiments are whether information should be transmitted among nodes on a voluntary, self-directed basis (a node transmits what it thinks from its local perspective is an important piece of information) or as a result of an external request (a node transmits only information that is requested by another node as a result of a goal transmission). We also have been exploring strategies in which a node engages in both self-directed and externally-directed communication. In these mixed initiative strategies, the choice of information to send is based on balancing the importance attached to the information locally (hypothesis belief), the importance of satisfying the requests of the other node (organization authority relationships among the nodes), and the importance the node requesting the information attaches to the request (priority attached to the received goal). These different communication strategies are defined by setting in the organizational blackboard of a node which types of hypotheses and goals (blackboard level, interest area, belief threshholds) should be sent and to whom, and how a node should evaluate hypotheses and goals received from other nodes.

In a similar manner, we have been exploring different control relationships among nodes. For example, a node can use information (hypotheses) received from another node only to makeup for information not available from its local sensors, or it can also use received information (both hypotheses and goals) to direct its local processing. As with communication strategies, mixed initiative control strategies can be set up which balances self-directed control and externally directed control by varying the importance and credibility a node attaches to information received for use in focusing its activities.

Table 1 shows the performance results of different architectures and control/communication strategies for a simple scenario containing one vehicle track and a "ghost" track that parallels it. The stopping criteria for these experiments is the generation of the correct answer. These results are very preliminary and are for only a single scenario, but they do show the capabilities of the testbed as an experimental tool. We plan to use these capabilities to understand, through an extensive set of test cases, the relationship among organizational architectures, control/communication strategies, and environmental scenarios.

VIII. SUMMARY

We have described a high-level simulation testbed for evaluating alternative cooperative problem solving strategies applied to a flexibly parameterized task domain. The task is monitoring of moving vehicles being sensed by a set of acoustic sensors with overlapping ranges. Both vehicle and sensor characteristics are variable to permit control of the spatial distribution of ambiguity and error in the task input data. We have developed a parameterized, distributed interpretation system based on an extended Hearsay-II architecture that includes mechanisms for goal-directed control. Node configurations and communication channel characteristics can be independently varied in this simulated system. The monitoring task knowledge sources and knowledge source scheduler used in the nodes of the testbed system have resolution capabilities which can be varied independently. This permits control of the distribution of problem solving expertise across all the nodes in the system at a detailed level.

The testbed provides a means for exploring the importance and interrelationships of the following factors in cooperative distributed problem solving:

- node-node and node-sensor configurations;
- mixes of data- and goal-directed control in the system;
- distributions of uncertainty and error in the input data;
- distributions of problem solving capability in the system;
- types of communication policies used;
- communication channel characteristics.

The testbed is used by specifying each of these factors, independently, for each of a set of test cases and comparing the results of testbed simulation of each case. The multiple dimensions of independent control and the detailed level of simulation in the testbed provide what we feel is a very useful vehicle for exploring high-level issues in cooperative distributed problem solving.

ACKNOWLEDGEMENTS

We would like to acknowledge contributions to the construction of the testbed by Ram Mukunda, Peter Bates, Robert Markey, and Sheryl Franklin.

REFERENCES

BATE81 Peter C. Bates, Jack C. Wileden, and Victor R. Lesser.
A language to support debugging in distributed systems.
Technical Report 81-7, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 01003, February 1981.

CORK80 Daniel D. Corkill.
An organizational approach to planning in distributed problem solving systems.
COINS Technical Report 80-13, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 01003, May 1980.

CORK81 Daniel D. Corkill and Victor R. Lesser.
A goal-directed Hearsay-II architecture: Unifying data-directed and goal-directed control.
COINS Technical Report 81-15, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 01003, June 1981.

CORK82a Daniel D. Corkill, Victor R. Lesser, and Eva Hudlicka.
Unifying data-directed and goal-directed control: An example and experiments.
To appear in Proceedings of the Second National Conference on Artificial Intelligence, August 1982.

CORK82b Daniel D. Corkill.
A Framework for Organizational Self-Design in Distributed Problem Solving Networks.
PhD Thesis, Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, September 1982.

ERMA80 Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy.
The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty.
Computing Surveys 12(2):213-253, June 1980.

FOX79 Mark S. Fox.
Organization structuring: Designing large complex software.
Technical report CMU-CS-79-155, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, December 1979.

KAHN78 R. E. Kahn, S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman.
Advances in packet radio technology.
Proceedings of the IEEE 66(11):1468-1496, November 1978.

LACO78 R. Lacoss and R. Walton.
Strawman design of a DSN to detect and track low flying aircraft.
In Proceedings of the Distributed Sensor Nets Workshop, pages 41-52, December 1978.

LESS80a Victor R. Lesser and Lee D. Erman.
An experiment in distributed interpretation.
IEEE Transactions on Computers C-29(12):1144-1163, December 1980.

LESS80b Victor R. Lesser, Jasmina Pavlin, and Scott Reed.
Quantifying and simulating the behavior of knowledge-based interpretation systems.
In Proceedings of the First Annual National Conference on Artificial Intelligence, pages 111-115, August 1980.

LESS81a Victor R. Lesser and Daniel D. Corkill.
Functionally accurate, cooperative distributed systems.
IEEE Transactions on Systems, Man, and Cybernetics SMC-11(1):81-96, January 1981.

LESS81b Victor Lesser, Daniel Corkill, Jasmina Pavlin, Larry Lefkowitz, Eva Hudlicka, Richard Brooks, and Scott Reed.
A high-level simulation testbed for cooperative distributed problem solving.
Technical Report 81-16, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 01003, June 1981.

SMIT80 Reid G. Smith.
The contract net protocol: High-level communication and control in a distributed problem solver.
IEEE Transactions on Computers C-29(12):1104-1113, December 1980.

SMIT81 Reid G. Smith and Randall Davis.
Frameworks for cooperation in distributed problem solving.
IEEE Transactions on Systems, Man, and Cybernetics SMC-11(1):61-70, January 1981.

SYSTEM CONFIGURATION	COMMUNICATION: VOLUNTARY/ EXTERNAL		CONTROL: SELF/ EXTERNAL		CYCLES TO CORRECT ANSWER	HYPS SENT	GOALS SENT
1-node					61	-	-
4-node	V		S		49	43	-
	V			E	55	27	-
	V		S	E	40	32	-
		E		E	50	17	-
5-node	V		S		36	20	-
	V	E	S	E	38	22	20
	V	E		E	49	36	30
	V	E	S		34	18	16

TABLE 1: Experimental Results on a Simple Scenario

In the 4- and 5-node system configurations, one of the nodes only senses part of the false ghost track. This node quickly generates a high-level interpretation of moderate belief for this false data and transmits it. In the control regimes that emphasize externally-directed control, this early transmission of a high-level false hypothesis distracts the receiving nodes from working on their own low-level data which contains a mixture of both correct and false data. This delays the generation of the correct interpretation.

These preliminary results show the need for a balance between self-directed and externally-directed control.