COOPERATIVE DISTRIBUTED PROBLEM SOLVING:

A New Approach for Structuring Distributed Systems

Victor R. Lesser and Daniel D. Corkill

COINS Technical Report 78-7
May 1978

ABSTRACT

A new paradigm, called cooperative distributed problem solving, is proposed for use in distributed processing systems. This paradigm extends the range of distributed processing applications and permits systems to function with both inconsistent and incomplete data.

In cooperative distributed systems, the only effective way to distribute the basic problem solving task is to decompose the problem solving algorithms and control structures in such a way that they operate on local data bases which are incomplete as compared to a centralized formulation. In order to resolve the uncertainties in these incomplete local data bases, processors must exchange information with other processors. Since this information may be based on processing which used incomplete data, an iterative, coroutine type of processor interaction is required. This type of processor interaction is incompatible with the master-slave control structures used in existing systems where processing goals are both local and disjoint.

Recent developments in Artificial Intelligence provide a basis for developing this paradigm. Preliminary work in traffic control and multi-sensor interpretation indicates the potential of this approach.

1. INTRODUCTION

A new paradigm, called cooperative distributed problem solving, is proposed for structuring distributed processing systems. This paradigm extends the range of distributed processing applications and permits systems to function with both inconsistent and incomplete data. Distributed processing is defined as computation which is decomposed into subtasks that are executed on relatively independent processing elements. We differentiate distributed processing from multiprocessing by defining distributed processing systems as those systems where interprocessor communication costs prohibit the maintenance of a centralized global view of the problem solving data base and control structure.

Recent developments in microprocessor technology [Noyce 1976] and network technology [Cerf and Kahn 1974, Kimbleton and Schneider 1975] have lowered the cost of processors and communication to a level where distributed processing is now practical. The potential advantages of a distributed processing approach over a centralized approach include:

increased reliability and flexibility -- achieved through redundancy in communication paths and processing elements and through modularity of design (which permits incremental addition of new processors and communication paths);

enhanced real-time response -- achieved through parallelism and through placing processors near sensing devices and devices to be controlled;

lower communication costs -- achieved by abstracting (preprocessing) data for transmission (lowering communication

bandwidth requirements) and by reducing the distance data must be transmitted by placing processing elements near the data;

lower processing costs -- achieved through use of cheaper, less complex processing elements which can be mass produced and by load sharing (allowing relatively idle processing elements to handle some of the work of a busy processing element); and

the ability to handle increased complexity -- achieved by decomposition of the problem solving task into subtasks, each reduced in the range of possible inputs as compared with the overall task. The result of this decomposition is the creation of a process structure in which each processing element has only a portion of the total system task.

These potential advantages have not yet been exploited in a wide range of application areas. Only in the areas of process control [Hoffman 1975, Dimmler et al 1976, Schoeffl and Rose 1976], distributed data bases [Peebles and Manning 1975, Deppe and Fry 1976, Manning and Peebles 1977], and computer networks [Karp 1973, Pyke and Blanc 1973, Crowther et al 1975] have some of the promises of distributed processing been realized [Infotech State of the Art Report 1976].

There are additional applications, which we will term "cooperative" distributed problem solving applications, that cannot be realized by using either a centralized problem solving approach or the distributed techniques developed to date. Many of these applications occur in situations that have a natural spatial distribution (e.g., where sensors for collecting raw data are widely distributed and/or mobile). In order to perform appropriate actions in these applications, the development and execution of a distributed control strategy based on a global interpretation of sensory data is required. In these

applications; a distributed architecture which locates processing capability at the sensor site is especially advantageous and perhaps is the only practical way to solve the problem. Air traffic control, network (automotive) traffic control, inventory control (e.g., car rentals), power network grids, sensor networks, and tasks involving mobile robots are all examples of potential cooperative distributed applications. We feel these types of applications can take full advantage of the benefits of distributed computation.

No software technology for building cooperative distributed problem solving systems has yet been developed. Current methodologies for distributed applications, such as distributed data bases, are not directly applicable to cooperative applications due to the relatively complex interaction required for cooperative tasks. These methodologies also fail to deal with some issues crucial to cooperative distributed computation, such as working with incomplete or inconsistent data. We feel recent developments in Artificial Intelligence (AI) arising from work on large "knowledge-based" systems provide a basis for developing new, more appropriate methodologies.

The remainder of this paper is divided into four sections. Section 2 differentiates the class of cooperative distributed problem solving systems from existing distributed systems, provides an example of a cooperative system (a demand bus scheduling system), discusses design questions for cooperative

distributed systems, and explains why existing distributed processing techniques are not sufficient for these systems.

Section 3 describes our approach to the development of appropriate techniques for the design of cooperative distributed systems, based on work in AI and on the structure of natural systems.

Section 4 presents our preliminary research in two areas relating to cooperative distributed systems: a distributed version of the Hearsay-II architecture (as applied to the multi-sensor interpretation problem) and an approach to distributed network traffic light control.

Section 5 describes a vehicle, called the Distributed Processing Game, for developing new cooperative distributed methodologies.

2. COOPERATIVE DISTRIBUTED SYSTEMS

2.1. Nearly-Autonomous versus Cooperative Distributed Systems

Distributed systems can be grouped into two broad classes based on processor (process) interaction patterns:

nearly-autonomous distributed systems and cooperative distributed systems.

Nearly-autonomous distributed systems display processor interaction in which one processor rarely needs the assistance of another processor in carrying out its problem solving function. Each processor has a local data base and algorithms appropriate for carrying out its desired task; there is little need for dialogue between the initiator of the task and the called processor in order to complete the task. When viewed from this perspective, processor interactions look like subroutines which can be executed asynchronously (i.e., there is a master-slave relationship between the initiator and the called processor) [Anderson 1965, Gosden 1966, Constantine 1968].

Most existing distributed systems can be classified as nearly-autonomous. This is largely due to the fact that the emphasis in nearly-autonomous systems is on distributing the problem solving data base. The basic problem solving algorithms and control structures which are appropriate in a non-distributed environment do not need to be modified (decomposed) for the distributed environment, but rather are replicated or partitioned based on the distribution of the problem solving data base. Where more complex control structures have been introduced in these distributed systems, their use has been directed at maintaining data base consistency (synchronization, deadlock detection and avoidance) and error recovery. These distributed

control structures are superimposed on the basic computational algorithms needed to perform the application processing [Eswaran et al 1976, Thomas 1976, Lampson and Sturgis 1977, Peebles 1977].

The use of a nearly-autonomous system structure is appropriate where the algorithms needed to perform the task can be compartmentalized to minimize interaction between processing elements.

In cooperative distributed systems, additional complexity over the nearly-autonomous structure is required in order to perform the basic problem solving task. This additional complexity arises because the only effective way to distribute the task is to modify (decompose) the problem solving algorithms and control structures in such a way that they operate on local data bases that are incomplete (i.e., contain a subset of the data required in the centralized formulation of the algorithms and control structures). In order to resolve the uncertainties in these incomplete local data bases, processors must exchange information with other processors. Since this information may be based on processing which used incomplete data, an iterative, coroutine type of processor interaction is required in which processing elements must communicate partial results among themselves in order to develop a satisfactory solution. If a processing element does not receive an appropriate partial result in a given amount of time, it must be able to continue processing utilizing whatever data are available at that time. A side

effect of this type of interaction is that local problem solving tasks in the cooperative decomposition are not disjoint as in the nearly-autonomous structure. Rather the tasks overlap in either the data they need or produce, forming a "cooperative network of tasks" which collectively define the processing needed to solve the global task.

Another effect of non-localized processing in cooperative systems is that reliability and flexibility issues cannot be addressed solely at the hardware and communication level (syntactically) [1], but must also be dealt with as an integral part of the basic problem solving process (semantically). In a nearly-autonomous system, it is easier to detect, isolate, and recover from an error since propagation of the error can be more easily determined (due to the simpler interaction structure) and appropriate recomputation performed. This is in contrast with cooperative systems where incorrect partial results might be propagated extensively through the system before detection if not corrected as part of the normal problem solving process.

---

[1] Work has also been performed on developing communication protocols that are robust in the face of communication and site failures [Cerf and Kahn 1974, Kimbleton and Schneider 1975, Tajibnapis 1976] and on embedding message passing protocols into existing programming languages and operating systems [Farber et al 1973, Feldman 1977, Fosdick et al 1977, Jefferson 1977]. When viewed from the perspective of the routing task alone, some algorithms used to determine message paths in a communication network have a cooperative problem solving character to them [Gerla 1973, Tajibnapis 1977].

2.2. An Example of a Cooperative Distributed System

The following example illustrates an application which can be structured as a cooperative distributed system and which highlights the design issues in constructing such systems.

Consider a demand bus system where a fleet of buses is to serve an urban area. Upon arrival at a bus stop, a customer might dial his desired destination on a selector device and this information would be used to plan bus routing dynamically. There are a number of elements which must be considered in such a system: buses should be kept reasonably full but not overloaded; the total mileage of the fleet should be kept as low as possible; the mean service time (waiting time at the bus stop and riding time) should be kept small; the maximum service time of any one customer should not exceed a reasonable amount (a customer wanting to get to Fifth and Main should not have to ride around all day merely because no one else needs to go near that location); and the system should be able to monitor and respond to special events (e.g., different traffic patterns at different times of the day, concerts, athletic events, local weather conditions, bus breakdowns, stalled traffic, etc.).

A cooperative distributed approach to this problem would place a microprocessor and a packet radio communication device [Kahn 1975] of limited range on each bus. Each bus stop (or cluster of stops) would have similar packet radio capability.

When a customer dials his desired destination this information would be broadcast to the nearby buses and stops.

A major aspect of designing a cooperative distributed system for the bus application is determining a proper decomposition of the overall scheduling task into subtasks for processing elements. The decomposition should minimize the communication requirements of the system, both in communication distance and in total bandwidth requirements. Processors in close proximity should communicate more among themselves than with more distant processors. The problem decomposition should also limit the complexity of any given subtask to a level which can be handled by its assigned processor. The decomposition should allow a suitable level of system performance and should exhibit a high degree of reliability (the loss of any individual processing element should not greatly degrade the system or leave a large number of waiting customers stranded).

In order to schedule buses effectively, some aspects of a non-local view must be developed in the cooperative system. Decisions based only upon local views would not be effective in handling overall changes in the service pattern caused by concerts, athletic events, a special sale at a shopping center, bus breakdowns, and similar situations where the desired service cannot be handled on a statistical basis.

Non-local view requirements for the bus problem must be determined. For example, it might be sufficient to partition the city into a number of static regions with non-local views consisting of bus occupancy, number of customers awaiting service, number of buses within each region and estimates on bus and rider transitions between adjacent regions in the near future. On the other hand, the planned routes of individual buses and riders may also be needed for effective scheduling.

No matter how much non-local information is required by buses, the cooperative system must be able to aggregate these views from individual local views that are possibly inconsistent and/or errorful in order to determine a satisfactory route for each bus.

Implicit in the scheduling task is the ability of the system to develop and execute plans. As each service request enters the system, a plan for picking up the customer at the bus stop and for depositing him at his desired destination must be developed. This plan would be based on the current and projected status of the system and should be developed and executed in a cooperative distributed fashion.

Due to the likelihood of bus breakdowns, stalled traffic, unplanned-for customers, customers entering a request for service and then choosing not to ride the bus, etc., the bus system would need a way to monitor its current status in a distributed

fashion. The system would also have to determine in a distributed way whether or not to replan in the event of a major difference between its current status and its previously projected status.

2.3. Design Questions for Cooperative Distributed Systems

As illustrated by the bus scheduling example, a number of issues must be resolved in the design of a cooperative distributed system

problem decomposition -- How should the overall problem solving task be broken into subtasks to: minimize communication requirements (between cooperating subtasks and in obtaining a global view), limit the complexity of any given subtask, and increase the reliability and performance of the overall system? Should the decomposition be static or evolved dynamically based upon the current status of the system?

obtaining a global view -- What aspects of the system need to be viewed globally? What levels of abstraction are approprate for representing this view? How can local views which are potentially inconsistent and/or errorful be aggregated to form a global view? How should the global view be held--should a single copy be distributed throughout the system or should each processing element have the portion of the view which it requires? Do multiple copies of a global view need to be completely consistent, or can the system perform without complete consistency assumptions? How can dynamically changing data be represented?

planning and plan execution -- How can planning be done in a cooperative distributed fashion? How can this plan be executed in a cooperative fashion? What degree of synchronization between processing elements is necessary for both planning and plan execution in a given application? To what degree can errors in synchronization be handled automatically by the same resolution techniques needed to resolve uncertainty in the problem domain itself?

monitoring and plan modification -- How can the system monitor in a distributed way how well it is achieving its goals? How can the system modify its current plan in the event of an unexpected change in the environment or within the system itself? How can

the system decide whether to modify its existing plan or generate a new plan based on the cost/benefits estimated for each?

reliability and flexibility -- How robust is the system in the presence of certain types of malfunction? Is there the possibility that the system can get into a severly degraded state due to the failure of a single component [Dijkstra 1974, Merlin 1975]? What type of communication bandwith requirements are necessary to support the cooperative problem solving? How fast need individual processing elements be? How does redundancy in processing and communication capacity affect the robustness of the system? How can error analysis be performed in the system?

Most of these questions have yet to be resolved, both regarding specific applications and for general classes of applications.

3. OUR APPROACH TO THE DESIGN OF COOPERATIVE DISTRIBUTED SYSTEMS

3.1. Artificial Intelligence and Cooperative Distributed Systems

In the last several years, there has been a trend in AI research to develop high-performance systems specialized for particular problem domains (e.g., medical diagnosis [Shortliffe 1974], chemical analysis [Feigenbaum et al 1971], image understanding [Riseman and Arbib 1977], and speech understanding [Reddy 1976]). In order to attain the high degree of performance desired, these systems require a large amount of problem-specific knowledge and, often, a large number of heuristics. Such systems

are commonly called "knowledge-based" systems, as differentiated from many of the earlier AI systems which used a small number of general heuristics and little problem-oriented knowledge.

Problem solving in knowledge-based systems is not merely the retrieval of facts from a large data base, but rather an incremental aggregation of partial solutions into an overall solution. These partial solutions arise from both the application of diverse sources of knowledge to the same aspects of the problem and from the application of the same knowledge to different aspects of the problem. As will be seen by the following examples, the AI paradigms developed for problem solving in knowledge-based systems, especially those that deal with errorful input data, provide models for problem solving in cooperative distributed systems.

Let us consider two systems, Hearsay-II [Erman and Lesser 1975] and MSYS [Barrow and Tenenbaum 1976], that exhibit this cooperative type of problem solving strategy. While Hearsay-II and MSYS were developed for speech understanding and vision understanding respectively, their basic structures have general applicability and have been applied to such tasks as: multi-sensor interpretation [Nii and Feigenbaum 1977] and protein-crystallographic analysis [Englemore and Nii 1977].

In Hearsay-II, as applied to speech understanding, the understanding of spoken utterances is accomplished by combining

partial solutions derived from acoustic, phonetic, syllabic, lexical, syntactic, and semantic knowledge applied to different portions of the utterances; each area of knowledge is encapsulated in an independent module ("knowledge source"). The interaction is based on a data-directed form of the hypothesize-and-test paradigm. In this paradigm, solution-finding is viewed as an iterative process. Each step in the iteration involves the creation of an hypothesis, which is an "educated guess" about some aspect of the problem, followed by a test of the plausibility of the hypothesis. Both of these steps use a priori knowledge about the problem (in this case, contained in the knowledge sources), as well as previously generated hypotheses. The guess-building terminates when a consistent hypothesis is generated which satisfies the requirements of an overall solution.

In MSYS, as applied to vision understanding, each knowledge source processes a portion of the data in terms of its own limited knowledge. Each knowledge source attempts to explain what object(s) could potentially occur in a specific part of a segmented image. The consensus is achieved by a network of processes (representing independent knowledge sources) that communicate via shared global variables. Each process attempts to explain a fragment of the data (a region or a few regions in a segmented scene) in terms of its own limited knowledge. The confidence of an explanation is communicated to other processes attempting to explain overlapping fragments, and may cause them

to reevaluate their own hypotheses. The confidence adjustment cycle continues until equilibrium is achieved" [Barrow and Tenenbaum 1976, p. 3]. When this equilibrium is achieved, a coherent set of local views has been constructed.

The MSYS problem solving technique is an example of a more general problem solving paradigm, "relaxation", that is contained in different forms in many types of problem solving systems. The term "relaxation" originates from successive refinement methods used to solve partial differential equations.

Similar uses of the aggregation of partial solutions arise in systems using the locus model [Rubin and Reddy 1977], and the cooperating experts paradigms [Arbib 1975, Lenat 1975, 1976] [2].

We feel these AI paradigms, especially hypothesize-and-test and relaxation, provide a basis for the development of methodologies for cooperative distributed systems. If viewed as modules operating on different portions of the data base using different sources of knowledge, cooperative distributed systems appear very similar to many knowledge-based problem solving

---

[2] The contract net model [Smith and Davis 1978] for distributed processing is based on the work by Lenat [1975] on the cooperating experts paradigm. However, the contract net takes a nearly-autonomous view of distributed processing in that interaction (contracts) exhibit an asynchronous subroutine call structure. The ACTORS system developed by Hewitt [1974, 1975] prior to the work of Lenat is also based on the society of cooperating experts paradigm. However, it as yet does not deal with issues crucial to cooperative distributed computation such as incomplete or inconsistent data and multiple instances of data and their migration among processing nodes [Svobodova 1977].

systems. As will be presented in the next section on preliminary research, the Hearsay-II architecture has been applied to interpreting in a distributed manner data originating from spatially separated sensors, and the relaxation paradigm has been applied to distributed network traffic control.

These AI paradigms are also appropriate because they are robust in their handling of errors from input data and errors from incomplete or incorrect knowledge. In these paradigms, errors are automatically handled as an integral part of the problem solving process. This is in contrast with handling errors as exceptional conditions, where additional processing outside the normal problem solving strategy is invoked. A side benefit of the ability to handle errors within the problem solution itself is the possibility that these systems will automatically correct for errors caused by hardware malfunction, by inconsistencies in the data base, and by partial views of the global data base. To the degree these paradigms can self-correct such errors, the need for a complete and consistent view of the global data base and for explicit synchronization is reduced. Preliminary work in testing this hypothesis, with respect to synchronization, in multiprocessor implementations of both the Hearsay-II paradigm as applied to speech understanding [Fennell and Lesser 1977] and the relaxation paradigm as applied to partial differential equations [Baudet et al 1977], has been encouraging.

While AI paradigms provide a good beginning, they have not completely distributed entire systems. Centralized global knowledge or global control has been used in these AI systems to coordinate various system modules. For example, the Hearsay-II paradigm relies on a centralized global data base (called the "blackboard") for the integration of local views generated by independent knowledge source modules and for communication of these views to other knowledge sources. Scheduling (planning) is also centralized, based on the current hypotheses on the blackboard and a global agenda mechanism. Relaxation relies on either synchronization (lock-step iteration) or an explicit ordering relationship between modules in order to speed up or guarantee convergence. In addition, relaxation does not generate a specific global solution, only a set of consistent local solutions. The locus model paradigm also requires explicit synchronization between processing elements and a centralized view for scheduling. The cooperating experts paradigm, in Lenat's formulation, relies on direct communication capability between any two processing elements and the existence of a global scheduler. However, it is important to reiterate that even though current formulations of these AI paradigms are not totally distributed, their ability to function with incomplete and incorrect knowledge make them easily adaptable to distributed situations in which only partial and potentially inconsistent views of the global data base are available. The ease of this adaptability is demonstrated in the reorganization of the

Hearsay-II architecture for the distributed multi-sensor environment (see Section 4.1).

In performing cooperative distributed problem solving there is also a need for distributed planning, plan execution, monitoring, and plan modification. AI techniques for developing non-linear plans appropriate for distributed execution exist [Sacerdoti 1974, 1975, 1977b; Tate 1975, 1976a, 1976b, 1977], but these techniques generate plans in a centralized fashion. Initial work on distributed planning has involved distribution of Sacerdoti's NOAH system by assigning conjunctive subgoals to different processors. Locally planned actions which might interact with the achievement of other subgoals are transmitted to the relevant processors. Problems with this approach include deciding which locally planned actions might interact with other processors and choosing a decomposition of the planning task which yields efficiency in the planning task itself and/or in the overall solution. A forthcoming paper [Corkill 1978] discusses distributed planning and its associated problems in greater detail.

There has also been work performed on generating distributed plans in a distributed framework using strict hierarchical structures that are nearly-autonomous [Fikes and Pease 1975]. This structure, however, is not directly applicable to cooperative distributed processing and makes plan modification very difficult without the existence of a centralized view.

While AI techniques for distributed planning are not as applicable as those for distributed interpretation, they do form a useful base for the development of new planning paradigms for cooperative distributed systems.

3.2. Natural Systems and Cooperative Distributed Systems

Natural systems also provide a set of paradigms which may be appropriate as a base for developing cooperative distributed systems. Management organizations [Simon 1962, 1969; Galbraith 1973, Fox 1977], social organizations in animals [Reed 1978], and the brain [Kilmer, McCullock, and Blum 1969, Arbib 1975, 1978] are all examples of systems which exhibit cooperative distributed problem solving. Each of these examples consists of a set of separate processing elements equipped with local control which communicate with other elements to obtain a solution to a complex task. This communication does not, in general, take the form of asynchronous subroutine calls, but rather has the form of a cooperative dialogue.

Because of the close relationship between these natural systems and distributed systems, we feel there is a very good chance that distributed control paradigms used in these natural systems will be relevant to the search for cooperative distributed system methodologies.

4. PRELIMINARY RESEARCH

Our preliminary research has attempted to demonstrate that AI does provide a basis for the development of methodologies appropriate for cooperative distributed systems. Our research strategy has been to conduct two pilot studies: distributed multi-sensor interpretation and distributed traffic light control.

4.1 A Distributed Architecture for Multi-sensor Interpretation

The first pilot study involved taking an existing AI problem solving model which is currently specified in a partially distributed way and reworking both its data and control structures to be completely distributed. The study concentrated on applying the Hearsay-II architecture to the multi-sensor interpretation problem, where each processor is mobile, has a set of (possibly non-uniform) sensing devices, and interacts with nearby processors through a packet-radio communication network. It is desired that processors communicate among themselves to generate a maximally consistent interpretation of "what is happening" in the environment being sensed. A hypothetical example of a multi-sensor application is storm reporting. Processors would be placed in weather reconnaisance airplanes equipped with sensors for measuring various atmospheric

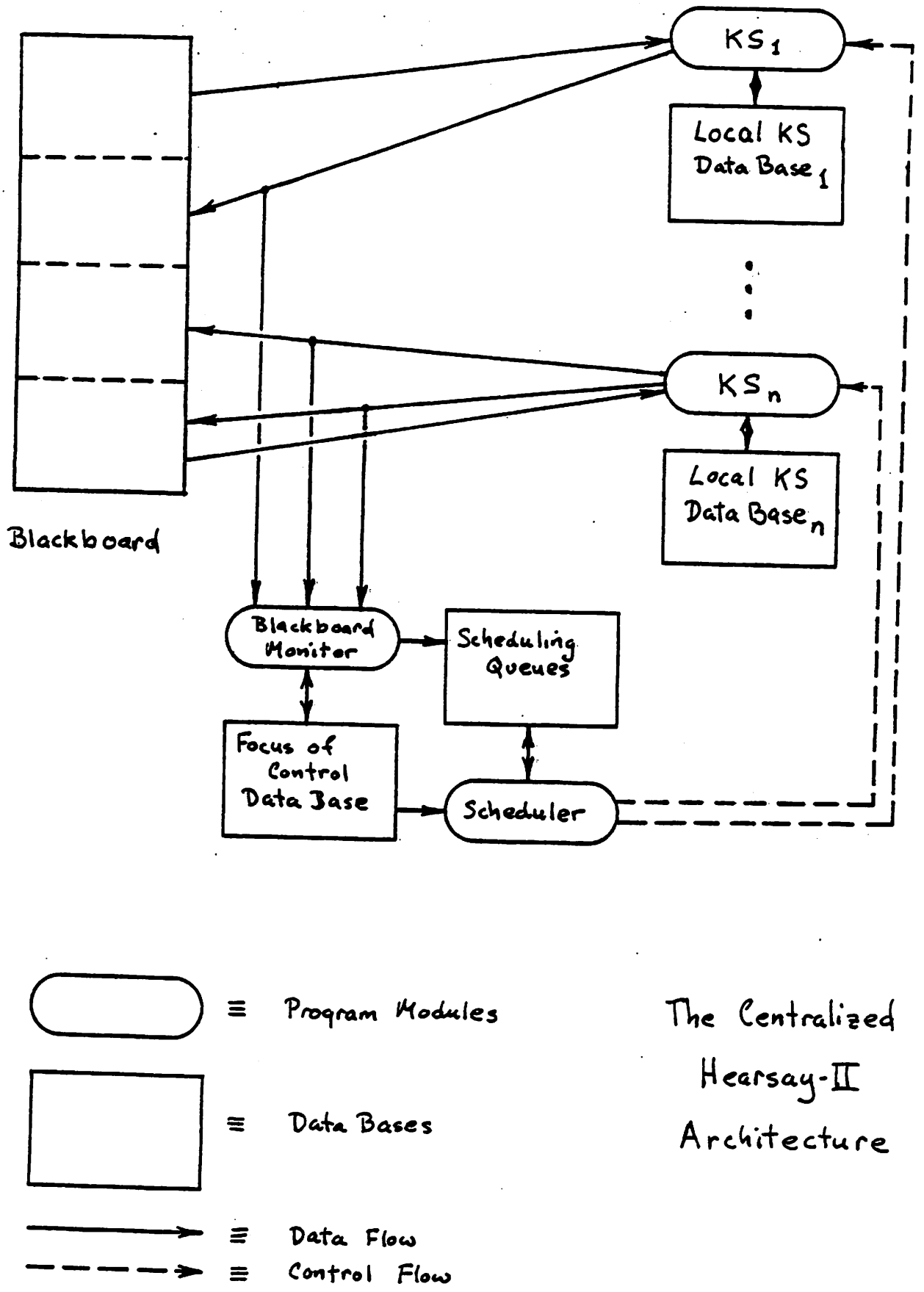
conditions (air pressure, wind direction and speed, etc.). The processors onboard these planes would communicate with each other to assess the direction, size, and severity of the storm, and to determine what further information must be acquired in order to verify the current assessment.

Overview of Hearsay-II

The Hearsay-II architecture is designed to permit cooperative and competitive problem-solving among a diverse set of knowledge sources (KSs). This architecture appears suitable for implementing the cooperation necessary to gain a consistent interpretaton of the environment from multiple sensors. What is most interesting about this architecture is that the basic problem solving strategy can be easily adapted to the multi-sensor interpretation problem. In fact, a centralized version of the Hearsay-II architecture has already been applied to a similar task [Nii and Fiegenbaum 1977].

A highly simplified schematic of the centralized Hearsay-II architecture is contained in Figure 1. The major data structures are the blackboard (shared global data base), focus of control data base, scheduling queues, and local KS data bases.

The blackboard is partitioned into distinct information levels, each used to hold a different representation of the



The Centralized Hearsay-II Architecture

figure 1

problem space. The major units on the blackboard are hypotheses. A hypothesis is a guess about a solution to a portion of the problem in terms of a particular problem space representation. Hypotheses at different levels in the blackboard are connected through an and/or directed graph structure.

Decomposition of the problem space into levels on the blackboard is a natural parallel to decomposition of the knowledge into separate KSs. Most KSs need to deal with only a few (usually two) levels in order to apply their knowledge. Processing is organized in terms of the incremental addition of small units of information by each KS. Every small increment of information should help to verify, refute, or augment (expand) a hypothesis (i.e., the hypothesize-and-test paradigm).

KS activity is managed by the scheduler using the focus of control data base and the scheduling queues. At any point, the scheduling queues contain a number of pending KS activations. The scheduler calculates a priority for each waiting KS and selects the KS with the highest priority for execution. The priority calculation attempts to estimate the usefulness of the KS's action in fulfilling the overall system goal. This estimation is based on the type of action the KS will perform and on the state of processing as indicated in the focus of control data base.

The focus of control data base contains information indicating the current best hypotheses on the blackboard, how much time has elapsed since these hypotheses were generated, and threshold values indicating the number and type of hypotheses currently desired from each KS. This information is updated by the blackboard monitor based on the generation and modification of hypotheses on the blackboard.

The blackboard monitor is also used to implement the data-directed activation of KSs. When a subset of hypotheses appropriate for action by a particular KS is recognized by the blackboard monitor, an activation record for that KS on those hypotheses is placed in the scheduling queues.

Issues in Distributing Hearsay-II

From this brief overview, it can be seen that many of the characteristics of this organization for knowledge-based problem solving make it potentially suitable for a distributed processing environment:

(1) Knowledge is grouped into KS modules which are largely independent, anonymous, and capable of asynchronous execution.

(2) KS activation is based on the generation and modification of hypotheses on the blackboard (data-directed control). To the extent that these hypotheses can be distributed, control of KS activation can also be distributed. The data-directed form of the hypothesize-and-test paradigm permits KSs to exchange partial results in a cooperative fashion.

(3) The blackboard data base is represented as a set of distinct information levels, and KSs generally access only a small number of these levels.

These aspects of the Hearsay-II organization have been exploited to achieve significant parallelism (a speedup factor of six with sixteen processors) in a multiprocessor implementation with shared common memory [Fennell and Lesser 1977]. This degree of parallelism was achieved using explicit synchronization techniques to maintain data integrity. However, the basic self-correcting nature of information flow in the Hearsay-II problem solving paradigm obviates the need for much of the explicit synchronization. In simulations where the unnecessary explicit synchronization was eliminated, the parallelism factor rose to fifteen with thirty-two processors.

There seems to be some difficulty, however, in efficiently implementing the Hearsay-II architecture in a distributed system with no shared common memory and in which KS executions do not all occur on the same processing element. These difficulties arise because:

(1) The scheduler, which requires a global view of the pending KS instantiations (scheduling queues) and the focus of control data base, is centralized.

(2) The blackboard monitor, which updates the focus of control data base and scheduling queues when a specific type of blackboard change occurs, is centralized.

(3) The patterns of KS access to the blackboard overlap, prohibiting the construction of compartmentalized clusters of levels accessed exclusively by small groups of KSs.

Without a shared common memory, an extensive amount of interprocessor communication is required to maintain a centralized view of the blackboard, scheduling queues, and focus of control data base, and to transfer information among KSs to support cooperation. It is these data structures and information (which control the degree and nature of KS cooperation) that are essential to the effective implementation of the hypothesize-and-test problem solving strategy. The key issue in distributing Hearsay-II is how to decompose these data structures to minimize communication costs and still permit the appropriate type of KS cooperation necessary to generate an interpretation distributively.

A Distributed Version of Hearsay-II

Our approach to decomposing Hearsay-II in the packet-radio network environment is based on the following premise: the only cost-effective way to do the decomposition is to modify the Hearsay-II problem-solving strategy to work with processors operating on partial and possibly inconsistent views of the current interpretations and system state.

There are two major ideas necessary to implement the distributed version of the Hearsay-II architecture. The first idea comes from the relaxation paradigm. In this paradigm, the current state (results) of a processor node is accessed only by nodes that are immediately adjacent in the node network. If the current state of the node is 'important' or 'relevant', the state

will be transmitted gradually to other areas of the node network, resulting in what can be thought of as a spreading excitation of important information. This same approach may be used to decentralize the blackboard. Not all processor nodes have to receive immediately all information that is pertinent to them; if the information is really important, it will eventually be spread through the network to all nodes.

The second major idea comes from viewing a processor node as a generator function which can be successively re-invoked as needed to generate additional (and possibly less credible) hypotheses for a particular portion of the interpretation. Generator function re-invokation is achieved through the use of threshold values (part of the focus of control data base) which are modified by the same relaxation scheme used to transmit important information.

Using these two ideas, we have developed a system which permits effective KS cooperation without the high communication bandwidth required to support a completely centralized data base and controller. This lower bandwidth requirement is accomplished by transmitting only a limited subset of the results generated at each node to only a small number of other processing nodes. The lack of a centralized data base may result in KS processes generating incomplete and/or inconsistent partial interpretations. However, use of the hypothesize-and-test paradigm for KS cooperation, together with the relaxation of

information among local nodes, should resolve these conflicting views as part of the normal problem solving process.

The system is structured in the following fashion:

KS processing is distributed by statically allocating to each processor a set of KSs based on the type of sensors attached to the processor. Locating KSs at the site of the sensors they use as input minimizes communication costs. Additional KSs, which do further processing on the output of KSs directly operating on sensor data could also be allocated to the processor if desired. Based on the type of processing being done, it may also be reasonable to include other KSs, or even all KSs.

The blackboard and focus of control data base is distributed by giving each processor a local copy of the blackboard with a subset of the information levels and a local copy of the focus of control data base. The information levels included are those used by the KSs allocated to the processor. These local copies reflect the results of both local processing and data received from other processors. The scheduler and global scheduling queues are also distributed by having each processor maintain scheduling queues for its (local) KS activations.

The blackboard monitor is distributed by establishing a dynamic communication network based on the current position of the mobile processors and the input/output characteristics of their associated KSs.

The dynamic communication network is re-formed whenever a mobile processor node moves so that the nodes in its local broadcast range change. When this occurs, the incoming processor announces its name and the input/output characteristics of its KSs. In addition, it transmits its focus of control data base (which indicates its current view of the state of system processing). The processors which receive this message in turn acknowledge with their input/output characteristics and focus of control data. As a result of this dialogue, a local connectivity pattern based upon matches between input/output characteristics

is established among appropriate KSs in the local network. In addition, the focus of control data bases of these newly connected nodes have been mersed with one another.

If the connectivity pattern does not have the appropriate richness in terms of the number of processors generating input data and using output data [3], the processor may ask its neighboring processors to act as repeaters, re-broadcasting its input/output characteristics to their neighboring processors. In this case, these neighboring processors act as store/forward message processors, supporting the desired connectivity pattern (see Figure 2).

Once communication paths are established, the processor exchanges the current best hypotheses with its communicating processors. From then on, the processor will transmit and receive only better hypotheses or modifications to previously transmitted hypotheses. The local focus of control data bases specify the criteria (thresholds) for what are good hypotheses to transmit. In this way, we have been able to design a distributed interpretation system which does not require a centralized data base or controller.

This discussion represents a high level overview of the decomposition, ignoring such important issues as reliability,

---

[3] The richness of the interconnection structure among processing nodes is a modifiable property based on the desired reliability and convergence properties of the system.

The Distributed Hearsay II Architecture

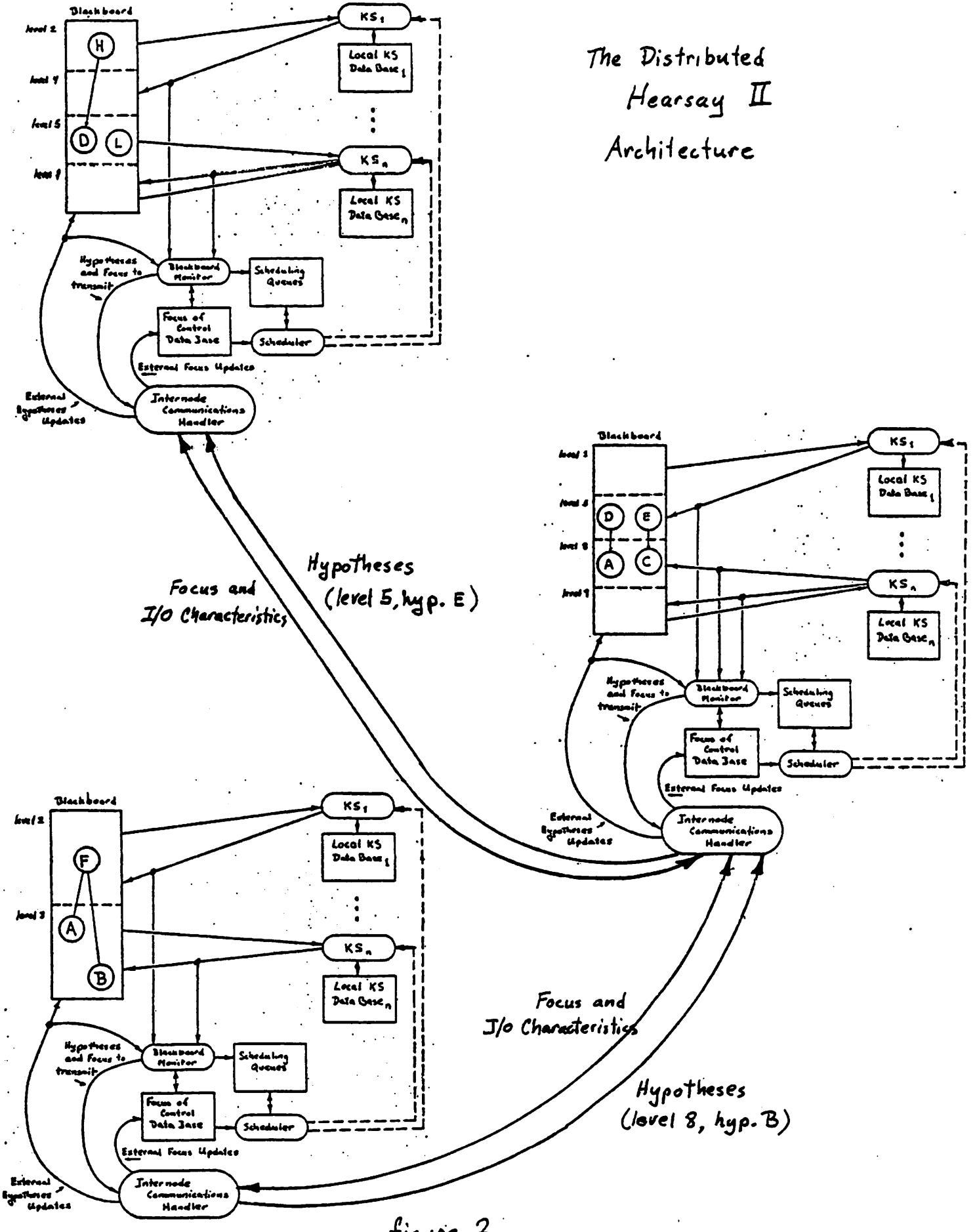


figure 2

resolution of inconsistency among local blackboards, etc. A more detailed presentation together with simulation data indicating how this reorganization of the Hearsay-II architecture affects its problem-solving behavior, its robustness in the face of errorful data (resulting from hardware malfunction, sensing, and processing) is presented in a forthcoming paper [Lesser and Erman 1978]. The simulation results described in that paper indicate that a simplified version of the distributed Hearsay-II architecture as applied to speech data (i.e., different nodes, each with a complete set of KSs, processing overlapping segments of the acoustic data) performs as well as the centralized system. The distributed system produces these results using a low degree of interprocessor communication (i.e., fewer than 50% of the locally generated hypotheses need to be transmitted). In addition, the distributed system continues to function, although in a somewhat degraded manner (i.e., lower recognition accuracy), with a noisy interprocessor communication channel (a randomly selected 25% of transmitted messages lost).

The major point of this study was to demonstrate that the cooperative nature of problem solving in the Hearsay-II architecture does permit the architecture to be reorganized for a distributed environment without significant difficulty.

4.2 Distributed Traffic Light Control

The second pilot study involved applying a cooperative distributed approach to network traffic control [4]. Network traffic light control is the problem of controlling signals so as to optimize traffic flow in a network of intersections [Wagner et al 1971, Lieberman et al 1974a, 1974b, DOT/FHA 1976]. This optimization is usually accomplished by maximizing traffic flow on major arteries using appropriate traffic signal settings which minimize delay and stops incurred by vehicles traversing the arteries.

This study concentrated on investigating the suitability of parallel relaxation as a methodology for distributed problem solving. Current experience with parallel relaxation has been mainly in the area of image processing. We chose network traffic control as an interesting and useful domain for testing this methodology.

SIGOP II [Lieberman and Woo 1976] is a centralized network traffic control scheme which appears suitable for decomposition into the parallel relaxation structure. SIGOP II uses a serial relaxation process to arrive at an appropriate timing plan for signals in the network. This serial relaxation process is used to transform an N-dimensional optimization problem (where N is

---

[4] There have been other attempts to develop fully or partially (e.g., hierarchically) distributed traffic light systems [Lonsley 1968, Rosdolesky 1972, Cutler 1976, Kinney et al 1977].

the number of intersections in the network) to an iterative sequence of N 1-dimensional optimization problems. Each iterative cycle in the serial relaxation process involves sequentially updating the timing plans at each intersection in the network by performing a local optimization based on incoming and outgoing traffic flow patterns (determined from examining the current timing plans of neighboring intersections).

Our approach to distributing SIGOP II is to place a processor at each intersection of the network with local sensors for traffic flow and with communication only among neighboring intersections. This type of single-level network is excellent in terms of cost (low communication cost and use of relatively inexpensive processing elements), reliability (processor failure would at most result in degraded performance in a small area in the immediate vicinity of the failure), and flexibility (incremental additions are easily incorporated and require only local modifications to the network).

A cycle of the relaxation process in the distributed version of SIGOP II involves updating all individual intersection timing plans in parallel (in contrast to the sequential process used in the centralized SIGOP II scheme). Since our distributed version of SIGOP II performs the same optimization calculations (using the same data) as the centralized case, the focus of this pilot study is the replacement of a serial relaxation control structure with a parallel (and distributed) relaxation control structure.

A test program was written which simulates a parallel relaxation version of SIGOP II. Although the test program does not employ all the knowledge used by SIGOP II to perform local node optimizations, it has produced encouraging results for a number of test cases which do not require the omitted knowledge in their solutions. However, in a number of cases the parallel scheme does not perform as well as the serial version, resulting in slow convergence, convergence to near optimal solutions, or oscillation. A closer investigation of these cases highlighted some basic problems in our use of the parallel relaxation control structure in distributing SIGOP II.

The local node optimization procedures used by the serial SIGOP II program are based partly on the fact that current neighboring timing plans are known and remain fixed during the updating of each intersection's timing plan. The introduction of parallelism generates uncertainty as to the values of neighboring timing plans which leads to oscillation. Figure 3 shows such a situation. Two neighboring intersections initially have timing plan X. These timing plans are incompatible, but if either intersection (but not both) changes to timing plan Y, an optimal solution is produced. In the parallel relaxation scheme, both intersections change to Y on the first iteration. Then again noticing that Y and Y are incompatible, both intersections change back to X, and so on. This problem is similar to the problem of inconsistency in distributed databases caused by multiple updating without synchronization. The standard relaxation

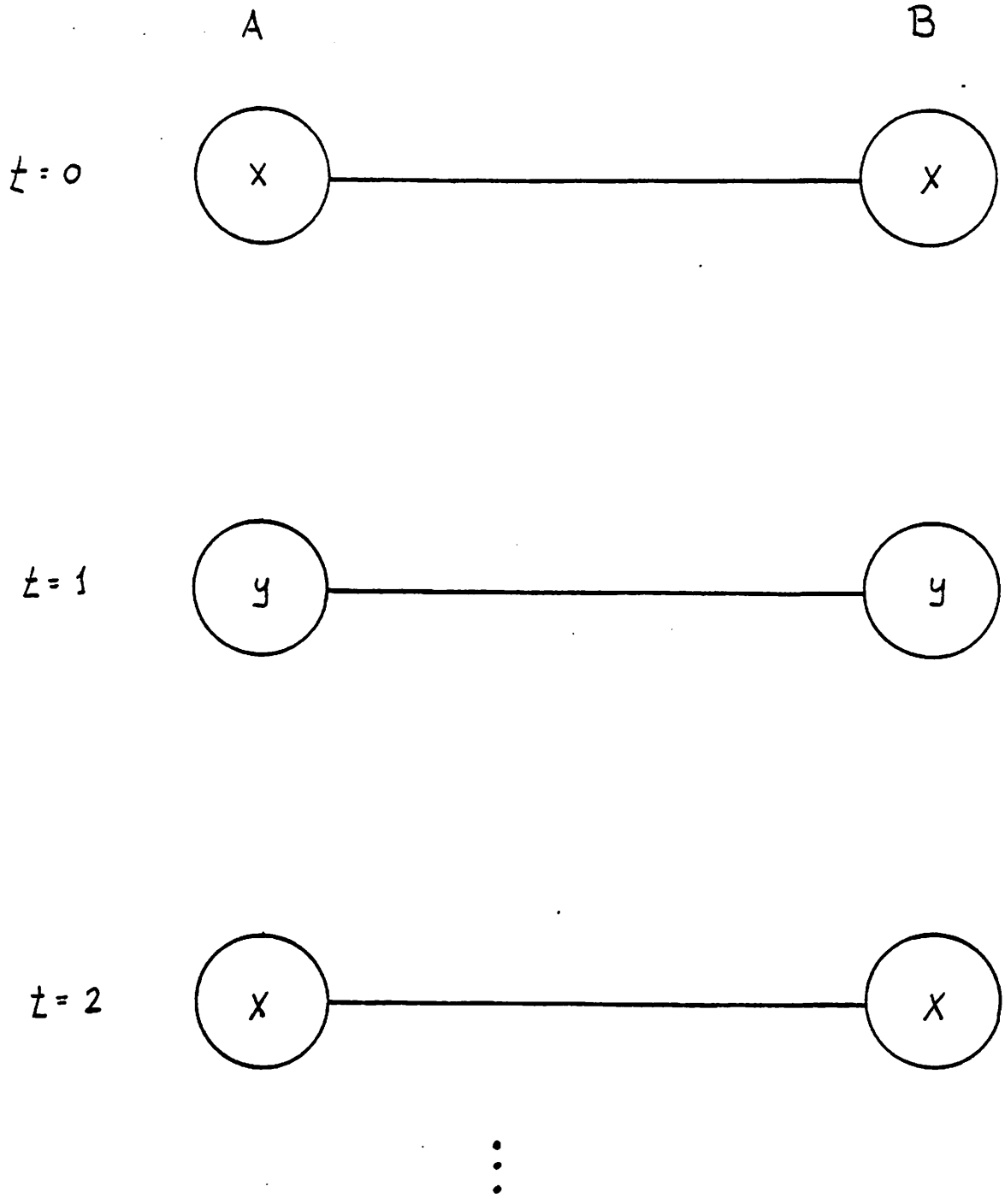


figure 3

approach of slowing convergence to attenuate oscillation cannot be applied since the compatibility relationship between timing plans is not well behaved (i.e., small changes to a timing plan may result in major changes in the compatibility relationship between neighboring timing plans). In the simple example above, an intermediate timing plan Z may not exist with a suitable compatibility.

A more crucial problem with the use of a parallel relaxation scheme is the loss of implicit ordering information utilized in the serial SIGOP II procedure. The serial version uses a maximal spanning tree ordering based on intersection volume as an ordering for node timing plan updating. In the traffic control domain, there is a high degree of non-local interaction between timing plans because a major component of the appropriateness of an intersection's timing plan is based on when the traffic flow will arrive at that intersection. Therefore, a change in a timing plan may affect the arrival times of traffic at distant intersections in the network (i.e., traffic flow implicitly defines a compatibility relationship between non-neighboring intersections). The use of the serial updating order based on the maximal spanning tree provides a heuristic which minimizes (during a single cycle) the effects on already updated timing plans of changes to traffic flow caused by updates to remaining timing plans.

These problems with the parallel relaxation version demonstrate that the control structure used in the centralized SIGOP II scheme (global node ordering based on the maximal spanning tree) is crucial to its success. This use of a global node ordering is incompatible with parallel relaxation involving refinement at all nodes simultaneously. Unfortunately, this node ordering is used by the SIGOP II algorithm in order to generate an optimal solution as well as for quick convergence. Current research is being directed to modification of the parallel relaxation scheme to introduce appropriate cooperation among neighboring intersections in the generation of local timing plans without requiring extensive synchronization. One direction being explored is to employ the maximal spanning tree heuristics in a distributed way. Another direction being pursued is the recasting of the knowledge used in SIGOP II into a probabilistic relaxation structure with alternative timing plans at each intersection evaluated simultaneously [Rosenfeld, Hummel, and Zucker 1976]. A forthcoming paper [Brooks and Lesser 1978] details this current research. In addition, further research is planned on how to handle special events (e.g., accidents, priority vehicles, concerts) within the relaxation paradigm, and to study the robustness of the relaxation process in terms of hardware failure.

The major point of this study was to examine the feasibility of using the relaxation paradigm as a methodology for doing cooperative distributed computation. It has illustrated some

basic problems with parallel relaxation in domains which have ill-behaved compatibility relationships between nodes and non-localized interactions between nodes.

5. A VEHICLE FOR THE DEVELOPMENT OF COOPERATIVE DISTRIBUTED METHODOLOGIES

In addition to the two studies discussed in Section 4, we have defined an artificial problem which can be parameterized to reflect a wide range of cooperative distributed applications. This problem does not require a large amount of expert knowledge in order to obtain some initial solutions, but does require a sophisticated problem solving and communication structure to perform the interpretation, planning, and monitoring functions necessary for an effective solution.

The problem is called the "Distributed Processing Game", a two-team game played in an infinite two-dimensional space. Two stationary points are specified in this space, one for each team. These points, or "homes", represent crucial areas for each team to defend. Each team is a separate distributed system charged with a single goal: capture of the opponent's home while defending its own home. The only active pieces in the game are nodes, mobile processors equipped with sensors, communication devices, and the capability of capturing the opponent's nodes and home. The game is not interactive; each team is provided with

cooperative procedures which must succeed or fail without external assistance.

At the start of the game, each team has an equal number of nodes, located at its own home area. The game proceeds in a series of discrete game cycles in which all nodes can perform any or all of the following actions:

(1) receive messages sent by other nodes of the same team;

(2) transmit messages to be received by other nodes of the same team;

(3) make a change in the node's current velocity (limited by a fixed maximum velocity); and

(4) attempt to capture an opponent's node(s) and/or home.

From the viewpoint of an individual node, we will look at each aspect of node activity in detail.

Sensing

Every node receives a scan of its immediate environment at the start of each game cycle. The form of this scan is a list of node identification numbers (opponent nodes are indicated only as opponents--no individual identification is provided), position, and current velocity of each node within the range of the scan. Scanning is subject to three types of error: failure to sense a node, erroneous sensing of a non-existent node, and incorrect sensing of the identification, position, and/or velocity of an

existing node. In addition to the scan, each node also correctly knows its own postion and velocity as well as the correct time (for time-stamping messages).

Due to the limited range of sensing and the possibility of sensor error, cooperation between a team's nodes is required in order to obtain a more accurate, global view of the world. Such cooperation, in conjunction with the integration of past views, provides a more accurate world interpretation on which to base decisions than can be obtained from the current sensor scan alone.

Communication

Each team has a separate channel for communicating between its nodes. There is no inter-team communication and no possibility of intercepting messages of the opponent.

A node can transmit messages to any or all teammates within transmission range. Three modes of transmission are provided: global broadcast, limited broadcast (specified by a set of node identification numbers), and point-to-point. Message transmission is subject to error. However, garbled messages are not modeled in the game, a message is either correctly received or not received at all.

The characteristics of a radio-like medium are modeled by the following restrictions on message communication:

finite message velocity -- Longer communication distances introduce longer communication delays. The delays are proportional to distance. Repeater nodes must be used (and correctly positioned) for long range communication.

limited number of transmitted messages per game cycle -- Each node can send only a fixed maximum number of messages in any given cycle. Global broadcast, limited broadcast, and point-to-point messages all count equally toward this limit.

limited number of received messages per game cycle -- Each node can receive only a fixed maximum number of messages in any given cycle. If more messages arrive at a node than can be received in a cycle, the messages travelling the shortest distance override the others. These overridden messages are lost to that node.

These communication characteristics make both the maintenance of a suitable communication configuration and the proper allocation of the medium important aspects of successful play.

Capture

During every game cycle, each node can perform one capture attempt. Capture affects all opponent nodes within a sector emanating from the capturing node, called the "capture zone" (see Figure 4). If the opponent's home is also in the capture zone, it too is affected. The potency of the capture attempt decreases with distance from the capturing node and with the angle between the two radii defining the capture zone. Although the same capture attempt can affect both nodes and the home, these effects are different and are discussed separately.

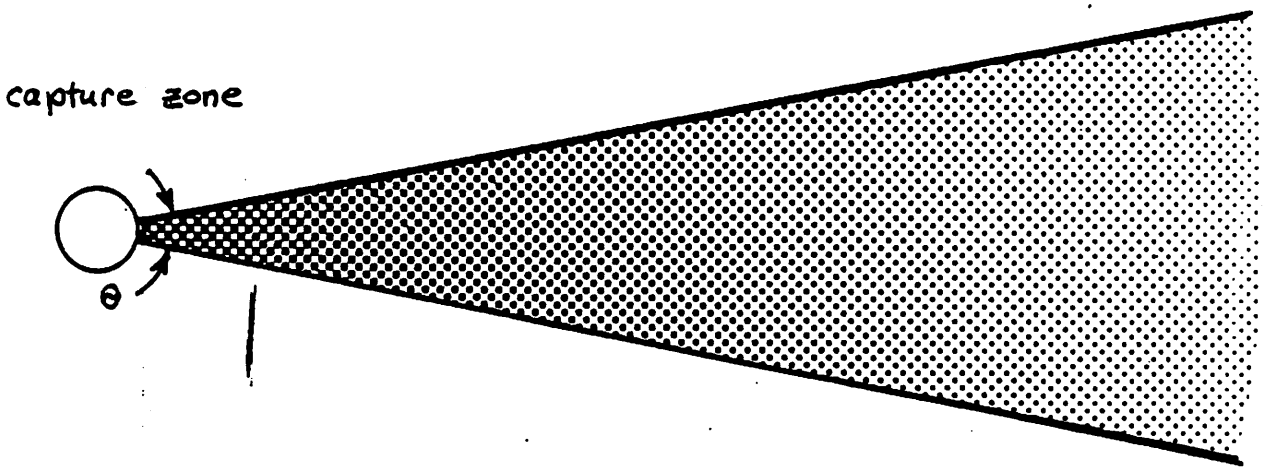


figure 4

cooperation during capture

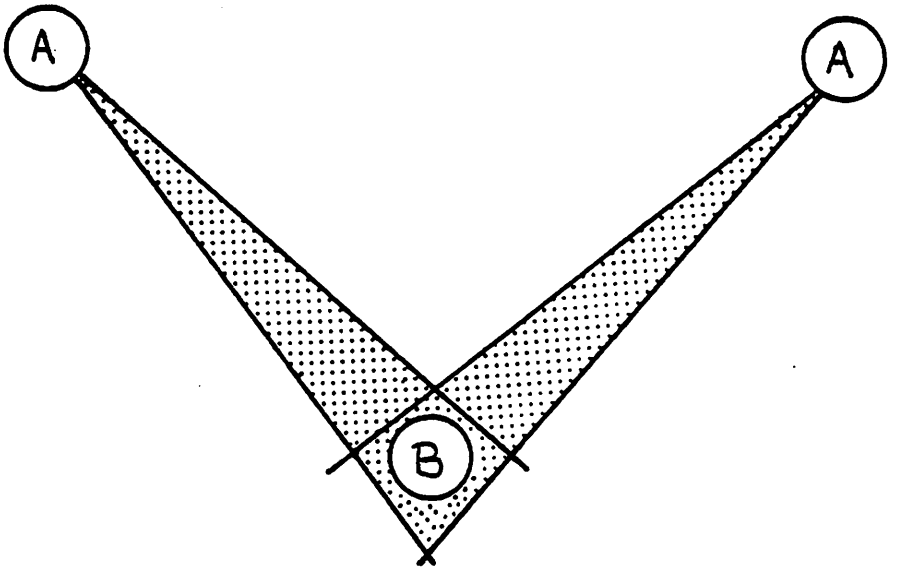


figure 5

Processors must be captured in a single game cycle, i.e., the total effect of all capture attempts in the cycle must meet or exceed the amount required for node capture. If at the end of the cycle a node is not captured, it begins the next cycle undamaged. Captured nodes, however, disappear forever from the game without announcement.

Homes are more resistant to capture than are nodes, but the effects of capture attempts on a home accumulate over the entire game. If at the end of a cycle the cumulative effects on a home exceed the amount required for home capture, the home is lost and its team declared the losers.

Both homes and nodes are considered as points (in the mathematical sense); there is no shielding of capture effects by a home or node. (Similarly, there is no potential for processor-processor or processor-home collision.)

Due to the cumulative nature of capture attempts (single cycle for nodes, the entire game for homes), cooperation between a team's nodes can greatly enhance the possibility of capturing an opponent. A simple example of this is illustrated in Figure 5. Two nodes of team A are attempting to capture a single node of team B. The geometry of the situation is such that the combined capture effects of the nodes of team A exceeds the

threshold of node B, while the capture effect of node B is insufficient to capture the nodes of team A.

Uncertainty and the Game

The environment of the game creates a large amount of uncertainty with which both teams must deal:

sensing- potentially errorful and must be resolved by the integration of multiple views.

communication -- messages are not always received.

nodes -- are subject to capture, disappearing without warning.

The competitive nature of the game further complicates the environment. A team is not the sole active agent in the game; the opposing team also changes the state of the world, potentially attempting to interfere as much as possible with the other team.

The game embodies many of the design issues for cooperative distributed systems arising from such applications as the demand bus scheduling example discussed in section 2.2. Given the complexity of implementing any such real world cooperative distributed system, the game provides a convienent test bed for quickly exploring new cooperative methodologies and highlights research problems which must be solved. The ability to change the parametrization of many of the properties of the game also provides an environment which can model a wide and diverse set of

constraints that exist in these real world applications. A simulator for the same has been written and various simple cooperative strategies are currently being explored.

6. CONCLUSION

As a result of this preliminary work, we feel that there is a strong possibility that methodologies can be developed for cooperative distributed systems in which the distributed algorithms and control structures function with both inconsistent and incomplete data. This methodology for decomposition is necessary in order to extend the range of applications that can be effectively implemented in a distributed environment. We also feel that work in knowledge-based AI systems and on understanding natural systems has strong relevence to the development of these methodologies.

ACKNOWLEDGMENTS

We would like to acknowledge the research efforts of Richard Brooks on network traffic control which are discussed in this paper. We would also like to acknowledge the very useful criticisms received from Michael Arbib, Lee Erman, John Lowrance, and Nils Nilsson.

REFERENCES

Anderson, J.P. (1965). "Program Structures for Parallel Processing," CACM, Vol. 8, No. 12, 786-789.

Arbib, M.A. (1975). "Artificial Intelligence and Brain Theory: Unities and Diversities," Annals of Biomedical Engineering 3, 238-274.

Arbib, M.A. (1978). "Segmentation, Schemas, and Cooperative Computation," S. Levin (ed.), Studies in Biomathematics, MAA Studies in Mathematics, Mathematical Association of America.

Barrow, H.G. and Tenenbaum, J.M. (1976). "MSYS: A System for Reasoning about Scenes," AI Center Technical Note 121, SRI.

Baudet, G.M., Brent, R.P. and Kung, H.T. (1977). "Parallel Execution of a Sequence of Tasks on an Asynchronous Multiprocessor," Department of Computer Science, Carnegie-Mellon University.

Brooks, R.S. and Lesser V.R. (1978). "Distributed Problem Solving Using Iterative Refinement," COINS, University of Massachusetts (in preparation).

Cerf, V.G. and Kahn, R.E. (1974). "A Protocol for Packet Network Intercommunication," IEEE Trans. Commun., Vol. COM-22, 637-648.

Constantine, L. (1968). "Control of Sequence and Parallelism in Modular Programs," SJCC, Vol. 26, 409-414.

Corkill, D.D. (1978). "Hierarchical Planning in a Distributed Environment," COINS, University of Massachusetts (in preparation).

Crowther, W.R., Heart, F.E., McKenzie, A.A., McQuillan, J.M., and Walden, D.C. (1975). "Issues in Packet Switching Network Design," Proc. AFIPS Nat. Comput. Conf., Vol. 44, 161-175.

Cutler, S.E. (1976). "Microcomputer Networks in Control Applications," Ph.D. Thesis, M.I.T.

Deppe, M.E. and Fry, J.P. (1976). "Distributed Data Bases - A Summary of Research," Computer Networks, Vol. 1, No. 1, 130-138.

Dimmler, D.G., et al. (1976). "Brookhaven Reactor Experiment Control Facility - A Distributed Function Computer Network," IEEE Nuclear S., 23(1), 398-405.

Dijkstra, E.W. (1974). "Self-stabilizing Systems in Spite of Distributed Control," CACM, Vol. 17, No. 11, 643-644.

DOT/FHA (1976). "Traffic Control Systems Handbook," U.S. Department of Transportation/Federal Highway Administration.

Engelmore, R.S. and Nii, H.P. (1977). "A Knowledge-Based System for the Interpretation of Protein X-ray Crystallographic Data," Technical Report Stan-CS-77-589, Stanford University, Stanford, California.

Erman, L.D. and Lesser, V.R. (1975). "A Multi-Level Organization for Problem Solving Using Many Diverse, Cooperating Sources of Knowledge," Proc. IJCAI-75, 483-490.

Eswaran, Gray, Lorie, and Traiger. (1976). "The Notions of Consistency and Predicate Locks in a Database System," CACM, Vol. 19, No. 11, 624-633.

Farber, D.J., et al. (1973). "The Distributed Computing System," in Advances in Computer Comm., ARTECH House.

Feigenbaum, E.A., Buchanan, B.G. and Lederberg, J. (1971). "On Generality and Problem Solving: A Case Study Using the DENDRAL Program," Machine Intelligence 5, American Elsevier, 165-190.

Feldman, J.A. (1977). "A Programming Methodology for Distributed Computing (among other things)," Department of Computer Science, University of Rochester, Rochester, NY.

Fennell, R.D. and Lesser, V.R. (1977)."Parallelism in AI Problem Solving: A Case Study of HEARSAY-II," IEEE Transactions on Computer - Special Issue on Multiprocessors, C-26, 2, 98-111.

Fikes, R.E. and Pease, M.C. (1975). "An Interactive Management Support System for Planning, Control, and Analysis," Stanford Research Institute Technical Report 12.

Fosdick, H.C., Schantz, R.E. and Thomas, R.H. (1977). "Operating Systems for Computer Networks (1)," 2nd Annual Brown University Workshop on Distributed Computation.

Fox, M.S. (1977). "On Organizational Theory and its Relation to A.I.," private communication.

Galbraith, J. (1973). "Designing Complex Organizations," Addison-Wesley.

Gerla, M. (1973). "Deterministic and Adaptive Routing Policies in Packet-Switched Computer Networks," Proceedings of the Third IEEE Data Communications Symposium, 23-28.

Gosden, J.A. (1966). "Explicit Parallel Processing Description and Control in Programs for Multi- and Uni-Processor Computing," FJCC, 651-660.

Hewitt, C. (1974). "Protection and Synchronization in Actor Systems," AI Laboratory, MIT, Working Paper 83.

Hewitt, C. (1975). "Viewing Control Structures as Patterns of Passing Messages," AI Laboratory, MIT, Working Paper 92, Revised April 1976.

Hoffman, T.G. (1975). "Upgrading Gauge Measurement with Distributed Computers," Control Eng., 22(2), 39-41.

Infotech State of the Art Report (1976). "Distributed Systems," Infotech International Ltd., Berkshire, England.

Jefferson, D. (1977). "Hydra-Message Facility," private communication.

Kahn, R.E. (1975). "The Organization of Computer Resources into a Packet Radio Network," NCCC, 177-186.

Karp, P.M. (1973). "Origin, Development and Current Status of the ARPA Network," 7th Annual IEEE Comput. Soc. Int. Conf. Dig. Papers, 49-52.

Kilmer, W.L., McCulloch, W.S. and Blum, J. (1969). "A Model of the Vertebrate Central Command System," Int. J. Man-Machine Studies 1, 279-309.

Kimbleton, S.R. and Schneider, G.M. (1975). "Computer Communication Networks: Approaches, Objectives and Performance Considerations," Comp. Survey, Vol. 7, No. 3, 129-173.

Kinney, Kumar, Lin, Ulrich, Petz and Smith (1977). "Distributed Computer Systems for Traffic Control," Prepared by the Department of Electrical Engineering, University of Minnesota, for DOT, Office of Secretary, Office of University Research.

Lampson, B. and Sturgis, H. (1977). "Crash Recovery in a Distributed Data Storage System," 2nd Annual Brown University Workshop on Distributed Computation. To appear in CACM.

Lenat, D.B. (1975). "Beings: Knowledge as Interacting Experts," AI Laboratory, Stanford University, Proc. IJCAI-75, 126-133.

Lenat, D.B. (1976). "Beings: Synthesis of Large Programs from Specific Dialogues," Stanford AI Laboratory.

Lesser, V.R. and Erman, L.D. (1978). "An Experiment in Distributed Problem Solving," Computer Science Department, Carnegie-Mellon University (in preparation).

Lieberman, et al. (1974a). "Variable Cycle Signal Timing Program: Third Generation Policies for UTCS," KLD Assoc., Inc. for FHA, NTIS PB-241 717, Vol. 1.

Lieberman, et al. (1974b). "Variable Cycle Signal Timing Program: CYRANO: Cycle-Free Responsive Algorithms for Network Optimization: Moderate, Congested, and Light Flow Regimes," KLD Assoc., Inc. for FHA, NTIS PB-241 720, Vol 3.

Lieberman and Woo (1976). "SIGOP II: A New Computer Program for Calculating Optimal Signal Timing Patterns," Transpn. Res. Record, Report 596.

Longley, D. (1968). "A Control Strategy for a Computer-Controlled Traffic Network," Transpn. Res. 2, 391-408.

Manning, E. and Peebles, R.W. (1977). "A Homogeneous Network for Data Sharing - Communications," 2nd Annual Brown University Workshop on Distributed Computation.

Merlin, P.M. (1975). "A Methodology for the Design and Implementation of Communication Protocols," IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

Nii, H.P. and Feigenbaum, E.A. (1977). "Rule-Based Understanding of Signals," Conference on Pattern-Directed Inference Systems, Hawaii.

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

Peebles, R. and Manning, E. (1975). "A Computer Architecture for Large (Distributed) Databases," Proc. International Conference on Very Large Databases.

Peebles, R. (1977). "Concurrent Access Control in a Distributed Transaction Processing System," 2nd Annual Brown University Workshop on Distributed Computation.

Pyke, T.N. and Blanc, R.P. (1973). "Computer Networking Technology - A State of the Art Review," IEEE Computer, Vol. 6, 12-19.

Reddy, D.R. (1976). "Speech Recognition by Machine: A Review," Proc. of IEEE 64, 501-531.

Reed. S. (1978). "The Honey Bee Colony as a Metaphor for Distributed Computing," COINS, University of Massachusetts (in preparation).

Riseman, E. and Arbib, M.A. (1977). "Computational Techniques in the Visual Segmentation of Static Scenes," Comp. Graphics and Image Processing 6, 221-276.

Rosdolsky, H.G. (1972). "A Method for Adaptive Traffic Control," Transpn. Res., Vol. 7, 1-16.

Rosenfeld, A., Hummel, R.A. and Zucker, S.W. (1976). "Scene Labeling by Relaxation Operations," IEEE Transactions on Systems, Man, and Cybernetics, SMC-6.

Rubin, S. and Reddy, R. (1977). "The Locus Model of Search and its Use in Image Interpretation," Proc. IJCAI-77, 590-595.

Sacerdoti, E. (1974). "Planning in a Hierarchy of Abstraction Spaces," Artificial Intelligence, Vol. 5, No. 2, 115-135.

Sacerdoti, E. (1975). "The Non-Linear Nature of Plans," Proc. IJCAI-75, 206-214.

Sacerdoti, E. (1977a). "Language Access to Distributed Data with Error Recovery," Proc. IJCAI-77, 196-202.

Sacerdoti, E. (1977b). "A Structure for Plans and Behavior," Elsevier Press.

Schoeffl, J. D. and Rose, C.W. (1976). "Distributed Computer Intelligence for Data Acquisition and Control," IEEE Nucl. S. 23(1), 32-54.

Shortliffe, E.H. (1974). "MYCIN--A Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection," Stanford AI Memo 251.

Simon, H.A. (1962). "The Architecture of Complexity," Proceedings of the American Philosophical Society, Vol. 106, No. 6, 467-482.

Simon, H.A. (1969). "Sciences of the Artificial," MIT Press, Cambridge, Massachusetts.

Smith, R.G. and Davis, R. (1978). "Partitioning and Distribution of Knowledge in a Distributed Problem Solver," Proc. CSCSI/SCEIO Conf.

Svobodova, L. (1977). "Distributed Computer Systems: Research Proposal Submitted to ARPA," 2nd Annual Brown University Workshop on Distributed Computation.

Tajibnapis, W.D. (1976). "Message-Switching Protocols in Distributed Computer Networks," Merit Computer Network, Michigan State University, Wayne State University and University of Michigan.

Tajibnapis, W.D. (1977). "A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network," CACM, Vol. 20, No. 7, 477-485.

Tate, A. (1975). "Interacting Goals and Their Use," Proc. IJCAI-75, 215-218.

Tate, A. (1976a). "Project Planning Using A Hierarchic Non-Linear Planner," University of Edinburgh, Department of A.I. Research Report No. 25.

Tate, A. (1976b). "NONLIN: A Hierarchic Non-Linear Planner," University of Edinburgh, Department of A.I. Draft Memo.

Tate, A. (1977). "Generating Project Networks," Proc. IJCAI-77, 888-893.

Thomas, R.H. (1976). "A Solution to the Update Problem for Multiple Copy Data Bases Which Uses Distributed Control," BBN Report #3340.

Wagner, Barnes, and Gerlough (1971). "Improved Criteria for Traffic Signal Systems in Urban Networks," National Cooperative Highway Research Program Report 124, Highway Research Board, Washington D.C.