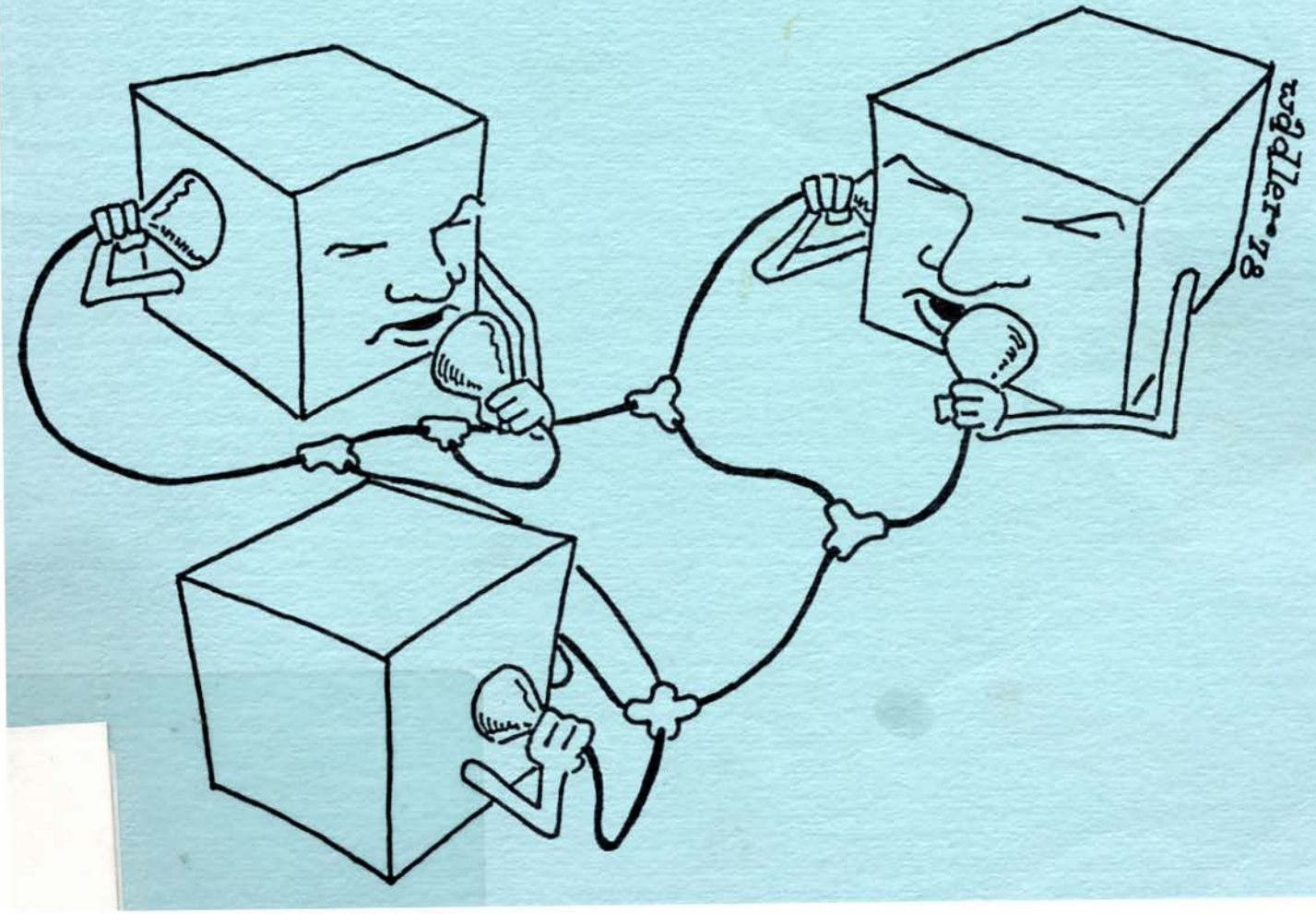
DISTRIBUTED SENSOR NETS

Sponsored By:
Information Processing Techniques Office
Defense Advanced Research Projects Agency

Hosted by:
Carnegie-Mellon University
Pittsburgh, Pennsylvannia
December, 1978



DISTRIBUTED SENSOR NETS

Proceedings of a Workshop
held at
Carnegie-Mellon University
December 7-8, 1978

Sponsored by the
Defense Advanced Research Projects Agency

A limited number of copies
are available from the
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

TABLE OF CONTENTS

SESSION VII - DISTRIBUTED AI

AUTHOR INDEX

Position Location in a TDMA Network

Stephen Cable

Rockwell International

Introduction

Many of the communications systems being developed today for the tactical environment use some form of TDMA in order to share the communications resource among the network elements. In addition, a time dependent waveform (eg, JTIDS, Packet Radio, etc.) is often utilized to increase jam resistance. These systems require time synchronization within the network in order to be able to successfully communicate. The timing measurements necessary to provide network synchronization can be extended to provide accurate range measurements between network elements. These range measurements can be used to calculate the positions of all network elements relative to some grid which has been established by the network. The tactical value of a relative position location function is well established for providing relative nagivation, targeting, resource mapping and allocation, routing and low probability of intercept information. The availability of range measurements, which could be used for position location, and the value of position location as a tactical tool motivated the development of the position location methods described in this paper.

The method for providing a relative position location function is summarized below. As a part of the network initialization process, a relative grid is established for the network. Each element in the network will be responsible for tracking its own position relative to that grid. Accurate range measurements are obtained by accurately controlling the time of transmission (TOT) and by accurately measuring the time of arrival (TOA) of messages. The propagation delay measurement obtained from the TOT and the TOA is multiplied by the speed of propagation to obtain a range measurement. If the transmitted message contains the position of the transmitter, the range measurement can be used along with similar range measurements from other sources to calculate an estimate of the receiver's position and the current network reference time. Periodically, these estimates are used by a Kalman filter linear estimator to develop and correct a state model of the dynamics of the unit and its internal time base. The rate at which these models must be updated is a function of the mobility of the platform and its position location accuracy requirements. By using these state models, each unit can track its estimated position and an estimate of the network reference time. This information is then made available to other units for use in their position location algorithms. A hierarchical structure is used to determine which potential sources of position information should be utilized by a particular user. This structure prevents the feedback of errors which can result in an unstable condition. The remainder of the paper will provide an overview of the major concepts introduced above, will discuss some of the factors which determine accuracy, and will consider the issues associated with actually implementing a position location function.

Multilateration

Multilateration is the use of range measurements from multiple sources to determine position relative to the locations of the sources. Multilateration is suited for use in a tactical TDMA network for a variety of reasons, including:

a. It utilizes available propagation delay measurements

b. Position estimates can be calculated by each element

c. It does not require fixed references

d. It does not rely upon critical nodes.

Given the TOT and TOA of the message, an estimate of the line of sight range between the transmitter and receiver can be computed. If the received message contains the absolute or relative position of the source (transmitter), the receiver can use this information, in conjunction with the measured range and similar measurements from other sources, to estimate its position. Each range measurement defines a circle of radius R (where R = measured range) with its center at the source. According to the measurement, the receiver lies somewhere on the circle. In the horizontal case (ie, altitude assumed known) perfect range measurements from three noncolinear sources will generate three circles with a unique common intersection which is the position of the receiver. However, error-corrupted measurements will produce a set of circles with no common intersection or an incorrect common intersection. This result is illustrated in figure 1 for four sources and one receiver. Given these error-corrupted range measurements, the receiver is required to calculate a position estimate which is in some sense optimum.

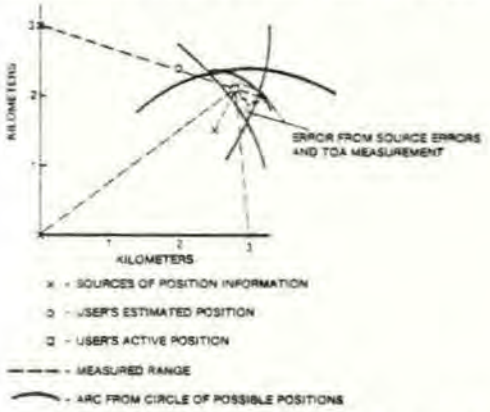


Figure 1. Multilateration Measurements for Position Location.

By linearizing the problem using a Taylor's series expansion and employing a weighted least square error criterion, this problem can be efficiently solved by a microprocessor. This method requires an initial position estimate provided by previous measurements or by an initialization algorithm which is beyond the scope of this paper. The result of the linearization can be observed in figure 2a. The ith

range measurement now constrains the solution to lie on the straight line which is tangent to the circle of radius R(i) at the point where the line connecting the ith source and the estimated position intersects the circle. The weighted least square error criterion is then used to solve for an estimate. The weights take into account the relative accuracy of the position and time estimates of the sources. The variance of the error in the estimate is also calculated based on the estimated variance of each measurement. The final estimate derived from the measurements of figure 1 is illustrated in figure 2b.

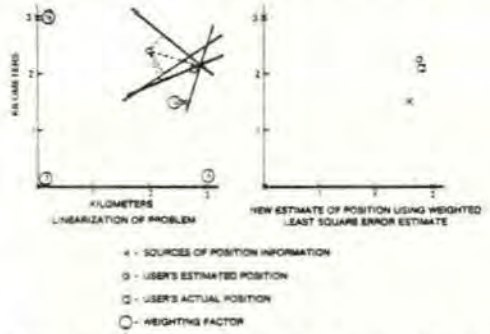


Figure 2. Linearized Solution of Multilateration Problem.

Thus, a network element provided with three range measurements of adequate quality can compute an estimate of its relative position.

Range Measurements

The accuracy and quantity of range measurements available to a unit is an important factor in the accuracy of the position location function. On the other hand, the portion of the network capacity required to support position location messages should be minimized. Thus, a range measurement technique which fulfills both of these requirements is desirable. There are two basic methods (modes) of obtaining a range measurement for position location and a clock offset measurement for time synchronization. One method, a passive mode measurement, requires only the reception of a position report from the source. The second method, an active mode measurement, requires an interrogation of the source of position information and a reply containing the information needed to calculate an active range measurement. A comparison of these two modes reveals that the selection of the measurement mode is dependent upon the operational requirements and the network environment of the unit.

In the passive mode of operation, the unit estimates its position and clock offset utilizing passively received position reports. The form of the passive mode measurement is illustrated in figure 3. By including terms to indicate the sources of error, the equation for the measurement, D, in figure 3 can be rewritten as

$$D = (\Delta t_2 + \Delta t_3) * V - ER + n + m \qquad (1)$$

where

n = errors introduced by incorrect source position estimates

m = TOA measurement error.

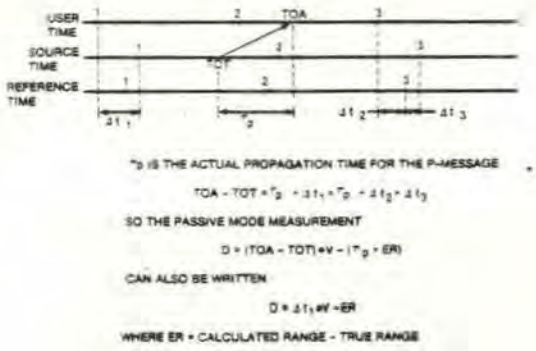


Figure 3. Passive Mode Observation.

Upon reception of similar measurements from at least three noncolinear sources, equation (1) can be linearized and an estimate of the clock offset and the error in the unit's previous position estimate can be obtained.

In the active mode of operation, a unit must interrogate the sources from which it desires to obtain position and time information. Upon reception of this interrogation, the interrogated unit replies with a position report containing the TOA of the interrogation and the position of the source at the time of the reply. The form of the active mode range measurement is illustrated in figure 4. The two transmissions of the active mode provide an independent time measurement, Dt, and an independent position error measurement, Dp. From figure 4, Dt and Dp can be written as

$$Dt = \Delta t_1 + (m2 - m1)/2 \qquad (2)$$

$$Dp = ER + (m2 + m1)/2 + n \qquad (3)$$

where

mi = ith TOA measurement error

n = errors introduced by incorrect source position estimate.

These equations show that clock offset and position errors have been decoupled in the active mode.

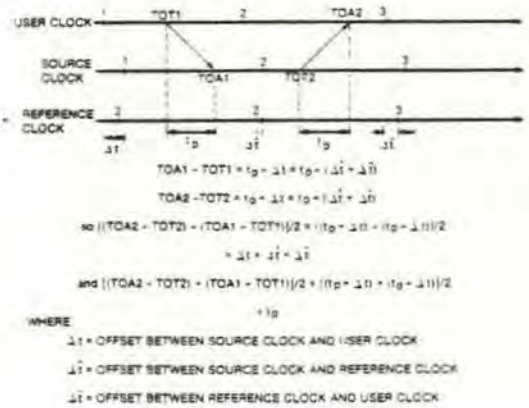


Figure 4. Active (RTT) Observation.

Analysis and simulation results indicate that the active mode range measurements produce better performance that is less dependent upon the geometry of the sources than for the passive mode case. However, there are reasons for choosing to use the passive mode at the expense of degraded performance. In an operational scenario, some units may be required to operate in a radio silent mode. At the same time, it may be important for these units to include position information in their infrequent transmissions. Such units would certainly operate in the passive mode. Another important consideration is the amount of network traffic required for position location. If network capacity is a problem and an adequate source geometry is available, then assigning some units to the passive mode can reduce the traffic requirements. Another solution to this problem is the use of a hybrid mode employing active time measurements with passive position measurements. A discussion of the hybrid mode is beyond the scope of this paper. In an attempt to satisfy accuracy requirements, capacity limitations and operational objectives, a typical network may include active mode units, passive mode units, and hybrid mode units.

Kalman Filter

In order to successfully track the clock and the position of a unit using noisy measurements and to extrapolate time and position between sets of measurements, a state model of the clock and the dynamics of the unit must be maintained. Thus, once position and time estimates have been obtained from the solution to the linearized multilateration equations, this new estimate must be combined with the previous estimates to update the state model. A Kalman filter was selected to track the state of the system. A Kalman filter provides improved performance over simpler fixed gain filters (eg α , - β tracker) and it provides the error covariance matrix necessary to implement the covariance defined hierarchy described below. The Kalman filter is defined by the state model description of the system and the error-corrupted measurements which are available for updating the model. For a ground, mobile network, a six-dimensional state vector has been demonstrated to provide an adequate state model. The six components are clock offset, clock drift rate, x-direction position, y-direction position, x-direction velocity, and y-direction velocity. This model assumes that altitude is provided from an external measurement, independently from the position location function. A variant of the above state vector would use heading and velocity in place of velocity in the x and y directions. The measurement used by the Kalman filter would be the solution to the linearized multilateration equations and its associated covariance matrix. Platforms exhibiting higher mobility, eg aircraft, would require an extended state vector and additional measurements from onboard navigation equipment in order to successfully track the platform.

Given a measurement and its variance, a Kalman filter attempts to update the state model in a way such that the trace of the covariance matrix of the state vector is minimized. The basic Kalman filter equations are summarized below:

Extrapolation

$$\underline{s}\,(\text{ext}) = A\,\underline{s}\,(\text{old})$$

$$C\,(\text{ext}) = A\,C\,(\text{old})A^T$$

Update

$$\underline{s}\,(\text{new}) = A\,\underline{s}\,(\text{old}) + K\,(\text{new})\,[\,\underline{z}\,(\text{new}) - HA\,\underline{s}\,(\text{old})]$$

$$K\,(\text{new}) = C\,(\text{ext})\,H^T\,[H\,C\,(\text{ext})\,H^T + R\,(\text{new})]^{-1}$$

$$C\,(\text{new}) = C\,(\text{ext}) - K\,(\text{new})\,H\,C\,(\text{ext})$$

where

$\underline{s}$ = state vector

A = state transition matrix

C = state vector error covariance matrix

K = Kalman filter gain matrix

$\underline{z}$ = measured values

H = linear transformation from $\underline{s}$ to $\underline{z}$

R = measurement error covariance matrix.

These Kalman filter equations and the multilateration measurements provide the basis for tracking the relative position of network elements.

Network Architecture

The ability of each network element to perform position location is dependent upon the existence of a common relative grid and the reception of an adequate number of useable position reports. The network structure of the position location function should perform four major tasks. Those tasks are:

a. Support time synchronization in the network

b. Establish a rectilinear coordinate grid for relative navigation

c. Ensure an adequate supply of position reports to support any unit attempting to perform position location

d. Provide for stability and robustness of the position location function.

Due to the interrelationship between time synchronization and position location, any network structure which performs the last three tasks will support time synchronization in the network. For applications where relative navigation is not required, the position location algorithms can provide time tracking only. The clock offset measurements can be provided by active mode measurements without position or by passive mode measurements with a rough position estimate.

In conjunction with the network initialization procedure, the position location network structure should define a relative rectilinear grid and, if possible, a geodetic grid. Some of the network elements are given the responsibility for establishing the grid. These elements are:

a. Master Unit (MU). An element whose relative position is defined to be the origin of the grid.

b. Relative Position Reference (RPR). An element which knows its position relative to the MU and one other RPR. These are at least two RPR's in the network. The ranges between the MU and the RPR's may be obtained through round-trip timing measurements, but some prior information must be available to obtain an unambiguous estimate of their relative positions.

c. Absolute Position Reference (APR). An element with an accurate estimate of its geodetic coordinates obtained from some source other than the position location function.

During network initialization, the MU and RPR's define a relative rectilinear grid. Given two APR's, the relative grid can be mapped onto a geodetic grid through a coordinate transformation algorithm. Once the grid is established, the loss of an MU, RPR, or APR does not disable the position location capability of the network. An alternate master unit (AMU) is designated to assume the role of the MU upon loss of the original master unit. This feature enhances the survivability of the position location function in the network.

The covariance defined hierarchy is used to select the useable sources of position information from the set of all elements within line of sight (LOS) of a particular element. Each element includes a quantized estimate of the error variances of its position and time estimates in its position report. These variances are provided by the Kalman filter of each element. The position report also contains information indicating the number of relays between that element and the MU. An element selects the sources with the most accurate position information which are closer to the MU than it. In this manner, positive feedback of errors is eliminated.

Performance Considerations

The algorithm and network architecture introduced above were the subject of an in-depth analysis to determine the accuracy of the system and its impact upon network operation. This effort utilized an analysis of the state model, a Kalman filter error sensitivity analysis program and a Monte Carlo simulation to characterize the effects of various factors upon performance. It was found that the performance of the position location function is a complex function of the update rate, the mobility of the platform, the TOA measurement accuracy, the propagation effects, the quality and topology of the sources used, and several other factors. By characterizing the individual and cumulative effects of these factors, required values for parameters, such as required update rate, required TOA estimator accuracy, etc., which will provide adequate accuracy for a given operational scenario can be determined. The impact of three of the performance factors is summarized below.

Update Rate/Platform Dynamics

A major impact of the position location function upon a network is the portion of the communications capacity of the network which must be allocated to position location. Ideally, the minimum number of position reports necessary to meet the performance goal should be used. This implies that the update rate of each unit should be as low as possible. The major factors determining the update rate are the operational accuracy requirements and the mobility of the platform. The two basic sources of error, which are a function of the update rate, are errors introduced by state extrapolation and errors caused by combining position reports received at different times into one measurement. These errors increase as the mobility of the platform increases and decrease as the update rate increases. A highly mobile platform exhibits significant acceleration and higher order derivatives of motion. These unmodeled states can cause divergence of the estimate unless they are compensated for by age weighting or similar techniques. An analysis of these errors was performed using the Kalman filter error sensitivity analysis. From these results and given the accuracy requirement, the expected dynamics of the platform, and the conditions (source error, GDOP, etc.) under which the accuracy is required, a minimum update rate can be selected. This interaction is illustrated in figure 5, where maximum update periods were selected for various platforms and performance goals.

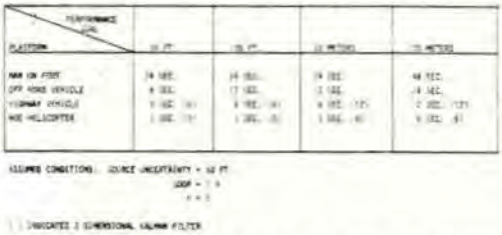


Figure 5. An Example of Update Rate vs. Platform Goals.

Propagation Delay Measurement

The sources of error in the propagation delay measurement include TOT errors, propagation effects, and TOA errors. The sum of all these errors can be lumped together and called the TOA measurement error. The impact of TOA measurement errors on position accuracy is a function of the mode of the unit (active or passive), the number of sources used, the dynamics of the unit, and the state model used by the Kalman filter. The effect of TOA uncertainty was characterized under various conditions using the analysis tools mentioned earlier. The uncertainty in the TOA measurement increases the uncertainty of the measurement supplied to the Kalman filter. This increases the response time of the filter, making it more difficult to track mobile platforms. The analysis indicates that a TOA of one sigma uncertainty of 100 ns may be acceptable for a slowly moving platform (eg, manpack), but the accuracy must increase as the mobility increases in order to maintain the same accuracy and update rate.

Network Topology

The topology of the network has a significant impact on the position location accuracy of the units in the network. A number of different network properties determine the configuration and quality of sources available to an individual unit. Some of these properties are listed in table 1.

Table 1. Network Properties and Source Quality.

PROPERTY	COMMENTS
Covariance defined hierarchy	Determines quality and choice of sources
Number of units in net	Effects number of sources available
Area of coverage	Effects network connectivity
Location of MU, RPR and APR	Effects source quality
Network dynamics	Effects availability and quality of sources
Network terrain	Effects network connectivity

The geometric dilution of precision (GDOP) is a measure of the errors introduced by the geometry of the sources. It is usually defined as

$$\sqrt{\sigma x^2 + \sigma y^2 / \sigma m^2} \qquad (4)$$

where

σx^2 = variance of the solution for position in the x-direction

σy^2 = variance of the solution for position in the y-direction

σm^2 = variance of the measurement error from sources.

As GDOP increases, the accuracy of a measurement decreases. GDOP is a function of the method of range measurement. The correlation between time and position errors in the passive mode accounts for much of the GDOP present in that mode. As a comparison between passive and active mode GDOP, consider the contours of equal GDOP plotted in figure 6 and figure 7. Obviously, for the source configuration of the figures, the active mode should provide better performance. This premise is borne out by the simulation results recorded in figure 8 for a unit traveling with constant velocity along the path indicated in figure 6 and figure 7.

One response of a passive mode unit with poor performance due to GDOP would be to switch to active mode operation. However, operational objectives may force some units to operate in the passive mode. In that case, it is the responsibility of the network to provide adequate sources of position information.

Implementation Issues

The impact of the position location function can be identified in three areas. These areas are: (1) network throughput, (2) hardware, and (3) software.

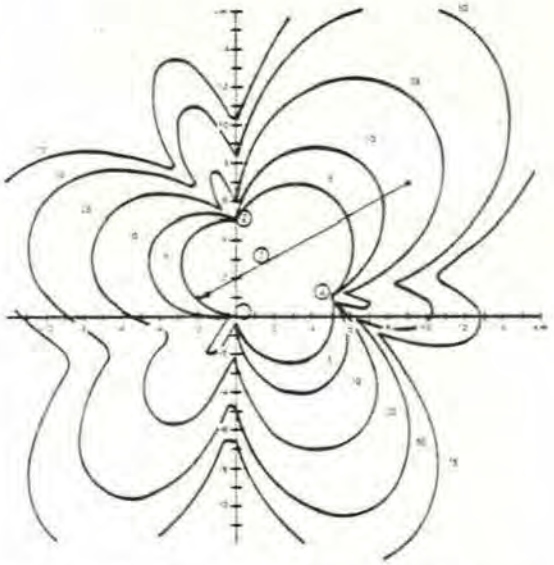


Figure 6. GDOP Contours for Basic Position Location Element (Passive).

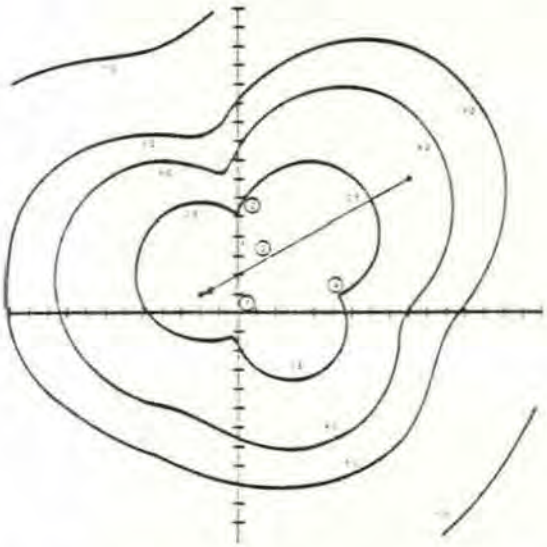


Figure 7. GDOP Contours for Basic Position Location Element (Active).

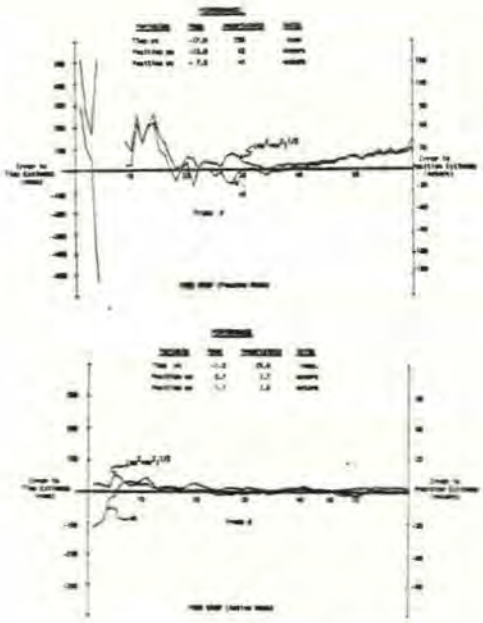


Figure 8. Simulation Results.

Throughput

The throughput or data distribution capability of the network is impacted in two ways. First, some amount of overhead traffic is introduced into the network to support the requirement for position reports. This traffic is in the form of overhead bits added to existing messages and additional messages created solely for the purpose of carrying position information. Second, some of the processing time of the unit must be used to process position information. For some networks, processing delays can become the limiting factor on throughput. Consequently, the amount of processing required for POSLOC could have an adverse effect on network throughput.

In figure 9, the average bits/sec of position information received by an average unit is plotted as a function of the number of units within LOS of that unit. This is a realistic measure of network loading because the position reports are not relayed when used solely for computing position. Several plots are included in figure 9, representing several network make-ups and position locaton strategies. The important point to note is that for less than 20 LOS units, all of the scenarios require less than 1 kb/s of position information.

The amount of processing time required by the position location algorithm can be estimated by determining the time required for the floating point operations. An estimate was done for a 16-bit processor with a hardware multiply and divide. With a 6-s update rate, approximately 3 percent of the processors capability would be used for position location. Thus, this type of processor should be able to provide position location without seriously increasing node delay.

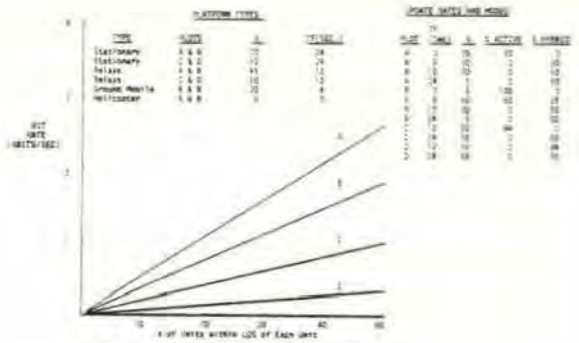


Figure 9. POSLOC Bit Rate Example.

Hardware

In general, the hardware would be impacted in the following three areas:

a. Accurate TOA estimator

b. Accurate TOT control

c. Increased memory

The TOA estimator hardware is determined by the type of waveform used. In some systems, a delay lock loop is used to track the TOA of the incoming signal. In a ground, mobile environment, it is desireable to detect the leading edge of the received multipath signal. Detection of the leading edge may require coherent averaging of the signal in order to increase the signal to noise ratio of the leading multipath component. The TOT can be controlled by retiming the signal immediately before modulation of the IF. A survey of TOA estimator and TOT control techniques indicates that the TOA estimator and TOT controller add to the complexity of the system, but their complexity is not unreasonable. An algorithm supporting two modes of operation and coordinate conversion routines could be stored in less than 8 k 16-bit words. Thus the hardware modifications required are feasible.

Software

The use of range measurements to provide time synchronization and position location requires a significant amount of software. The software tasks would include the following functions:

a. Position location algorithms

   1. Multilateration solution

   2. Kalman filter

b. Interface with the operating system

c. Creation and transmission of position reports

d. Protocols for position reports.

Conclusion

In summary, usable position accuracies can be obtained within a TDMA network. A position location algorithm utilizing multilateration and a Kalman filter linear estimator can track the position and clock offset within each network element. An analysis of this method shows that performance is a complex function of a number of factors. With the tools which have been developed for analyzing performance, the obtainable accuracy can be predicted or the parameters required to meet a performance goal can be chosen, given a particular scenario. The cost of implementing position location in terms of network throughput and hardware complexity should not be unreasonable. Consequently, position location may be a valuable, obtainable addition to a tactical data distribution network.

REFERENCES

1. "Feasibility of Position Location in a UPR Network", Collins Quarterly Technical Report, Pocket Radio Communications, Telecommunications Systems Division, Rockwell International, August 1978.

2. Fried, W. R., "Principles and Simulation of JTIDS Relative Navigation", IEEE Trans. AES, Vol. AES-14, No. 1, January 1978.

3. Snodgrass, R. C., "Recursive Navigation Algorithm for Electronic Grid Integration", The MITRE Corporation Technical Report MTR-2821, April, 1974.

4. Snodgrass, R. C., "Range Determination and Synchrnization in SEEK BUSS", The MITRE Corporation Working Paper WP-5447, February, 1975.

5. Rome, H. J. and Stambaugh, J. S., "Evaluation of the Accuracy of Community Relative Navigation Organization Concepts", Navigation: Journal of the Institute of Navigation, Vol. 24 No. 2, Summer 1977.

6. Ricke, G. G. and Williams, J. R., "Adaptive Tracking Filter for Maneuvering Targets', IEEE Trans. on AES, Vol. AES-14, No. 1 January 1978.

7. Sage, A. P. and Melsa, J. L., Estimation Theory with Applications to Communications and Control, McGraw-Hill Book Company, 1971, pp 376-416.

WHAT LANGUAGE UNDERSTANDING RESEARCH SUGGESTS ABOUT DISTRIBUTED ARTIFICIAL INTELLIGENCE*

Earl D. Sacerdoti
Artificial Intelligence Center
SRI International

Natural language communication requires that dialogue participants know not only the structure of language but also something about the subject matter under discussion and about the processes through which concepts in the subject area may be expressed by means of words, phrases, and sentences in the language. Similarly, a rich dialogue among distributed intelligent processors will require that the processors have a (partial) knowledge base in common and that they employ and understand common groundrules about how to convey portions of that knowledge base within the available communication protocols. Therefore, designers of distributed artificial intelligence (DAI) systems might find useful concepts in the designs of artificial intelligence systems for natural language understanding.

Computer science, as opposed to traditional mathematics, views computation as a process performed with finite resources over time. In an analogous manner, natural language understanding research in artificial intelligence, as opposed to traditional linguistics, is increasingly concerned with communication as a process performed with finite resources over time. DAI systems of any power will certainly also have to view interprocessor communication as a process performed with finite resources over time. The particular perspective of artificial intelligence on the problem of understanding natural language may help us to articulate some issues of importance that must be faced in developing distributed systems that display intelligent behavior.

* The preparation of this paper was supported by the Advanced Research Projects Agency of the Department of Defense. It was prepared in haste, for which I apologize to the reader. I wish to thank Barbara Grosz and Peter Hart for their helpful suggestions. Please don't expect them to stand firmly behind the papers' conclusions, though. For that matter, don't expect me to!

This brief note will attempt to draw an extended analogy between natural language communication and interprocess communication, thereby hopefully identifying in advance some of the gaps that must be filled as the processors that are communicating become increasingly powerful.

A. Introduction

Communications specialists have generally concerned themselves with the format and content of individual classes of interprocess messages, rather than on the ongoing interaction (spanning many instances of many classes of message) through which particular information is communicated. This is analogous to the linguists' traditional concerns with the form and meaning of individual words and sentences. In contrast, AI researchers on language understanding view communication as an activity performed by two or more cooperating parties. This viewpoint on language understanding systems suggests that a crucial issue for DAI is a careful articulation of the processes that underlie participation in dialogue (as opposed to the structuring of the individual interprocess messages and communication protocols).

Below we will characterize three kinds of knowledge that language understanding research suggests must be available to a communicating process. Then we will discuss the activities involved in interpreting and responding to a message.

B. Knowledge about the Subject Domain

Communication between people or processors can be viewed as the incremental sharing and building of their respective knowledge bases. Recent work in computational linguistics has demonstrated the importance to each participant in a dialogue of having a very rich knowledge base that is much more than a static description of the facts that are currently true in the domain of discourse. The knowledge base must encode a dynamic environment consisting of various actors, objects, relationships and events, ordered (or partially ordered) with respect to time. It must describe

not only the environment as it truly is, but also the knowledge and beliefs about the environment that are held by each participant in the dialogue.

C. Knowledge about Context

Natural language is used for communication in a dynamically changing context. An utterance in a dialogue cannot typically be interpreted in isolation; it must be analyzed within the context in which it was produced. An interpretation is influenced by the current state of the environment, by a history of the previous states, by the overall structure and content of the dialogue, by knowing who produced the utterance and for whom it was intended, and by the preceding utterances in the dialogue.

This complex collection of required state information renders genuine comprehension of natural dialogue beyond the current state of the art. It is, however, an extremely efficient means of communicating parsimoniously over a noisy medium when there is sufficient processing power available on both sides of the communications link. By each having all this knowledge about the subject domain and the current context of the interaction, processors can communicate using many fewer bits. Furthermore, because the processors are continually engaged in augmenting and checking a shared knowledge base, errorful transmissions are much more likely to be noticed, and a subdialogue requesting confirmation of the suspicious message can be initiated.

Employing this state information, then, may provide significant additional efficiency and reliability in the communications process. On the other hand, maintaining and exploiting this state information imposes a very significant additional computation load. Researchers exploring the distribution of artificial intelligence capabilities will need to evaluate the tradeoffs between the increased processing required and the enriched communication provided by this approach.

D. Knowledge about Communicating

To enable the rich interactions we have been describing, a third kind of knowledge is needed. This is knowledge about the "rules of the game" of communicating. In typical systems that perform interprocess communication this consists of no more than the encoding of routines that can either create or interpret instances of particular classes of messages. By analogy with natural language understanding systems, communicating processes may also have to know enough about the activity of communication itself to determine when the focus of the communication is shifting, what the current goals of the sender are that lie behind his current transmissions, and when (and why) the current sequence of transmissions is coherent as a whole. The cues for these kinds of information are often encoded in subtle ways in natural language dialogues. Human participants seem able to perform the deductions required to pick up these cues easily. Language understanding systems have been rather poor at this to date. By encoding the cues more explicitly for interprocess communication, the requirement for the individual processors to perform complex deductions can be greatly reduced. This aspect of communication, then, appears to be one that can be incorporated rather easily into DAI systems.

E. Interpreting a Message

In analyzing and interpreting messages expressed in natural language, a variety of kinds of information must be brought to bear. Firstly, there is knowledge about the syntax, how individual words are combined into phrases and how phrases are combined into sentences. Secondly, there is lexical knowledge, about the meanings of individual words and the roles they can take on within larger phrases. Thirdly, there is knowledge about the mapping between the phrases of the input and descriptions of objects in the internal representation of the subject domain. (Strictly speaking, this is what philosophers mean when they refer to semantics, although the term is almost always used in a much wider sense in the literature

of artificial intelligence.) Finally, knowledge about the current state is needed to refine the descriptions of objects into designations of particular entities in the "real world."

While these types of knowledge are listed in order of increasing complexity and difficulty of use, they are not employed in a strictly linear order. Since the analysis at each level affects the confidence in the conclusions drawn at other levels, the overall interpretation is usually built up incrementally with many partial contributions from all levels.

The designer of a system that involves interprocess communication typically builds in to the communicating processes knowledge of the first two sorts described above. He worries about the structure of each message and about the values and meanings of the fields within those messages. The other types of knowledge are "hard-wired" into his programs, and are typically extremely simple. Each message typically has an unambiguous meaning independent of its ordering within the overall dialogue.

This design of a communication protocol is appropriate for situations where the processing cost of sending or receiving each message must be kept low. The use of the semantically oriented kinds of knowledge makes the processing for message transmission very much more expensive. However, if processing at the source and destination is relatively cheap (and we expect this to be the case for DAI systems), the overall cost of the system might be minimized by trading off higher processing requirements for lower bandwidths and higher noise levels.

F. Responding to a Message

We have just sketched the (rather complex) process by which a message might be interpreted by a processor in a DAI system. Once the message is interpreted, it must then be reponded to. As might be expected, this is also a rather complex process for a natural language understanding system, and will probably be complex for a DAI system as well.

The complexity of the response derives mostly from the need to update and maintain the complex knowledge about state information described in Section C above. The response must include:

- a check for the validity of the current assumptions about the state of the other communicating process;
- a check for cues that the sender of the message is shifting the focus of the communication;
- a determination of the overt actions to be taken in response to the message;
- a determination of whether (and under what conditions and when) a return message is requested or required;
- the generation, if needed, of a return message (which will involve encoding sufficient information for the other process to perform the same set of tasks).

As was the case with the task of interpreting messages, responding to messages will be easier for computer-to-computer interactions than for natural language communication. By requiring explicit indications of shifts of focus and, perhaps, an explicit indication of nonstandard or unexpected assumptions, the response to messages can probably be performed with tolerable efficiency.

G. Conclusions

We have speculated on the possible relevance of the AI approach to natural language understanding to the problem of communication between processors in a distributed artificial intelligence system. While the processing required to interpret a natural language is almost certainly far more than a DAI system needs, a simplified form of this processing that performs all or most of the tasks performed in understanding natural language may well be required. This suggests that designers of DAI systems might best design their communications protocols by narrowing down the capabilities required for natural language communication rather than by building up from the traditional, individual-message-oriented approach. The key point I have tried to make is that, to develop a robust distributed system that is able to communicate over a noisy, relatively narrow-bandwidth channel, one must consider not just the

format and content of the individual messages that are used to communicate, but also the role that sequences of messages play in the overall process of communication.

H. Relevant References

1. Cohen, P.R., "On Knowing What to Say: Planning Speech Acts," Ph.D. thesis, University of Toronto, 1978.
2. Grosz, B.J, "Focusing in Dialog," Proc. TINLAP-2, Urbana, Ill., 1978.
3. Lesser, V.R. and Erman, L.D., "A Retrospective View of the HEARSAY-II Architecture," Proc. 5th IJCAI, Cambridge, Mass., 1977.
4. Levy, D.M., "Communicative Goals and Strategies Between Discourse and Syntax," Proc. Symposium on Discourse and Syntax, UCLA, Nov. 1977.
5. Robinson, A.E., "Investigating the Process of Natural Language Communication," SRI AI Center Tech. Note 165, 1978.

APPLICATIONS OF THE CONTRACT NET FRAMEWORK: DISTRIBUTED SENSING

Reid G. Smith and Randall Davis

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

Abstract[1]

We describe the initialization and operation of a distributed sensing system based on the contract net framework. In a departure from earlier systems, task distribution is viewed as a local mutual selection process, a discussion carried on between a node with a task to be executed and a group of nodes that may be able to execute the task. This leads to the use of a control formalism based on a contract metaphor, in which task distribution corresponds to contract negotiation.

1 Distributed Problem Solving: Overview

Distributed problem solving is carried out in a processor architecture in which the individual nodes include memory as well as processing capability. In such a problem solver, control is decentralized, the nodes are loosely coupled (i.e., they spend a far greater percentage of time in computation than in communication with other nodes), they communicate via messages, and they cooperate in the solution of a single overall problem.

A framework for distributed problem solving has been developed that specifies mechanisms for communications, control, and knowledge organization. The framework is based on the human model of experts that cooperate to solve problems by transfer of messages and use a contract negotiation process to distribute tasks to be executed concurrently. In the distributed problem-solving context, the human experts correspond to processor nodes. Each node contains a number of task-specific knowledge-sources (KSs). In this paper we will focus on the communications and control aspects of the framework.

A key problem that must be resolved in a distributed problem solver is how nodes with tasks to be executed find other nodes capable of executing those tasks. We will call this the *connection problem*. In centralized problem solvers it is called the *invocation problem*; that is, which KS to invoke at any given time for the execution of a task. Because *AI* applications do not generally have well-defined algorithms for their solution, *AI* problem solvers need considerable heuristic knowledge to guide them, making the connection problem crucial.

[1] This work has been supported in part by the Advanced Research Projects Agency under contract MDA 903-77-C-0322, and the National Science Foundation under contract MCS 77-02712. It has been carried out on the SUMEX-AIM Computer Facility, supported by the National Institutes of Health under grant RR-00785. Reid Smith is now at the Defence Research Establishment Atlantic, Dartmouth, Nova Scotia, Canada. Randall Davis is now at the MIT Artificial Intelligence Laboratory, Cambridge, Mass.

2 The Contract Net Framework: Communications And Control

The contract net framework includes a *problem-solving protocol* [Smith, 1977] [Smith, 1978b], an extrapolation to the problem-solving level of the standard network communications protocol. This protocol encodes task-independent information that specifies the possible actions and interactions of the nodes of the problem solver and also includes slots for the task-dependent information necessary for the decisions that guide the control. The task-dependent information in the framework is encoded in a *common-internode-language*, understandable to all nodes.

The problem-solving protocol uses an announcement - bid - award sequence of *contract negotiation* to solve the connection problem. It views task distribution as an interactive process, one that that entails a discussion between a node with a task to be executed and nodes that may be able to execute the task. The protocol defines a set of messages that has so far proven adequate for both control and data distribution.

A contract net is a collection of interconnected processor nodes whose interactions are governed by a problem-solving protocol based on the contract metaphor. Each node in the net operates asynchronously and with relative autonomy. The execution of an individual task is handled as a contract. A node that generates a task advertises existence of that task to the other nodes in the net with a ***task announcement***, then acts as the ***manager*** of that task for its duration. In the absence of any information about the specific capabilities of the other nodes in the net, the manager is forced to issue a *general broadcast* to all nodes. If, however, the manager possesses some knowledge about which of the other nodes in the net are likely candidates, then it can issue a *limited broadcast* to just those candidates. Finally, if the manager knows exactly which of the other nodes is appropriate, then it can issue a *point-to-point* announcement.[2] As work on the problem progresses, many such task announcements will be made by various managers.

Nodes in the net have been listening to the task announcements, and have been evaluating their own level of interest in each task with respect to their specialized hardware and software resources. When a task is found to be of sufficient interest, a node submits a ***bid***. A bid indicates the capabilities of the bidder that are relevant to

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

execution of the announced task. A manager may receive several such bids in response to a single task announcement; based on the information in the bids, it selects one (or several) node(s) for execution of the task. The selection is communicated to the successful bidder(s) through an ***award*** message. These selected nodes assume responsibility for execution of the task, and each is called a ***contractor*** for that task.

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

The contract negotiation process is expedited by three forms of task-dependent information contained in a task announcement. An ***eligibility specification*** lists the criteria that a node must meet to be eligible to submit a bid. This specification reduces message traffic by pruning nodes whose bids would be clearly unacceptable. A ***task abstraction*** is a brief description of the task to be executed, and allows a potential contractor to evaluate its level of interest in executing this task relative to others that are available. An abstraction is used rather than a complete description in order to reduce the length of the message (and hence message traffic).[3] Finally, a ***bid specification*** details the expected form of a bid for that task. It enables a potential contractor to transmit a bid that contains only a brief specification of its capabilities that are relevant to the task (this specification is called a ***node abstraction***), rather than a complete description. This both simplifies the task of the manager in evaluating bids, and further reduces message traffic.[4]

Contracts are queued locally by the node that generates them until they can be awarded. If no bids are received for a contract by the time an ***expiration time*** (included in the task announcement) has passed, then the contract is re-announced. This process is repeated until the contract can be awarded.[5]

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

The manager may terminate the execution of contracts as necessary (with a ***termination*** message), and further (sub)contracts may be let in turn as required by the size of a contract or by a requirement for special expertise or data that the contractor does not have.

It is important to note that individual nodes are not designated a priori as managers or contractors. These are only roles, and any node can take on either role. During the course of problem solving, a particular node normally takes on both roles (perhaps even simultaneously for different contracts). This leads to more efficient utilization of nodes, as compared, for example, to schemes that do not allow nodes that have contracted out subtasks to take on other tasks while they are waiting for results. Individual nodes, then, are not statically tied to the control hierarchy.

Note also that the idea of transfer of expertise between nodes (via transfer of procedures or data) can be readily handled by the protocol. It can be handled as a standard contract between a node that announces (in effect) *I need the code for <procedure-description>*, and a node that bids on the task by indicating that it has the required information.

To review, the normal method of negotiating a contract is for a node (called the manager for a task) to issue a task announcement. Many such announcements are made over the course of time. Other nodes are listening and submit bids on those announcements for which they are suited. The managers evaluate the bids and award contracts to the most suitable nodes (which are then called contractors for the awarded tasks).

The normal contract negotiation process can be simplified in some instances, with a resulting enhancement in the efficiency of the protocol. If a manager knows exactly which node is appropriate for execution of a task, a ***directed contract*** can be awarded. This differs from the ***announced contract*** in that no announcement is made, and no bids are submitted. Instead, an award is made directly. In such cases, nodes awarded contracts must acknowledge receipt and have the option of refusal.

The protocol has also been designed to allow a reversal of the normal negotiation process. When the processing load on the net is high, most task announcements will not be answered with bids because all nodes will be already busy. Hence the protocol includes a ***node availability announcement*** message. Such a message can be issued by an idle node. It is an invitation for managers to send task announcements or directed contracts to that node.

Finally, for tasks that amount to simple requests for information, a contract may not be appropriate. In such cases, a request - response sequence can be used without further embellishment. Such messages (that aid in the distribution of data as opposed to control) are implemented as ***request*** and ***information*** messages. The request message is used to encode straightforward requests for information for which contracting is unnecessary. The information message is used both as a response to a request message and as a general data transfer message.

---

[3] One component of the abstraction is the *task type*, or generic classification of the task.

[4] The information that makes up the eligibility specification, task abstraction, and bid specification for any given example must be supplied by the applications programmer. In Section 3 we will see examples of this type of information.

[5] The is a simplified version of the actual process (and does not work in the case of a task that cannot be executed due, for example, to lack of sufficient data); see [Smith, 1978c] for the complete description.

3 Distributed Sensing

In this section, we demonstrate the use of the contract net framework in the solution of a problem in area surveillance, such as is encountered in ship or air traffic control. The example will help to demonstrate the contract net approach to communications and control.

We consider the operation of a network of nodes, each having either sensing or processing capabilities and all spread throughout a relatively large geographic area. Such a network is called a ***Distributed Sensing System*** (DSS).

The primary aim of the system is rapid, reliable, accurate, and low-cost analysis of the traffic in a designated area. This analysis involves detection, classification, and tracking of vehicles; that is, the solution to the problem is a dynamic map of traffic in the area. Construction and maintenance of such a map requires the interpretation and integration of a large quantity of sensory information received by the collection of sensor elements.

There are many trade-offs involved in the design of a DSS architecture. We present only one possible approach that considers a limited number of these trade-offs.[6] The primary intent of this example is to demonstrate the contract net approach, and we therefore focus on the initialization and communications aspects of the DSS. For a discussion of other aspects of this problem see [Nii, 1978].

3.1 Hardware

All communication in the DSS is assumed to take place over a broadcast channel (using for example, packet radio techniques [Kahn, 1975]). The nodes are assumed to be in fixed positions known to themselves but not known a priori to other nodes in the net. Each node has one of two capabilities: sensing or processing. The sensing capability includes low-level signal analysis and feature extraction.[7] We assume that a variety of sensor types exist in the DSS, that the sensors are widely spaced, and that there is some overlap in sensor area coverage. Nodes with processing capability supply the computational power necessary to effect the high-level analysis and control in the net. They are not necessarily near the sensors whose data they process.

A DSS may have several functions, ranging from analysis of vehicle data in the overall area of coverage to control over the courses and speeds of those vehicles. We consider here the analysis function. In this case the overall area map must be integrated at one node in the system (it could also be integrated by an agent outside the system, like a monitoring aircraft). We therefore distinguish one processor node as the monitor node. Its function is to begin the initialization of the DSS, and integrate the overall area map for communication to an agent outside the DSS. We will see that it does not correspond to a central controller.

[6] Further discussion of the background issues inherent in DSS design is presented in [Smith, 1978a].

[7] In a real DSS, it is likely that sensors and low-level signal analysis devices would not be considered as statically connected parts, as it is often the case that many different types of analysis are applied to the output of a single sensor, or to that of groups of sensors taken together.

Figure 3.1 is a schematic representation of a DSS. Processing nodes are shown in black, and sensing nodes in white. The monitor node is shown in white with an "M" in the middle.

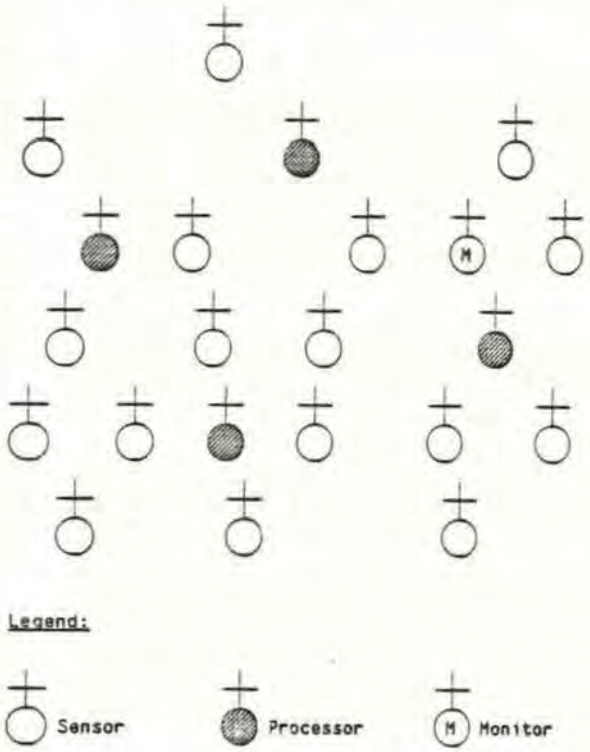


Figure 3.1. A Distributed Sensing System.

3.2 Data And Task Hierarchy

The DSS must integrate a large quantity of signal data, reducing it and transforming it into a symbolic form meaningful to and useful to a human decision maker. We view this process as occurring in several stages, which together form a data hierarchy (Figure 3.2). The hierarchy offers an overview of DSS functions and suggests a task partitioning suitable for a contract net approach. A particular node in the DSS handles data at only one level of the data hierarchy at any given moment, and communicates with nodes at other levels of the hierarchy.

Figure 3.2. Data Hierarchy.

For purposes of this example, the only form of signal processing we consider is narrow band spectral analysis. The *signal* has the following features: frequency, time of detection, strength, characteristics (e.g., increasing signal strength), name and position of the detecting node, and name, type, and orientation of the detecting sensor.[8]

Signals are formed into *signal groups* at the second level of the data hierarchy. A signal group is a collection of related signals.[9] For this example, the signal groups have the following features: the fundamental frequency of the group, the time of group formation, and the features of the signals in the group (as above).

The next level of the hierarchy is the description of the *vehicle*. It has one or more signal groups associated with it and is further specified by position, speed, course, and type.[10] Position can be established by triangulation, using matching groups detected by several sensors with different positions and orientations. Speed and course must be established over time by tracking.

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

The final level is the complete or *overall area map*. In this example, the map is integrated from the individual area maps by the monitor node.

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

The area contractors also manage several *vehicle* contractors that are responsible for integration of information associated with individual vehicles. Each of these contractors manages: a *classification* contractor that determines vehicle type; a *localization* contractor, that determines vehicle position; and a *tracking* contractor, that tracks the vehicle as it passes through the area.[11]

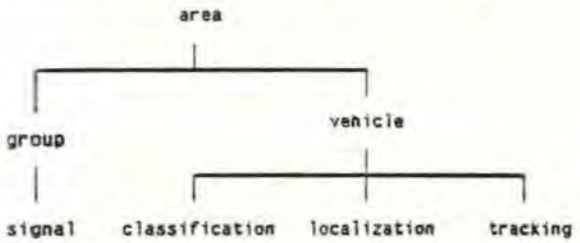


Figure 3.3. Area Task Partitioning.

Note that this particular partitioning of tasks is only one of many possibilities that might be specified by the system designer.

3.3 Contract Net Implementation

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

3.3.1 Initialization

The monitor node is responsible for initialization of the DSS and for formation of the overall map. It must first select nodes to be area contractors and partition the system's span of coverage into areas, based on the positions of those selected nodes. For purposes of illustration we assume that the monitor node knows the names of nodes that are potential area contractors, but it must establish their positions in order to partition the overall span of coverage. Hence, it begins by announcing contracts for formation of area maps of the traffic. Because the monitor node knows the names of potential area contractors, it can avoid a general broadcast and can instead use a focused addressing scheme. The announcement contains the three components described in Section 2: a task abstraction, an eligibility specification, and a bid specification. The task abstraction is simply the task type, and the eligibility specification is blank (because the monitor node knows which other nodes are potential contractors and addresses them directly). The bid specification is of primary interest for this task. It informs a prospective area contractor to respond with its position. Remember that the purpose of a bid specification is to enable a manager to select, from all of the bidders, the most

[8] The frequency spectrum of noise radiated by a vehicle typically contains narrow band signal components that are caused by rotating machinery associated with the vehicle (e.g., engines or generators). The frequencies of such signals are correlated with the type of rotating machine and its speed of rotation. They are indicators of the classification of the vehicle. Narrowband signals also undergo shifts in frequency, due to doppler effect, or instability and change in the speed of rotation of the associated machine. Alterations in signal strength also occur as a result of propagation conditions and variations in the distance between the vehicle and the sensor.

[9] A signal group that is often used to integrate narrow band signal data, for example, is the harmonic set, a group of signals that are harmonically related (i.e., the frequency of each signal in the group is an integral multiple of the lowest, or fundamental frequency). A single rotating machine often gives rise to several narrow band signals that form a harmonic set.

[10] For simplicity, we have ignored a level in the hierarchy that can be called *component sources of signal*, as in [Nii, 1978]. At this level, a DSS would normally try to attribute signals and signal groups to particular pieces of machinery associated with a vehicle.

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

appropriate nodes to execute a contract. Node position is the information required by the monitor node to make that selection. Given that information, the monitor node can partition the overall span of coverage into approximately equal-sized areas, and select a subset of the bidders to be area contractors. Having decided upon a partitioning, the monitor node broadcasts an information message to the other nodes in the system. This message defines the names and specifications (in terms of latitude and longitude ranges) of the individual areas. Each selected area contractor is then informed of its area of responsibility in an award message.[12]

The area contractors' purpose is to integrate vehicle data into area maps. They must first establish the existence of vehicles on the basis of signal group data. Therefore, each area contractor solicits other nodes to provide signal group data. In the absence of any information about which nodes are suitable, each area contractor announces the task using a general broadcast. The task abstraction in these announcements is the type of task. The eligibility specification is the area for which the individual area contractor is responsible: that is, a node is only eligible to bid on this task if it is in the same area as the announcing area contractor. This restriction helps to prevent a case in which a signal group contractor is so far away from its manager that reliable communication is difficult to achieve. The bid specification is again node position. Potential group contractors respond with their respective positions, and, based on this information, the area contractors award signal group contracts to nodes in their areas of responsibility.

The signal group contractors' task is to integrate signal data from sensor nodes into signal groups. Therefore, they must first find nodes that will provide raw signal data. This is done with signal task announcements. The eligibility specification in these announcements indicates that only those nodes located in the same area as the announcer and having sensing capabilities should bid on this task. The task abstraction indicates the task type and position of an individual signal group contractor. This information assists potential signal contractors in determining the group contractors to which they should respond.[13]

The potential signal contractors listen to the task announcements from the various group contractors. They respond to the nearest group contractor with a bid that supplies their position and a description of their sensors. The group contractors use this information to select a set of bidders that covers their immediate vicinity with a suitable variety of sensors, and then award signal contracts on this basis. The awards specify the sensors that each signal contractor is to use to provide raw data to its managing group contractor. Figure 3.4 depicts the exchange between one group contractor (the black node), and several potential signal contractors (the white nodes). Successful bidders are connected by solid lines to the group contractor and unsuccessful bidders are connected by dashed lines.

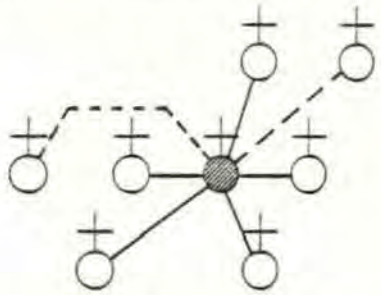


Figure 3.4. Signal Contract Negotiation.

There are some potential problems of asynchrony in the receipt of announcements from group contractors. A potential contractor for signal tasks must determine the group contractor that is closest to it by listening to several signal task announcements. The potential signal contractors use the expiration times of the announcements (i.e., the times after which no further bids will be accepted) as a guide to the length of time during which they can listen to announcements before submitting a bid. In the best case, the expiration times are long enough to allow announcements from group contractors to reach all local potential signal contractors so that optimum partitioning can be achieved. In the worst case, however, a potential signal contractor may submit a bid to a group contractor that is not the closest one to it (because the task announcement of the closest group contractor is not received until after a bid has already been submitted to another group contractor). The result is a sub-optimal partitioning.

The signal contract is a good example of the contract negotiation process, illustrating how the matching of contractors to managers is an interactive process. It involves a mutual decision based on local processing by both the group contractors and the potential signal contractors. The potential signal contractors base their decision on a distance metric and respond to the closest manager. The group contractors use the number of sensors and distribution of sensor types observed in the bids to select a set of signal contractors that ensures that every area is covered by every kind of sensor. Thus each party to the contract evaluates the proposals made by the other, using a different evaluation function, and a task distribution agreement is completed via mutual selection.

Reviewing the status of the DSS, we have a single monitor node which manages several area contractors. Each area contractor manages several group contractors, and each group contractor manages several signal contractors. The data initially flows from the bottom to the top of this hierarchy. The signal contractors supply raw signal data; each group contractor integrates the raw data from several signal contractors to form a signal group, and these groups are passed along to the area contractors, which eventually form area maps by integrating information based on the data from several group contractors. All the area maps are then passed to the monitor which forms the final traffic map.

As we have noted, in this example one area contractor manages several group contractors and each group contractor in turn manages several signal contractors. It is possible, however, that a single group contractor should

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

[13] The signal task is to detect signals and report them to a signal group contractor in a particular position. Hence the position of the group contractor is reasonable as part of the abstraction for the task.

supply information to several area contractors, and a single signal contractor should supply information to several group contractors. It may be useful, for instance, to have a particular group contractor near an area boundary report to the area contractors on both sides of the boundary. This is easily accommodated within our framework.

3.3.2 Comments On The DSS Organization

We have taken a top-down, distributed approach to initializing the DSS. An alternative approach might involve acquisition of the positions of all nodes at a very early stage, followed by area definition, award of area contracts, and so on. This would involve a more global approach to the problem of initialization, using a single node that initialized the net by gathering together all the necessary data. We have not pursued this approach for several reasons, primarily because it would tell us little about solving problems in a distributed manner. An underlying theme of this research is a search for ways in which to effect distributed problem solving--rather than ways to do traditional problem solving in a distributed architecture.

Moreover, there are two practical difficulties with the global approach. First, it concentrates a large amount of message traffic and processing at a single node (say the monitor node) because such a node would be responsible for accepting position messages from every other node in the net. Second, it may not be possible for any single node to communicate directly with all other nodes in a widely separated collection. This would mean that either indirect routing of messages would be required for communication, or that each of the nodes would require powerful transmitters. This is one of the advantages of the distributed and dynamically-defined organization we have adopted: only pairs of nodes that are close together enough to communicate directly are linked together with contracts.

3.3.3 Operation

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

When a signal is detected or when a change occurs in the features of a known signal, the detecting signal contractor reports this fact to its manager (a group contractor) (Figure 3.5). This node, in turn, attempts to integrate the information into an existing signal group or to form a new signal group.

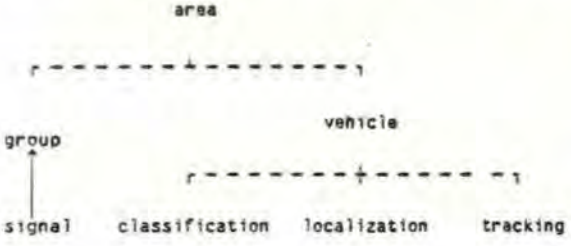


Figure 3.5. Signal Contract Reporting.

A group contractor reports the existence of a new signal group to its manager (an area contractor) (Figure 3.6).

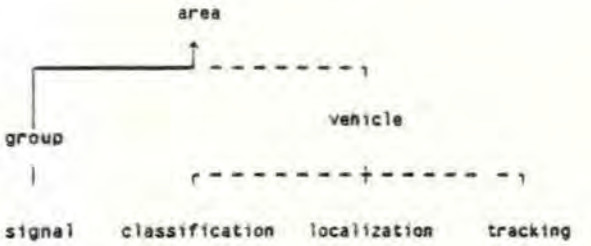


Figure 3.6. Group Contract Reporting.

Whenever a new group is detected, the managing area contractor attempts to find a node to execute a vehicle contract (Figure 3.7). The task of a vehicle contractor is to classify, localize, and track the vehicle associated with the signal group. Since a newly detected signal group may be attributable to a known vehicle, the area contractor first requests from the existing collection of vehicle contractors a measure of confidence in the fact that the new group is attributable to one of the known vehicles. Based on these responses, the area contractor either starts up a new vehicle contractor or augments the existing contract of the appropriate existing vehicle contractor, with the task of making certain that the new group corresponds to a known vehicle. This may entail, for example, the gathering of new data via the adjustment of sensors or contracts to new sensor nodes.

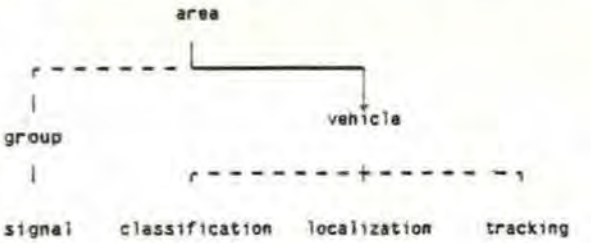


Figure 3.7. Vehicle Contract Initiation.

The form of the signal group confirmation request demonstrates a tradeoff that arises in many distributed problem solving applications--a tradeoff between communication and local processing. The area contractor has the option of transmitting to the existing vehicle contractors either the complete signal group or an abstraction of it (e.g., its fundamental frequency). In either case, the response to the request is a measure of the individual vehicle contractor's confidence that the group corresponds to a vehicle it knows about. If a vehicle contractor returns a high confidence measure in its bid, then in the first case (complete signal group announced) the area contractor has the information it wants. In the second case, however, the area contractor must still transmit the complete signal group description and await a further report.

The first approach has the disadvantage of using up more local processing time in each of the vehicle contractors than does the second, in that the original request message is

more complex. The question of interest is, *Under what conditions should a complete signal group description be announced instead of an abstraction?* The answer appears to depend on the quality of the abstraction. If the abstraction is good enough that the vehicle contractors are able to make definitive statements on the basis of its use, then the second approach is likely better than the first, in that it minimizes local processing time. On the other hand, if the abstraction does not allow definitive statements to be made, then its use will result in increased message traffic and local processing time, since the area contractor will not be certain as to the best course of action as a result of uncertain responses from the vehicle contractors.

The vehicle contractor then makes two task announcements: vehicle classification and vehicle localization. The task abstraction of the classification task announcement is a list of the fundamental frequencies of the signal groups currently associated with the vehicle. This information may help a potential classification contractor select an appropriate task (a contractor may, for example, already be familiar with vehicles that have signal groups with the announced fundamental frequencies). The eligibility specification is blank. The bid specification indicates that a bidder should respond with a tentative classification and an associated confidence measure. This measure is used to select a classification contractor--the bidder with the highest confidence is chosen. The award is the complete current description of the vehicle.

A classification contractor may be able to classify directly, given the signal group information; or, on the other hand, it may require more data, in which case it can communicate directly with the appropriate sensor nodes (Figure 3.8).

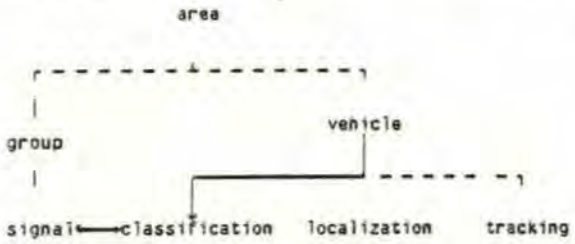


Figure 3.8. Classification Contract Communication.

The task abstraction for a localization task announcement (Figure 3.9) is a list of positions of the nodes that have detected a vehicle. The eligibility specification and bid specification are blank. The bid is simply an affirmative response to the announcement and the contract is awarded to the first bidder, which does the required triangulation to obtain the position of the vehicle.

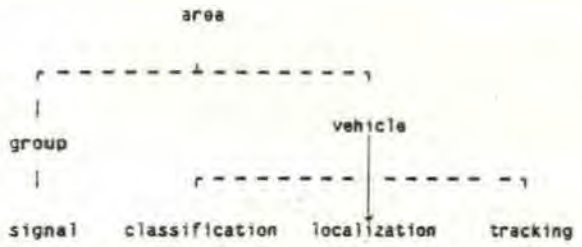


Figure 3.9. Localization Contract Initiation.

Once the vehicle has been localized, it must be tracked. This is handled by the vehicle contractor by entering into follow-up localization contracts from time to time and using the results to update its vehicle description (Figure 3.10). As an alternative, the area contractor could award separate tracking contracts. The decision as to which method to use depends on loading and communications. If, for example, the area contractor is very busy with integration of data from many group contractors, then it seems more appropriate to isolate it from the additional load of tracking contracts. If, on the other hand, the area contractor is not overly busy, then we can let it handle updated vehicle contracts, taking advantage of the fact that it is in the best position to integrate the results and co-ordinate the efforts of multiple tracking contractors. In this example, we assume that the management load would be too large for the area contractor.

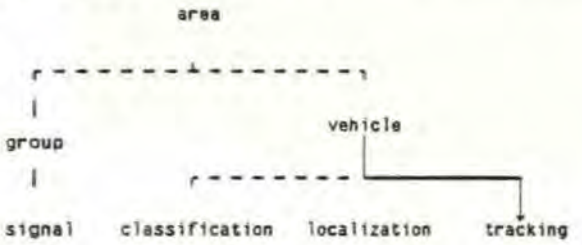


Figure 3.10. Tracking Contract Initiation.

There are a variety of other issues that must be considered in the design and operation of a real distributed sensing system. Most of these are quite specific to the DSS application, and would take us away from our primary concern with the use of the contract net framework. One issue, however, presents an interesting example of the relative utility of the use of contracting compared to the use of information messages. Consider the case of a vehicle that is moving from the area of one area contractor to that of another. How is responsibility for tracking the vehicle to be transferred?

There are two possibilities. First, the area contractor that is currently responsible for the vehicle could send out an information message to neighboring area contractors. This message would serve to set up expectations in the neighboring area contractors that a vehicle was about to

pass into one of their areas. The processing is fully decentralized in this approach.

Second, the area contractor currently responsible for the vehicle could send a report to its manager (the monitor node) that contains the same information. The monitor node could then award a new contract to one of the neighboring area contractors to handle the vehicle when it passes out of the original area. This is a hierarchical control approach to the problem. It entails more (centralized) processing by the monitor node.

With the first approach there is no guarantee that another area contractor will pick up on the information message and immediately take responsibility for the vehicle, where the second approach does provide such a guarantee. The lack of guarantee is not generally serious, but may result in excess processing because the vehicle must be re-detected in the new area, classified, localized, and so on. The tradeoff is thus one of more processing by the monitor node against (possibly) more processing by nodes in the new area. In the simulation we have used the hierarchical control approach because of its guarantee of transfer of responsibility.

4 Conclusion

The use of the contract net framework in the DSS enables the implementation of a dynamic configuration, depending on the actual positions of sensor and processor nodes and the ease with which communication can be established. Such a configuration offers a significant operational improvement over a static a priori configuration: specifically, it ensures that nodes that must cooperate for the solution of the area surveillance problem are able to communicate with each other. This avoids the necessity for either indirect routing of messages or powerful transmitters for all nodes.

The distributed control enabled by the framework enhances both reliability and equalization of the processing load. It is also possible to recover from the failure of nodes, because there are explicit links between nodes that share responsibility for execution of tasks (managers and contractors). The failure of a signal contractor, for example, can be detected by its manager, and the contract for which it was responsible can be re-announced and awarded to another node.

The framework is also advantageous for this problem because it enables addition of new nodes to the net, even after operation has commenced. This is possible for two reasons. First, the nodes communicate in a common protocol. This protocol enables a new node to interpret the task-independent portions of messages (e.g., that a particular message specifies a task to be executed). Second, the protocol is augmented by a common internode language. This enables a new node to identify, for example, the specific information that it must have to execute a particular task. The protocol and language also enable a new node to request this information from other nodes in the net.

We have shown the effectiveness of interactive mutual decisions (by those with tasks to be executed and those in a position to execute the tasks) in distributing tasks throughout the system. This was used, for example, in setting up signal contracts. The potential signal contractors used the information in the signal task announcements to select the closest managers to which to respond. The managers in turn used the number of sensors and the distribution of sensors in the returned bids to make a final selection of signal contractors.

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

The contract net framework is in general applicable to problems that use a hierarchy of tasks and levels of data abstraction. The manager-contractor structure provides a natural way to effect hierarchical control (in the distributed case, it's actually concurrent hierarchical control), and the managers at each level in the hierarchy are an appropriate place for data integration and abstraction. It should also be noted that the control hierarchies in the contract net framework are not simple vertical hierarchies, but are the more complex generalized hierarchies discussed by [Simon, 1969]. The manager-contractor links are not the only means of communication. Nodes are able to communicate horizontally with related-contractors or indeed any other nodes in the net, as we saw in the DSS example, where classification contractors were able to communicate directly with signal contractors.

The announcement - bid - award sequence of contract negotiation enables more information, and more complex information to be transferred in both directions (between caller and respondent) before KS-invocation occurs. The computation devoted to the selection process, based on the information transfer noted above, is more extensive and more complex than that used in traditional approaches, and is *local* in the sense that selection is associated with and specific to an individual KS (rather than embodied in, say, a global evaluation function). As a result, the framework is most useful when the specific KS to be invoked at any time is not known a priori, and when specific expertise is required.

It also follows that the framework is also primarily applicable to domains in which the subtasks are large (in the loose-coupling sense), and in which it is worthwhile to expend a potentially non-trivial amount of computation and communication to invoke the best KSs for each subtask, so as to prevent backtracking as much as possible.

Appendix A

DSS Sample Messages

This appendix includes abbreviated sample messages for the signal task in the DSS example. For brevity, the messages shown contain only the information mentioned in Section 2. Terms written in upper case are included in the core internode language, while terms written in lower case are specific to the DSS application.

Italicized statements are commentary about the content and sequence of messages.

<*Announcements of the following form are transmitted by the various group contractors.*>

To: * <"*" *indicates a general broadcast.*>
From: node-sg1
Type: task announcement
Contract: s
Message:
<*Needed - signal data for traffic. My position is p. If in possession of sensors, and located in area A, respond with position, and type and number of sensors.*>

Task Abstraction:
TASK TYPE signal
NODE NAME sg1 POSITION p
Eligibility Specification:
MUST HAVE DEVICE TYPE sensor
MUST HAVE NODE NAME SELF POSITION area A
Bid Specification:
NODE NAME SELF POSITION
EVERY DEVICE TYPE sensor TYPE NUMBER

<*Nodes with sensors respond to the nearest group contractor.*>

To: node-sg1
From: node-s1
Type: bid
Contract: s
Message:
Node Abstraction:
NODE NAME s1 POSITION q
sensor TYPE S NUMBER 3
sensor TYPE T NUMBER 1

<*Several similar awards are transmitted.*>

To: node-s1
From: node-sg1
Type: award
Contract: s
Message:
<*Report signals. Use sensors S1 and S2.*>
Task Specification:
sensor NAME S1
sensor NAME S2

References

[Kahn, 1975]
R. E. Kahn, The Organization Of Computer Resources Into A Packet Radio Network. *NCC Proceedings*, Vol. 44, Montvale, N. J.: AFIPS Press, 1975. Pp. 177-186.

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

[Simon, 1969]
H. A. Simon, *The Sciences Of The Artificial*. Cambridge, Mass.: MIT Press, 1969.

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

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

[Smith, 1978b]
R. G. Smith and R. Davis, Distributed Problem Solving: The Contract Net Approach. *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence*, Toronto, Canada, July 1978, pp. 278-287.

[Smith, 1978c]
R. G. Smith, *A Framework For Problem Solving In A Distributed Processing Environment*. Ph.D. Dissertation, Dept. of Electrical Engineering, Stanford University, December, 1978.

FUNCTIONALLY ACCURATE DISTRIBUTED PROBLEM SOLVING SYSTEMS

Victor R. Lesser and Daniel D. Corkill

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

1. INTRODUCTION AND OUR APPROACH

There is a large class of applications that seem naturally suited to the use of distributed processing hardware for which adequate distributed problem solving methodologies have not yet been developed. 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) where the development and execution of a distributed control strategy based on a non-local view of sensory data is required. In these applications, a distributed architecture which locates processing capability at the sensor site and which requires only limited communication among processors is especially advantageous and perhaps is the only practical way to perform the processing. Sensor networks (e.g., low flying aircraft, hydrophones, etc.), distributed command and control systems, air traffic control, automotive traffic light control, inventory control (e.g., car rentals), power network grids, and tasks involving mobile robots are all examples of this type of application.

The central theme of our research is that the exact emulation of a centralized approach to these applications in distributed environments is, in general, inappropriate due to the high communication and synchronization costs required to maintain sufficient and consistent local views of the overall problem solving data base. However, if one is led to emulating the centralized approach only approximately, inter-node communication can be reduced, but at the risk of inconsistency and incompleteness of local views and the possibility of unnecessary, redundant, or incorrect processing. The key issue is whether an acceptable design exists which has sufficiently low communication costs and which still allows the overall system performance to be accurate and efficient enough to satisfy the task requirements.

The long range goal of our research is to develop a general theory for "functionally accurate" problem solving structures in which all intermediate aspects of the computation are required to be correct and consistent.

Such a general theory is also important to the implementation of complex applications in centralized environments. These applications are often organized in the form of a collection of independent modules. In such a structure it is often conceptually difficult to develop and expensive to maintain a complete and consistent centralized problem solving data base with which the modules interact. A theory which permits relaxation of completeness and consistency requirements would be a significant aid in the development and maintenance of these systems.

One example of algorithmic structures which exhibit functionally accurate behavior are those that involve the incremental aggregation of partial results into an overall solution. "Knowledge-based" Artificial Intelligence (AI) interpretation algorithms developed recently for speech, image, and signal interpretation applications [Erman and Lesser 1975, Barrow and Tenebaum 1976, Rosenfeld, Hummel, and Zucker 1976, Rubin and Reddy 1977] use this incremental aggregation approach. Errors and uncertainty in input data and from incomplete or incorrect knowledge are resolved as an integral part of the interpretation process by testing for inconsistency among partial results. This is in contrast to more conventional techniques for error recovery in which errors are handled as exceptional conditions, requiring processing outside the normal problem solving strategy.

The ability to handle errors within the normal problem solving process itself creates the possibility that these systems will also handle the additional uncertainty introduced by a distributed decomposition: the restricted and inconsistent local view of the overall problem solving data base at each node. To the degree that systems can handle these problems, the need for a complete and consistent local view of the overall data base and for explicit synchronization to maintain such a view is reduced.

Preliminary research in the development of functionally accurate cooperative distributed problem solving techniques (discussed in Section 3) has produced encouraging results which we feel justify further explorations. As outlined in Section 4 on current research directions, we feel generic research in the areas of distributed search and distributed planning will be important for the development of a general theory of functionally accurate cooperative distributed systems.

2. STATE OF THE ART AND OUR APPROACH

Most distributed processing systems work on

local data bases which contain exact copies of appropriate portions of the overall problem solving data base. We call this type of distributed processing decomposition nearly autonomous, because each processor usually has the information it requires to complete processing; a processor rarely needs the assistance of another processor in carrying out its problem solving function.

In nearly autonomous systems, the problem solving data base is distributed in such a way that the centralized problem solving algorithms are not significantly perturbed. Therefore, the basic algorithms and control structures which are appropriate in a centralized environment do not need to be modified (decomposed) for the distributed environment, but rather can be 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. This results in a two-level structure in which the more complex distributed control structures are superimposed on the basic computational algorithms needed to perform the application processing [Eswanan et al 1976, Thomas 1976, Lampson and Sturgis 1977, Peebles 1977].

An alternative and new approach to structuring distributed problem solving systems is the development of functionally accurate cooperative distributed systems. In cooperative distributed systems, additional complexity over the nearly-autonomous structure is required because the algorithms and control structures are decomposed to operate on local data bases which are incomplete and possibly errorful. In order to resolve the uncertainties in these incomplete local data bases, processors must exchange partial results with each other. Since new information may also be based on processing which used incomplete or incorrect data, an iterative, coroutine type of processor interaction is required. 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 the data they need and produce, forming a "cooperative network of tasks" which collectively define the processing needed to solve the overall task. This perspective shifts the view from that of a system distributed over a network to that of a functionally accurate network of cooperating systems, each of which can perform significant local processing on incomplete and inconsistent information. From this viewpoint, the key issue becomes how to introduce new communication paths to handle the inter-node interactions needed for sharing the information required to correct for incomplete and inconsistent local views.

This perspective is essentially identical to the theory of "nearly decomposable systems" devised by Simon [1962, 1969] to describe complex organizational structures. The term "nearly decomposable" emphasizes the fact that systems can be decomposed naturally into clusters which have a high degree of intra-cluster interaction and a lower degree of inter-cluster interaction.

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.

No software technology for building cooperative distributed problem solving systems which exhibit functionally accurate behavior 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 and inconsistent data.

3. PAST AND PRESENT WORK

Under previous support from ARPA [2] and current support from NSF, a number of functionally accurate computational techniques (e.g., relaxation and distributed hypothesize-and-test ala Hearsay II) developed in the framework of know-

---

[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, Schantz, and Thomas 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] This was joint work with Lee Erman at Carnegie-Mellon University.

ledge-based AI systems are being suitably modified for use in distributed problem solving environments. In order to test whether these modified techniques have relevance to distributed problem solving, a number of distributed applications using these techniques are being developed.

3.1 DISTRIBUTED AUTOMOTIVE TRAFFIC LIGHT CONTROL

One study has concentrated on investigating the suitability of relaxation as the basis of a cooperative distributed approach to automotive traffic light control. Parallel relaxation has been explored in previous work in image processing, but to our knowledge it has not been explored in distributed domains which have ill-behaved compatability relationships between nodes and non-localized interactions among nodes in the network.

A test program was written which simulates a distributed relaxation version of traffic control in which the knowledge applied at each node is similar to that used in a standard centralized traffic control system called SIGOP-II [Leiberman and Woo 1976]. The simulations have produced encouraging results for a number of test cases. However, in a number of cases the parallel scheme does not perform as well as the centralized version, resulting in slow convergence, convergence to near optimal solutions, or oscillation.

These problems are directly attributable to changes in the centralized version, a global node ordering for the relaxation is precomputed using a maximal spanning tree based on the traffic volume at each node. This ordering is very important in reducing the effects of ill-behaved compatability relationships between nodes and non-localized interactions among nodes. Current research is aimed at introducing, in a distributed way, non-local coordination between nodes to eliminate these problems. We are examining such techniques as multi-level relaxation and distributed versions of the maximal spanning tree heuristics.

A more detailed discussion of this research is contained in a forthcoming report by Brooks and Lesser [1978].

3.2 AN EXPERIMENT IN DISTRIBUTED INTERPRETATION

A second study concentrated on applying the Hearsay-II architecture to the distributed 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 consistent interpretation of "what is happening" in the environment being sensed.

Our approach to decomposing Hearsay-II in the packet-radio network environment is based on the following premise: the only cost-effective way to decompose Hearsay-II is to modify the 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 comes from the relaxation paradigm, where the current state (results) of a processor node is accessed only by nodes that are neighboring 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 Hearsay-II blackboard. Not all processor nodes have to immediately receive 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 form 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-invocation is achieved through the use of threshold values (part of the focus of control data base) which are modified by the same spreading excitation scheme used to transmit important information.

Using these two ideas, we have developed a system which permits effective inter-node 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.

A set of experiments to determine how the distributed decomposition of Hearsay-II affects its problem solving behavior is now being analyzed. Aspects of behavior studied include the accuracy of the interpretation, the time required for interpretation, the amount of inter-node communication required, and the robustness of the system in the face of communication hardware malfunction (i.e., communication without positive acknowledgment). These experiments were only in part simulations, since they used an actual interpretation system analyzing real data: the Hearsay-II speech understanding system [Erman and Lesser 1978].

Tentative results of these experiments 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 knowledge sources, 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 (fewer than 50% of the locally generated hypotheses need to be transmitted). In addition, the distributed system con-

tinues 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).

A forthcoming paper by Lesser and Erman [1978] further details this research.

4. CURRENT RESEARCH DIRECTIONS

Our exploratory research in the development of functionally accurate cooperative distributed problem solving techniques has produced promising results. This work has highlighted many key design issues that must be dealt with in the development of a general theory for functionally accurate cooperative distributed systems. We believe that sufficient intuition has been obtained in the initial stages of this research to direct the investigation to a set of more generic research topics in knowledge-based AI problem solving and in organizational theory. The results of this research will be important in shaping a general theory.

Further investigation of specific applications remains appropriate to the continuing development of intuitions regarding the key design issues. However, the selection of these should be done in a more directed way, relating directly to generic research topics. The emphasis of our current research is on an investigation of the following two generic topics.

4.1 DISTRIBUTED SEARCH

Experiences with the two distributed interpretation applications discussed in Section 3 has led us to the conjecture that all functionally accurate cooperative distributed structures have at their heart distributed search. Distributed search involves the integration of partial results emanating from multiple, semi-independent (disconnected) loci of search control. An adequate model for describing and analyzing distributed search techniques has not been developed. This lack of a developed model of distributed search exists even in centralized environments such as the Hearsay-II speech understanding system.

We are developing a model for distributed search that provides a common framework for addressing the following questions:

*What is the nature of the search space in a given application?

*How is the search space represented in the system (levels of abstraction, data structure, etc.)?

*What is the nature of a partial result in this representation?

*How are partial results extended and merged together?

*What is the nature of information that must be communicated between semi-independent loci of search control?

*How are certain types of error resolved as part of the search process?

*How are the semi-independent loci of search control coordinated?

*How is the search space searched [3]?

We are developing this model by generalizing four distributed search techniques: relaxation, the locus model, cooperating experts, and hypothesize-and-test (ala Hearsay-II). We hope this model will have a taxonomic character that will provide a framework in which new, alternative search techniques become apparent. The model should also lead to the development of a small set of control primitives and data structures which are appropriate for all types of distributed search techniques and applications.

4.2 DISTRIBUTED PLANNING

Experiences with the two distributed interpretation applications has also lead us to understand that distributed focus of attention is a crucial aspect of all functionally accurate cooperative distributed systems. Distributed focus of attention involves the dynamic allocation of processing power, memory, data, and communication capability. We feel distributed focus of attention is part of the larger issue of distributed planning (focus of attention is planning which is directed at one's own immediate internal processing).

To approach distributed planning and distributed focus of attention in the same way as we approach distributed search (i.e., the development of a generic model) seems premature. There is much less experience with distributed focus of attention and distributed planning, and sufficient intuitions have not yet been developed. Therefore, a more directed, experimental approach in this area seems appropriate.

We are engaged in two investigations to help develop these needed intuitions. The first investigation involves determining the implications of reorganizing, for a distributed environment, two existing focus of attention techniques: shortfall density scoring [Woods 1977] and the Hayes-Roth and Lesser focus of attention mechanism [Hayes-Roth and Lesser 1977]. The shortfall density

---

[3] Current characterizations of search using such terms as breadth-first and depth-first are inadequate even in centralized environments.

scoring technique is an admissible focus of attention strategy and the Hayes-Roth and Lesser mechanism is a heuristic strategy based on a number of general focus of attention principles.

The second investigation approaches the more general issue of distributed planning. The analogy between most current planning techniques and distributed planning is similar to that between heuristic search and distributed search (i.e., in most planning systems there is only one locus of control).

We are taking existing AI planning techniques and generalizing them to more than one locus of control in which the planning proceeds in a functionally accurate cooperative manner. We are also taking models of group planning developed in Organizational Theory and placing them in a computational framework.

5. CONCLUSION

We feel methodologies can be developed for functionally accurate cooperative distributed systems in which the distributed algorithms and control structures function with both inconsistent and incomplete data. These methodologies are necessary in order to extend the range of applications that can be effectively implemented in distributed environments. Our current generic research studies in the areas of distributed search and distributed planning as outlined in the paper will aid in the development of a general theory for functionally accurate cooperative distributed systems.

6. REFERENCES

Barrow, H.G. and Tenenbaum, J.M. (1976). "MSYS: A System for Reasoning about Scenes," AI Center Technical Note 121, Stanford Research Institute, Menlo Park, California.

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

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

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," Proceedings of the 4th International Joint Conference on Artificial Intelligence, Tbilisi, Russia, 483-490.

Erman, L.D. and Lesser, V.R. (1978). "The Hearsay-II System: a Tutorial," Chapter 16 in W.A. Lea (ed.) Trends in Speech Recognition, Prentice-Hall, Englewood Cliffs, New Jersey (in press).

Eswaran, K.P., Grav, J.N., Lorie, R.A. and Traiger, I.L. (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.

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

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.

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

Hayes-Roth, F. and Lesser, V.R. (1977). "Focus of Control in the HEARSAY-II System," Proceedings of the 5th International Joint Conference on Artificial Intelligence, Boston, Massachusetts, 27-35.

Jefferson, D. (1977). "Hydra-Message Facility," Private Communication.

Kimbleton, S.R. and Schneider, G.M. (1975). "Computer Communication Networks: Approaches, Objectives and Performance Considerations," Computer Survey, Vol. 7, No. 3, 129-273.

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

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

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

Nii, H.P. and Feigenbaum, E.A. (1977). "Rule-Based Understanding of Signals," Proceedings of the Workshop on Pattern-Directed Inference Systems, Hawaii, 83.

Peebles, R. (1977). "Concurrent Access Control in a Distributed Transaction Processing System," 2nd

Annual Brown University Workshop on Distributed Computation.

Rosenfeld, A., Hummel, R.A. and Zucker, S.W. (1976). "Scene Labeling by Relaxation Operations," IEEE Transactions on Systems, Man, and Cybernetics, SMC-6, 420-433.

Rubin, S. and Reddy, R. (1977). "The Locus Model of Search and its Use in Image Interpretation," Proceedings of the 5th International Joint Conference on Artificial Intelligence, Boston, Massachusetts, 590-595.

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

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

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.

Thomas, R.H. (1976). "A Solution to the Update Problem for Multiple Copy Data Bases Which Uses Distributed Control," BBN Report, No. 3340.

Woods, W.A. (1977). "Shortfall and Density Scoring Strategies for Speech Understanding Control," BBN Report No. 3303, Bolt Beranek and Newman Inc., Proceedings of the 5th International Joint Conference on Artificial Intelligence, Boston, Massachusetts, 18-26.

Distributed Intelligence for Situation Assessment

F. Hayes-Roth and R. Wesson
The Rand Corporation

ABSTRACT

The principal features of distributed intelligent (DI) systems include a diversity of perspectives arising from separated individuals and a problem-solving method whereby the individuals can coordinate their activities to accomplish a collective objective. These features require numerous capabilities and supporting structures. A system that embodies such capabilities, however, seems especially well suited to situation assessment (SA) tasks. These tasks require the collection, analysis, and interpretation of widely disparate data. Moreover, comparably complex and distributed systems are often required to disseminate the results of these analyses effectively. In this paper, we explore the relationships among structural capabilities of DI systems and functional requirements of effective SA systems. We discuss some of the options and trade-offs of various architectural decisions in the context of two contrasting DI designs.

1. INTRODUCTION

Intelligent behavior depends primarily on two capabilities: first perceiving and interpreting the environmental situation, and then performing actions that seem likely to help attain goals while preventing potential undesirable outcomes. We refer to the first type of task as situation assessment (SA). The second general class of activities we refer to as planning and control (PC). Because most human activity aims to achieve goals and avoid risks, SA and PC tasks are ever present.

Although the bulk of previous research on intelligent systems has focused primarily on how an individual person or machine might accomplish SA and PC tasks intelligently, we believe that the most significant and important problems of human culture are approached collectively. We refer to systems that coordinate the activities of separate intelligent individuals as distributed intelligence (DI) systems. DI systems behave intelligently by coordinating decisions of separated individuals who can communicate limited amounts of information to one another.

SA tasks generally require the collection of information from disparate sensor systems that may vary in their locus, specific sensitivities, or performance characteristics. Because the purpose served by SA is to enable an organism or organization to plan appropriate behaviors, the important outputs of an SA system often span a range of concerns. In support of a military campaign, for example, different commanders will require analyses that vary according to their own battle management responsibilities. Both the data for and results of such SA tasks are highly distributed geographically. This leads us to believe that DI systems provide a natural architecture for SA. This paper first examines the principal features of SA and DI to strengthen this intuition. We then define a large space of alternative DI-SA system architectures, and consider two specific, extreme examples. The contrast between these two suggests numerous research questions which are briefly noted.

2. THE CONCEPT OF DISTRIBUTED INTELLIGENCE

In our pursuit of conceptualizations of DI and their relationship to SA, we have found it useful to schematize the core concepts in representations akin to those used in knowledge-based systems (cf. Lenat, 1975, 1977; Martin et al., 1977; Hayes-Roth et al., 1978). Figure 1 displays in outline form the concept of distributed intelligence and some of its principal facets. It succinctly describes our current view of DI. In the remainder of this section we elaborate on some of the more interesting and less obvious characteristics.

2.1 Individuals

In our view, DI's most interesting property arises from the multiple viewpoints or perspectives of its diverse individuals. Within a "perspective," we intend to include any measurement system, frame of reference, or internal representation scheme that distinguishes one individual from another. Thus, owing to their potentially varying features or loci, different individuals may assess the same observable differently. Subsequent inferences by these individuals may further diverge.

These differing perspectives, while powerful, may be difficult to reconcile. In areas of overlapping responsibilities, conflicts may occur as the individuals involved each attempt to interpret the same data in different ways. Resolving these conflicts can prove to be difficut and expensive. Complicating matters further, the limited communication channels usually available greatly reduce the effective sharing of these diverse perspectives. All this combines to generate the ill-founded, "accomodating" solutions which characterize much of today's organizational behavior.

2.2 Organization

We identify the foremost features of organizations as their structure and government. Both of these directly affect how well the system can achieve its purposes. Structure provides a static characterization of the system's behavior, govern-

CONCEPT NAME	Distributed Intelligence (DI)
DEFINITION	A DI system consists of a set of individuals who cooperate in solving a problem and who necessarily have different perspectives (owing to different data, limited communications, etc.)
EXAMPLES	
COMMERCIAL	committees, businesses, trucking, economies
COMMUNICATIVE	telephones, stock market, banking systems
ANATOMICAL	embryo, body, nervous system, visual pathway
SURVEILLANCE	infectious disease control, DIA, CIA, NSA, fusion centers
PHYSICAL	spider web, football teams, armies
INDIVIDUALS	
FEATURES	sensors, knowledge, memory, processing, transmission
NICHE	environmental resources, demands, opportunities
LOCUS	space, time, culture
PERSPECTIVE	How does what individuals sense/know/transmit differ?
FUNCTIONS	notice, interpret, seek information negotiate, confirm, repudiate, and elaborate decisions store, forget, transmit
THINK-TALK RATIO	computation rate/communication rate
ORGANIZATION	
STRUCTURE	Whose resources/perspective does an individual share? communication network; shared communication contexts
GOVERNMENT	How are authority/responsibility allocated? policies and adjudication power and enforcement
CONNECTEDNESS	average number of individuals each individual talks to
COMMUNICATION-EFFICIENCY	proportion of communication requests serviceable without delaying any individual
COOPERATION-EFFICIENCY	communication-efficiency / think-talk ratio
INTELLIGENT BEHAVIOR	
PERCEPTION	recognizing instances of generic categories
ATTENTION	focusing on perishable, important issues
DEDUCTION	logical reasoning
INTERPRETATION	modeling purposes and intentions
INDUCTION	plausible reasoning, modifying knowledge
PREDICTION	generating expectations
GOAL SETTING	formulating subproblems and objectives
SEEKING	searching for needed information
ANSWERING	replying to others' questions
LANGUAGE	cooperative communication conventions abstractions and symbols
KNOWLEDGE	world models, multiple abstractions
MEMORY	storing, retrieving information
EMERGENT PROPERTIES	
DEFINITION	unpredicted macroscopic system behavior
EXAMPLES	evolution through natural selection centralization of authority in crises
INFLUENCES	system complexity environmental forces organizational adaptability
CONSEQUENCES	potentially very significant

Figure 1. A conceptual description of DI.

ment its dynamic aspects.

Our view of organizational structure differs from conventional ones in its emphasis on information processing as the organizing principle. This special focus ignores other sources of organizing influence, such as institutional forces or the requirements of production, distribution, sales, etc. Within this pure information processing framework, many interesting issues arise.

One structure gives immediate or proximate access to all parties interested in comparable data (by associating multiple individuals with a common memory, for example). More generally, a system can facilitate communication by arranging for different individuals to share perspectives. This in turn creates opportunities for related individuals to encode their messages efficiently in ways that each other can decode, e.g., through the use of jargon.

We define the government of a DI system as the set of processes which controls the interaction among individuals. To define a goverment one decides which individuals are responsible for what actions and which have authority to modify or control the behavior of others. Implementing a goverment is then accomplished by giving power to those with authority and compelling them to enforce the governmental policies.

Unfortunately, numerous simple policy objectives (such as "fair" elections) are not achievable by DI systems even in theory (cf. Arrow, 1951). Empirically, moreover, it seems that very few DI systems even correctly implement their explicit policies. Even if there were no theoretical obstacles, the problems of controlling competing and cooperating individuals in their pursuit of multiple, sometimes incompatible objectives exceeds the capabilities of most DI systems. As a manifestation of this limitation, many adaptive systems exhibit great flexibility in the execution of governmental functions. Vague policies, historical precedents and opportunistic decisionmaking often achieve governmental robustness and flexibility where explicit policy implementations could not.

2.3 Intelligent Behavior

Presumably DI systems exhibit intelligent behavior. What exactly do we mean by intelligence? Without attempting to formalize a definition, we can usefully identify a number of capabilities that intelligent systems possess. Most of these occur in all DI systems, and we anticipate that they would be equally useful in new applications of DI to SA tasks. The aspects of intelligent behavior listed in Figure 1 provide a good overview of these capabilities. We will consider a few of these in more detail in this section.

The first class of capabilities, including perception, attention, deduction and interpretation, might best be called model-based reasoning. This type of reasoning compares sensory data with internal models or expectations to test hypotheses about the source and meaning of the observables. To achieve efficiency, most intelligent systems focus their attention on important events or data by allocating scarce processing resources in accordance with the expected significance of the processing outcome. One basic process is used both to perceive objects and interpret events. Matching processes identify the most likely cause of observed data from among a potentially large set of stored pattern templates or generative models. This stored set of patterns comprises the bulk of the system's declarative knowledge. The matching processes themselves exploit primarily simple deductive schemes, though inductive reasoning is often necessary to assess the credibility of alternative hypotheses.

The second general category of capabilities might best be considered planning. This category would include induction, prediction, goal setting, and information seeking skills. Basically, these skills support goal-directed reasoning. Inductive methods enable the system to infer likely outcomes of current or planned events. These predictions can be used to evaluate prospective alternatives or, in a reverse way, to search for feasible methods of achieving desired goals. Often, of course, no immediate plan can be found to achieve some goal, because additional knowledge or information is required. In this case, the intelligent system sets appropriate information-seeking goals and plans methods to satisfy these subordinate goals.

The third class of intelligent capacities we identify as knowledge management. These concern the storage, retrieval, and communication of knowledge. The diversity of perspectives in DI systems create special problems for knowledge management. Frequently, a system "knows" something but is unable to use that knowledge because the relevant individual can't find it. In response to such problems, many DI systems develop schemes for partitioning knowledge in arbitrary ways. An interested individual might ask someone who, in turn, knows who to ask to find the knowledge. These retrieval schemes often correspond to hierarchical (associative) indexes. Alternatively, many systems develop knowledge management specialists to store and index knowledge centrally, complete with their own languages and communication protocols.

2-4. Emergent Properties

DI systems provide an unparalleled opportunity for the study of emergent properties. This term refers to macroscopic behavioral characteristics of a system that appears as a surprising consequence of individual features. The complexity of DI systems fosters the development of such emergent properties. Just as most governmental planners learn that nearly all policies have unintended but significant consequences, so will most DI systems manifest interesting and potentially important macroscopic properties.

Emergent properties can affect DI systems in many ways. They can directly affect the system's capability to perform well in its environmental niche. Some emergent properties (e.g., competition for limited resources) lead to adaptive consequences, while others (e.g., information gluts) lead to undesirable ones. They can also affect the organization indirectly. DI systems often react to their own emergent properties by actively trying to assess, exploit, and control them. Such practices might be considered open-loop feedback control. However, this type of control is quite difficult to achieve. By their very nature, emergent properties are surprising and may not be easily detected or correctly interpreted. Attempts to perturb them may well have their own unexpected results, with total confusion the outcome. Nevertheless, it seems clear that purposive DI systems need to address this type of assessment and control problem. Otherwise, there may be little assurance that the system will continue to achieve its objectives as the environmental conditions vary.

3. THE CONCEPT OF SITUATION ASSESSMENT

The concept of situation assessment and its principal facets are schematized in outline form in Figure 2. In the following subsections, we briefly consider some of the important features of the SA task.

3.1 Hypotheses and Tasks

In most complex environments, the primary output of SA consists of a set of hypotheses concerning the identity, location, capabilities, and intentions of the monitored elements. These will usually be multiple hypotheses corresponding to any set of sensor reports, since observations are generally incomplete and/or errorful. The general strategy for resolving uncertain hypotheses is to find additional information to confirm the valid hypotheses and disconfirm the erroneous alternatives. Frequently, this information is not immediately accessible. Somehow the system should acquire the needed data in a timely fashion and then route it automatically to the relevant analyst who knows why it was gathered and how to utilize it efficiently for its intended purpose.

3.2 Cooperating Sources of Knowledge

SA typically requires applying diverse sources of knowledge to the interpretation problem. This diversity can reflect separation of sensor observations in time or space or, more generally, different types of interpretative expertise. Among these types of expertise we frequently require knowledge about how sensors respond differentially to potential events, how the environment affects both the sensors and the monitored elements, and how the monitored elements are expected to behave. The knowledge, in each case, can be considered a model of behavior. Thus, SA incorporates sensor models, environmental models, and force models.

For military applications, force models are particularly important. The foremost objective in this type of application is to understand what the opposing forces intend to do and are capable of doing. To achieve this type of understanding, we need to construct an interpretation of their force disposition and activities. Our interpretations necessarily combine our diverse sources of knowledge with current sensor reports to attribute intentionality, capability, and specific locations to the various elements of their forces. To the extent that our interpretive methods and supporting data are conclusive, our interpretations are valid. Conversely, if our methods and data are inconclusive, our hypothetical interpretations are necessarily uncertain.

3.3 Inference Processes

We have already suggested that the basic method of SA is hypothesize and test. This label identifies a particular paradigm of problem-solving. To apply it to SA tasks requires that the diverse types of knowledge can contribute directly to the problem of mapping between observable data and various intermediate levels of interpretations. Thus, we use data from an earlier SA as well as current sensor reports to suggest plausible hypotheses about how the current situation should be viewed. These new hypotheses, in turn, require confirmation. Supporting evidence is used to strengthen the credibility of hypotheses, and contradictory data is used to eliminate alternative hypotheses from consideration.

Two principal problems occur in SA: too much raw data and too little high-level interpretation. These problems are connected intimately. If we had fewer sensor reports to evaluate, we could at least subject them to extensive analyses. On the other hand, the usual reason we collect so much information is to provide additional means of eliminating uncertainty from our resulting hypotheses. Thus, the core problem is to determine which hypotheses are most critical and which data directly contribute to their development and evaluation.

Several factors are involved in deciding how to allocate limited analysis resources in complex SA tasks. First, many data need not be attended to at all; these are data that are completely consistent with reliable predictions about what the sensors will see and report. Second, in most cases, our own intentions significantly affect what data we need to collect and interpret. Just as a runner generally needs only to look directly ahead for obstacles and dangers, so do most intelligent organizations have specific missions to accomplish that can focus their SA effectively. Finally, many SA problems can be decomposed into separable tasks. In many cases, the separable tasks concern different spatial or functional areas of assessment, and these naturally suggest parallel computations by separate components. Alternatively, even when parallelism is not appropriate, SA subtasks often exhibit especially efficient natural orderings. For example, over-

CONCEPT NAME	Situation Assessment (SA)
DEFINITION	SA tasks involve interpreting and predicting the behavior of a set of elements that can affect selected aspects of the environment.
EXAMPLES	
BATTLEFIELD	status of forces, rules of engagement, tactics, objectives
DISEASE CONTROL	type, locus, virulence, resistance, populations of pathogens
INVENTORY	current and projected stock needs
ECONOMIC FORECASTING	prices, wages, inflation, money supply, inventory
AIR DEFENSE	status of own and other air forces,
OUTPUTS	
HYPOTHESES	what, when, where, why, how, how much?
TASKS	what additional information is needed, where can it be obtained, and who should get it?
SENSORS	
CAPABILITIES	where, what, when, accuracy, noise
MODELS	generative models of their behavior
REQUIREMENTS	power, communications, computation, terrain
KNOWLEDGE SOURCES	
SENSOR REPORTS	
ENVIRONMENT	geography, terrain, weather
FORCE MODELS	Force elements: tanks, RPVs, etc. Force structure Force applications: what can they do, how? Force status: where are they? Force intentions: what purposes are they seeking? how would they achieve that? Vulnerabilities: how are we vulnerable? what activities of theirs should deserve our attention?
PRIOR SA	expect the future to be much like the past
INFERENCE PROCESSES	
METHOD	hypothesize and test
KNOWLEDGE REPRESENTATION	multilevel models mapping intentions into observables
EXPECTATIONS	use prior SA and knowledge sources to predct events
CONFLATION	combine sensor reports with knowledge-based predictions to validate critical hypotheses
DATA GLUT	too much low-level data--which to store, which to discard perishability and criticality
REDUCTION	remove expectable data from consideration
CRITICALITY	focus attention on surprising, important data
PARALLELISM	geographically disparate events are nearly independent and can be pursued in parallel
SERIALITY	gross assessments should precede detailed ones
SECURITY	
VULNERABILITY	what can be revealed, what can't?
COMMUNICATIONS	who threatens your communications and how?
RISK CONTROL	
SURPRISE	recognizing when expectations are violated generating new models of behavior
RESOURCE ALLOCATION	some resources to interpretation; some for surprises

Figure 2. A conceptual description of SA.

all gross assessments can sometimes obviate numerous alternative fine-grained considerations.

3.4 Security and Risk Control

An organization established to accomplish SA often faces two different types of threats. Their ability to monitor the forces of interest may be curtailed or subverted. Furthermore, they may be made foolish by surprise. Both of these threats significantly affect the design of SA systems.

The threat to security comes primarily from the fact that an intelligent opposition can thwart one's ability to collect or communicate important information. Threats like this cause SA systems to perform their functions under the protection of both physical and communications security (through encryption, etc.). Once the opposition knows what can be observed and how it is processed, it can often hide its force elements, modify or disguise its emissions, or capture or occlude sensors.

Surprise often plays a significant role in military campaigns. Because SA is inherently an interpretive process, it exploits prior knowledge to generate expectations and comprehend the observed data. In this way, knowledge simultaneously provides the basis for understanding and the basis for surprise. An opposing force that behaves in an unforeseen or incomprehensible way can befuddle the SA organization. Our view of surprise is complementary to that of interpretation--that surprising events can be monitored by their contrast with expected phenomena. This mechanism at least provides a clue to the data that deserve (or demand) further attention. If surprising data are left unattended, the SA system cannot control the associated risks. To accomplish its interpretation task as well as its risk control mission, it appears that the SA organization must divide its resources between interpretable and uninterpretable phenomena.

4. DI ARCHITECTURES FOR SA

Our current research project is aimed at the goal of developing and refining architectural principles for DI-SA systems. In this work, the concept of SA just discussed is being elaborated to compare and contrast it with that of DI. We have defined a testbed SA problem suitable for experimentally investigating various DI architectures. Below, we describe this problem and two extreme structures.

4.1 The Benchmark Problem

In a full-scale battlefield SA problem, we might be concerned with monitoring locations and movements of air forces, infantry, artillery, and armored divisions, as well as predicting their likely objectives, routes, and performance capabilities. Because this problem is exceedingly complex, we prefer to address a drastically simplified class of problems initially.

The principal features of the rich SA problem should be preserved in the benchmark problem, but their complexity should be parameterized and attenuated. Among these features we include: (1) multiple elements to be interpreted at multiple levels of abstraction; (2) the elements should be distributed spatially and should move through space; (3) multiple sources of knowledge should be able to provide constraint; and (4) individual sensors should be inadequate in scope or informativeness to solve significant portions of the SA problem, and (5) the value of their data should depend on timeliness. Furthermore, the benchmark problem should not require classified information or military expertise. We intend to employ hybrid, man-machine simulations of DI-SA systems, so the problem should be one that can be readily understood and solved by ordinary people with little formal training.

Currently, we are considering the following SA benchmark task, and it will be used to contrast the two alternative architectures. The task is called the message puzzle. It is outlined in Figure 3, and illustrated in Figure 4.

Several features of this task are worth noting. First, it readily manifests the four principal characteristics we desired in our benchmark: (1) the elements need to be interpreted at the letter, word, and message levels of abstraction; (2) the letters are spatially distributed and move through space in formations; (3) knowledge about the board, the clues (if any), and the language (lexicon, syntax, semantics) should provide multiple sources of constraint; and (4) no individual sensor will provide a significant amount of the needed information at any one time, and (5) the data themselves will be of perishable value. In addition to its satisfying these foremost objectives, the proposed benchmark problem also exhibits several other desirable properties. (6) The granularity of the sensors and their level of abstractions can be readily changed. If we wished, we could use more sensors and have them report only graphemic parts of letters thus necessitating cooperative analysis and interpretation of each letter. (7) The complexity of the overall SA task can be modulated at will. The parameters that we may vary include: the scope and accuracy of the sensors; the size and difficulty of the puzzle; the observability, identifiability, speed and coherence of the (word) formations; and the expectability or surprise value of the actual messages. (8) The SA organization can be evaluated for its ability to recognize different pieces at different locations or at different levels of abstraction. (9) Each additional correct interpretation simplifies further decision making; each additional incorrect hypothesis complicates subsequent interpretation decisions.

With this benchmark problem in mind, we now consider two hypothetical DI-SA architectures.

4.2 The Dynamic Hierarchical Cone

The first architecture is a variant of so-

CONCEPT NAME	Message Puzzle
DEFINITION	A situation assessment task where the elements to be interpreted include letters and words moving toward meaningful (spatial and semantic) objectives. The world consists of a crossword puzzle board. The words to fill the open slots move independently across the board until they reach their target slots, at which point they stop. The SA task is to guess as many of the fillers as possible before they arrive at their slots.
OUTPUTS	
HYPOTHESES	what messages are the letters forming?
TASKS	what additional information is needed, where can it be obtained, and who should get it?
SENSORS	
CAPABILITIES	can see a small block of squares, can report the letters in them, with various degrees of latency and certainty
MODELS	rules regarding how the sensors perform
REQUIREMENTS	the sensors may be limited by load and comm. constraints
KNOWLEDGE SOURCES	
SENSOR REPORTS	
ENVIRONMENT	the puzzle board and some knowledge of target characteristics (e.g., categories)
FORCE MODELS	Force elements: letters, words, phrases Force structure: syntax, semantics, crossings Force applications: the letter sequences move snake-like through the spaces toward destinations Force status: where are they?. Force intentions: what targets and meanings are they seeking? Vulnerabilities: which targets have highest payoffs?

Figure 3. A partial description of the benchmark SA task.

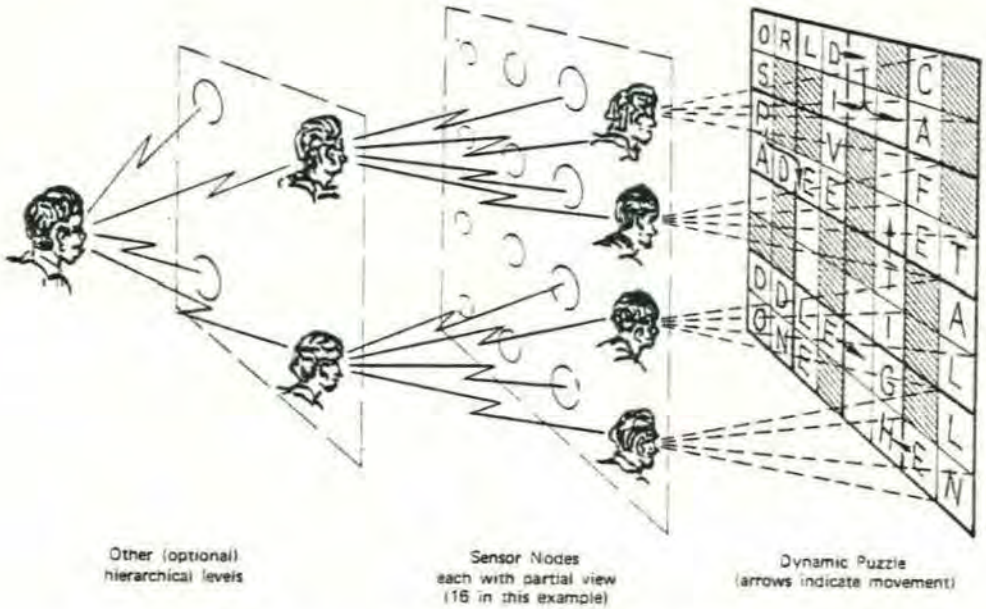


Figure 4. Message Puzzle Task

called "perceptual cones" (cf. Uhr, 1978; Tanimoto and Pavlidis, 1975). Where ordinary perceptual cones define a static hierarchical structure among processing individuals at different levels of abstraction, we propose to link lower individuals to higher ones in dynamic ways. Specifically, a lower level individual that has important information to report requests a superior to receive it. The higher-level superiors acknowledge those requests that appear most significant and timely. Thus the superior-subordinate mapping from one level to the next may vary over time.

For simplicity, assume an 8 x 8 puzzle board and a three-level cone with one, four, and sixteen individuals at the various levels, as shown in Figure 4. The lowest-level individuals can "see" a block of four squares in that they receive (possibly errorful) reports describing their contents. All communication is vertical, either from a subordinate to a superior or vice versa. Each individual can send messages to as many as four individuals and can receive up to four messages per time step. If more than four messages are sent to an individual in any time step, randomly selected ones are lost until just four arrive. Each individual has unlimited computing and storage capacities, for all practical purposes. The capacity of one message is limited to four hypothesized cell contents or a string message hypothesis plus corresponding instructions (e.g., "expect," "reject," "verify"). Some simple credibility measure of the hypotheses is also permitted. Additional simple messages may be communicated to control the behavior of subordinates, such as "quiesce for 2 time steps" or "sleep until awakened."

At the outset, the board is partially filled with letter strings, and some of these form partial messages. The low-level (level 3) individuals receive reports on their 2 x 2 windows. Some of these are more informative than others. For instance, the letter sequence "kw" which might suggest "awkward" but only a few alternative words. In response to the "kw" data, this individual might decide to send one message to all superiors that predicted "awkward" in the appropriate positions.

The immediate problems that arise from this simple example include the following. The individuals don't know exactly which superiors are taking responsibility for processing their hypotheses. Although this means that the potentially wasted capacity of static cones has been avoided, it also means that time and effort will be spent deciding which subordinates and superiors to couple. If individuals are allowed to play multi-level roles (thereby reducing the total number of individuals while retaining a hierarchical structure), the problem of who to talk to is further compounded.

Vertical communications of this sort have benefits as well as costs. If the network organizes itself into a static quaternary tree, a prediction can be communicated from one side of the board to the other in two time steps by linking up and then down. The higher-level individuals can develop plans for governing the lower-level ones to maximize the net result. Once an accurate prediction has been made and confirmed, it may be desirable to put sensors to sleep to minimize communication contention. Collective wisdom can thus be accumulated at higher-levels based on multiple inputs without duplicating efforts at many equivalent sites in the lower-level.

4.3 The Anarchic Committee Architecture.

In the previous architecture, knowledge accumulated at higher-levels of abstraction, and it was natural to suppose that power and authority would be similarly distributed. In contrast we now consider a design in which no individuals are distinguished by knowledge, authority, or power. Rather, the only distinctions that arise reflect increased self-assumed responsibility. Specifically, a single-level network is proposed wherein individuals who believe they have important information join with some of their peers in forming a consensus. Each decision then results from the formation of a corresponding committee that assumes responsibility for its judgment. Later, if any particular decision is challenged, the committee responsible should be prepared to defend its conclusion.

As previously, each individual has responsibility for monitoring a 2 x 2 block of squares. To compensate for the reduction in total number of individuals, we allow a higher communication rate. A typical scenario might go as follows. An individual reads his sensor report and decides to do one of four things. He will remain silent if he has nothing significant to say or if, in his opinion, he should delay communicating temporarily. He will formulate an hypothesis and volunteer to form and chair a committee if he is the first to recognize the possibility of needing a new committee. If he belongs to a committee with pending matters, he may contribute useful information to its chairing individual. Finally, if he is chairing a committee but believes the matter before it is moving out of its jurisdiction (because of limited communication range or departure of the message from its purview), he may attempt to enlist another relevant individual to form and chair a committee on this message. In this way, one committee hands off its problems to others.

This architecture manifests very different properties than those exhibited in the hierarchical system. Here, knowledge accumulates by forming increasingly large committees. At any point in time, the committee membership reflects those individuals that still believe they have an interest in pursuing a message. As the messages move, the committees become staffed with old, irrelevant members and a need arises to hand off committee membership. Higher-level analyses emerge as the committees either become larger or as new ones devoted to more abstract data processing are formed.

The advantages of this architecture seem to derive primarily from its reduced number of individuals and its simplified communication network. On the other hand, it entails significantly more communication than the hierarchical net. Where the hierarchy could use high-level assessments derived from many sensors to comprehend regional patterns and track them without hand-off, the anarchic committee design must constantly hand-off tracking problems as patterns move from one area to the next. This introduces problems of choosing individuals to hand the tracking problems to and re-establishing committee membership. Moreover, the need to form higher-level assessments with the same individuals who are performing other tasks seems to complicate the problem of indexing knowledge. When the same individual plays multiple roles, he becomes a potential bottleneck for the system.

4.4 A Large Space of Design Alternatives

What our preliminary analyses have indicated is a very large space of alternative designs for DI-SA architectures. To explore this space is the purpose of our current research. The two simple straw-man architectures just discussed vary primarily on one dimension, that of communication network structure. This variation can be seen to influence many of the opportunities for adaptive behavior. In particular, established hierarchies simplify tracking of motion and indexing of others' knowledge. In contrast, they create requirements for effective governmental policies. Whenever such policies are required, we anticipate their creating numerous unanticipated emergent properties.

Our current conceptual descriptions of DI and SA provide a rich set of dimensions for varying possible architectures. We expect that the communication structure will provide the best single experimental variable for study. This belief is based on the fact that many, if not most, characteristics of a DI system are responsive to the limited communications paths between individuals. We expect to find that many significant DI properties are adaptations to, rather than the cause of, their communication capabilities. In addition, we believe we can manipulate this dimension effectively, in both human and machine simulations of DI-SA.

5. CONCLUSIONS: TOWARD DESIGN HEURISTICS FOR DI AND SA

Our basic approach to DI and SA has been exploratory. We have attempted to formulate the principal features of each concept and to relate these two together. At this point, we believe experimental and analytical investigations of the effects of alternative communication network structures on SA performance would be appropriate. In formulating our experimental benchmark problem, we have emphasized the significant interpretation problems of SA and deemphasized the lower-level signal processing problems. This reflects our belief that DI is an important candidate for performing the multilevel interpretation problems that arise in battle management. These problems require bringing diverse sources of knowledge to bear in understanding activities at different levels of abstraction. In turn, we see the outputs of SA being provided to different personnel with correspondingly varied interests and needs.

In our overall approach, we have attempted to develop a mapping between DI and SA concepts to suggest architectural principles. Our highest level research heuristic suggests viewing DI and SA in different ways to develop alternative, interesting analytical or functional analogies. For example, viewing DI as a structure and SA as a problem, the following heuristics apply:

H1: Usually, structures evolve to support the needed functions; thus, consider what functions accomplish SA problem-solving and then develop supporting structures.

H2: Good solutions typically employ modularized functions.

H3: Good structures support modularized functions by a set of independent design features.

H4: Once a list of independent design features is accumulated, these may be realized by separate physical entities or by multifunction designs.

We could formulate other views of DI and SA that would stimulate other design heuristics. Some alternative views of DI systems would consider them principally as processes, individuals or populations. Alternative views of the SA task would emphasize its similarity to pattern recognition, to Gestalt perception, or to an abstract map (with insets).

From our preliminary human experiments and analyses, we have developed some tentative design concepts and heuristics, which we list without further elaboration:

H5: Individuals' diverse perspectives both provide the constraint needed for effective problem-solving and engender the complexity of cooperative problem-solving.

H6: Global (consensual) perspective can best be derived by abstracting local perspectives, and the resultant abstractions can provide an effective intermediary for indexing others' knowledge.

H7: Organizational elements should adapt differentially to variations in environmental complexity by restructuring subsets and selectively allocating resources to them.

H8: Emergent properties significantly affect the behavior of a DI system, and can most effectively be studied empirically.

H9: Close couplings of individuals facilitate their communication and provide the principal basis for dynamic organization.

H10: Assuming adequate memory, comparable individuals should possess identical capabilities. Individuals should be differentiated when different environmental or organizational niches imply specialization.

H11: Uncertainty can best be resolved by attending to the most (conditionally) diagnostic information at each point in time. Gross filters should therefore be employed before fine ones; these filters suggest which details are worth computing and which are not.

H12: Hierarchical networks minimize communication in interpretation tasks. However, their optimal structure can only be known dynamically, after the data are presented.

H13: Situation assessment can be significantly simplified by appropriate use of expectations. Such expectations are most easily implemented by passing responsibility for them directly to individuals who could locally confirm or disconfirm them.

H14: SA can be simplified by abstraction.

H15: Spatially hierarchical nets are faster than linear neighbor-to-neighbor communication paths.

H16: The same individual can play roles at different levels of abstraction, but this increases the difficulty of knowing which individual to communicate with.

H16: Centralization is undesirable when computing is slow relative to communication or communication is difficult. Otherwise, centralized computing is desirable. In particular, centralized committee decisionmaking is good.

H17: Consumer-producer models suggest locating processing near supply and demand, except when it is much cheaper to locate it elsewhere (e.g., because significant economies of scale exist).

H18: There are few cases where communication is preferable to computing--in research, where it is desirable to avoid rediscovering what is already known; or in any situation where expectations obviate computing.

ACKNOWLEDGMENTS

This work was supported in part by the Information Processing Techniques Office of ARPA under contract number MDA-903-78-C-0029. We have collaborated with several other Rand researchers and gratefully acknowledge their contributions to our project and ideas. These colleagues include John Burge, Carl Sunshine and Norm Shapiro.. We have also benefitted from discussions of military SA problems with Rand personnel who are too numerous to list.

REFERENCES

Arrow, K. Social Choice and Individual Values. Coles Commission Monograph # 12, New York: Wiley, 1951.

Cotrell, A. Emergent properties of complex systems. In Duncan, R. and Weston-Smith, M. (Eds.), The Encyclopedia of Ignorance. New York: Pocket Books, 1977.

Hayes-Roth,F. Klahr, P., Burge, J., & Mostow, D. J. Machine methods for acquiring, learning and applying knowledge. P-6241. The Rand Corporation, Santa Monica, California, 1978.

Lenat, D. BEINGs: Knowledge as interacting experts. Proc. 4th Int. Joint Conf. Artificial Intelligence, Tbilisi, USSR, 1975, 126-133.

Lenat, D. Automated theory formation in mathematics. Proc. 5th Int. Joint Conf. Artificial Intelligence, Cambridge, Mass., 1977, 833-842.

Martin, N., Friedland, P., King, J., and Stefik, M. Knowledge base management for experiment planning in molecular genetics. Proc. 5th Int. Joint Conf. Artificial Intelligence, Cambridge, Mass., 1977, 882-887.

Tanimoto, S. and Pavlidis, T. A hierarchical data structure for picture processing. Comput. Graphics Image Processing 4, 1975, 104-119.

Uhr, L. Pattern Recognition, Learning, and Thought. Englewood Cliffs, N. J.: Prentice-Hall, 1973.

DSN PROBLEMS--AN OVERVIEW

Jeffrey A. Barnett
USC/Information Sciences Institute

A sensor is a device that observes physical phenomena. The output of a sensor is a function of these observations. Here we are concerned with the particular problem of constructing a world picture from sensor data. The world of interest contains craft capable of motion (e.g., airplanes, water vessels, and land forces) and the terrain. The important terrain features are target locations and geographic data that may affect the operation and behavior of sensors and craft. The world picture comprises descriptions of craft behavior and location with respect to the terrain and to each other.

There are many families of sensor devices. Among them are phased-array acoustics, infrared, sonar, and many kinds of radars. The major distinguishing characteristic among different sensor families is the type of phenomena that they can observe.

Each device has three primary types of components: (1) analogue components--the observation collector such as a simple microphone or antenna. Active sensors also include an analogue illuminator component; for example, a radar generates the signals that it detects after reflection. (2) digital components--a processor that turns raw observations into the sensor's output such as azimuth, heading, velocity, position, etc. The digital component may also control and coordinate the analogue components. (3) interface components--accomplish delivery of the sensor's output to the user. This function often includes graphic display and communication with a central facility.

The remainder of this paper describes the concept of Distributed Sensor Networks--DSN.

WHAT ARE THE PROBLEMS?

Current sensor technology has reached a very sophisticated state. Radars are sensitive enough to detect the differential Doppler shift between the top and bottom of a tire on a moving vehicle, and phased acoustic arrays can listen to entire oceans. However, many problems remain.

The Coverage Problem. As mentioned above, radars can be very accurate. On the other hand, a radar's normal-aircraft detection radius is limited to about one hundred kilometers; for low-flying aircraft the coverage is much more limited. Acoustic arrays, however, may have much greater range, particularly the underseas variety, but they are no rival to a radar's accuracy. No single sensor produces optimal coverage. Between sensor families there are tradeoffs among area, accuracy, and sensitivity.

The Vulnerability Problem. As one might guess, powerful sensor devices are expensive. Therefore, they are few in number, and, in general, only one or a few cover any given area. This creates the vulnerability problem. If a single device fails (or is knocked out by enemy action) a critical region may be left uncovered. Even when multiple devices cover any given area, the system is still vulnerable if a centralized facility is responsible for data coordination because the centralized facility itself may fail.

The Use of Information Problem. This is also called the inference problem. In many instances, extra-sensor information is available. For example, an intelligence report states that a particular submarine is very likely to cruise in a particular area. How is this input to be used by a sensor to more accurately determine the world picture? For another example, consider the problem of correlating data from different sensors that cover a common area--say microphones and a radar. How is the data combined, particularly if multiple craft are in the area?

The Interface Problem. The output of sensor devices can be used in a multitude of ways. Current practice delivers the information to a human user. The user makes decisions based upon this output and is able to exercise limited control over the functioning of the sensor. The trend is toward more sophisticated systems in which many observations and control functions are automated. At the point where the human user interacts with the automated system, the data is highly correlated--inferences have been drawn and acted upon. The user has a need to ascertain the method employed by the system to reach its present state. That is, the system must be able to explain and defend its behavior. Further, the system must be able to initiate dialogue rather than remain passive in the interaction, especially in the face of unusual uncertainty or during a crisis. This becomes particularly important as sensor systems undertake craft-guidance functions.

The Cost Problem. Although the cost of processing (particularly for small to medium-sized computers) has been drastically reduced by LSI technology, there has been no corresponding reduction in the cost of analogue components and packaging for sensor devices. At issue is the development of system architectures that take advantage of the current and projected cost trends.

The next section discusses distributed sensor networks, a possible approach to overcoming many of these problems.

WHAT IS A DSN?

Many of the problems discussed in the preceding section can be addressed by using sensor networks. A sensor network is a group of sensor devices connected by a communications network trying to achieve a common goal--deriving an accurate world picture. Several things need to be understood: First, the area covered by the sensors on the network is larger than any one of them can cover (at least cover well). Second, there are some areas that are covered by more than one sensor. If this is not the case, the sensors can operate in a virtually autonomous manner, and the system will neither enjoy all the benefits nor suffer all the problems discussed below. Third, the number of sensor devices in the system is not trivially small.

Problem Scope. We wish to enlarge the scope of the problem. Namely, the following types of non-sensor components are to be considered examples of potential members of the network: (1) Command, Control, and Communications (C3) systems, (2) observers with portable terminals, and (3) cruise missiles--a combination of a craft and a sensor device.

The rationale for considering the above types of components is that the military is moving towards large integrated information-processing systems for the 1980s. Sensor networks per se represent only a part of the total system. Each of the additional component types expands the demand for cooperation and system feedback. Further, they all represent a variety of usages (contexts) for the sensor system.

In order to evaluate and compare different sensor-system architectures, it is necessary to view them in the environment of their applications. For example, a system that produces only high quality output for critical regions is probably more valuable than a system that generates uniform but mediocre results everywhere. To form such an objective (versus subjective) merit function, it is necessary to include the extended components whose operations may represent the payoff of the entire system.

Distribution and Decentralization. These two similar-sounding but quite distinct concepts are often confused. Distributed processing or architecture denotes a paradigm in which, although the computation is dispersed, a global (perhaps hierarchical) view and centralized control are maintained. A typical example is cooperative tasks on a multiprocessor with a shared memory. A major problem here is conflict resolution for the shared resources, a problem normally solved explicitly by synchronization primitive use by the tasks themselves. Another class of distributed processing occurs with networks. For example, a process on one machine (a host) maintains a data base. Processes on other hosts use the data base process like a subroutine, passing information back and forth via the network. Since the data base itself resides only at a single location, problems of consistency are minimal. The major problems with this mode of operation are the delays caused by the network and the vulnerability of the entire system if the single host housing the data base fails.

Decentralization is a paradigm in which no central global view is maintained and system control functions are not centralized. This type of organization can occur on a single processor when a computation is performed by separate but equal processes. Even though in this case there exists an operating system that actually allocates resources and settles implicit conflicts among the processes, the details of such scheduling and conflict resolution are not relevant to the correct operation of the processes if they are well-formed for decentralized computation. This scheme is often used to model AI problem solutions and team efforts. The members of the team are represented as independent experts with their individual viewpoints and problem-solving methods. They cooperate when it is appropriate, and work on their own when that appears best.

Decentralization is also compatible with distribution of processing. For the data base example above, a natural extension is to distribute the data. For example, different offices of a large corporation maintain local data bases. Computations that involve data from more than one office use a communications facility, the network, to transfer the necessary information. In this situation, there is a problem of consistency, particularly if external data or inferences drawn from it are stored for any length of time--updates are not simultaneously (if at all) reflected everywhere the data resides. On the other hand, the decentralized facility provides marked efficiency gains when most computations involve only local data; the system is also more robust. Failure of no subset of the sites cripples all the capabilities of the entire system. Further, there appears to be a cost advantage to using many medium scale processors versus a single large processor. Whether this cost savings is real or only apparent depends upon the future of the hardware component revolution and the costs of communications mechanisms.

POSSIBLE ADVANTAGES

For a variety of reasons, it is obvious that sensor networks, which are by definition distributed, should also be decentralized. Decentralization is a viewpoint on the management of system complexity. As such, it dictates, to a degree, a style of organization and interaction among system components. Many (but not necessarily all) of the components must be capable of autonomous operation, working alone when circumstances dictate and cooperating at other times. In addition, the reliability and responsiveness of other components will be doubted; that is, a component should be able to continue functioning in the face of system degradation. From these considerations flow the advantages of a system that is both distributed and decentralized.

Shared Utilization. It should be possible, without reorganizing a DSN, to add new consumers of its output. In principle, this expansion is no different than expanding the DSN itself, because in both cases there is a need to share results and determine common goals.

Extended Coverage. In a similar vein, a DSN should be able to employ sensors from several families. Components performing higher-level functions such as tracking and threat evaluation are presumably using information from a variety of sources--sensors, intelligence, etc. Geographically distributed information is also available from the network and can be used to enhance the scope of information produced as well as its accuracy and its timeliness. The simplest form of this is a component in one region notifying the component(s) in another region of an approaching craft.

More Reliable Systems. Because of decentralization, the opportunity to construct very reliable systems exists. Redundant and overlapping operation increases protection against failure; centralization has the opposite effect. Because of the flexibility inherent in a DSN, the system can dynamically reorganize itself and reallocate responsibilities in the face of degradation or specific threat identification.

Better Usage of Information and Interface. In order to properly function at all, a DSN must use protocols that allow requests for information at many different degrees of completeness and accuracy. Since the components are loosely coupled, it would be odd to impose rigid communications requirements. It is exactly this kind of protocol that is needed also at the system interfaces in order to explain and display system behavior. Such a system must be capable of drawing inferences from the data it does receive and have a rational mechanism for determining what to communicate and what data to request. This is not to say that decentralization solves these problems, but rather that if a DSN is to be constructed, the problems must be solved. The solutions are then available for a variety of usages.

Cost Benefits. A DSN is by nature processing and communication intensive. The cost of these items, particularly processing, is dropping at a remarkable rate. It is presumed that the very expensive components, e.g., large radar antennas, can be displaced with cheaper, less capable versions and sensors from less costly families. The accuracy at individual sites will be redeemed through the cooperation of several times as many cheaper components augmented by intelligence and sophisticated processing. Testing this hypothesis is one of the major goals of the DSN investigation.

New Problems. Several type of craft observation problems exist today that appear outside the ability of current sensor systems. A section below, "DSN Scenarios," discusses several such problems that can be better handled by DSN.

In order to accrue these benefits, it is necessary for the workers on DSN projects to carefully define the problems and work toward their solutions. The next section describes the ISI approach.

THE ISI APPROACH

Our approach to analyzing the DSN problem is to first develop a DSN framework. The framework is a meta-model that defines the problem space. As such, it describes the kinds of components and phenomena that are to be considered. Included besides craft and sensor equipment are the terrain, environment, and extended components mentioned above. The framework is then augmented by the addition of specific information to become a model for a specific DSN system. Naturally, many such models can be formed. The framework and models are represented in a notation being developed for this purpose.

The notation, like other modelling languages, allows procedural descriptions of components and their interactions. However, to allow the framework to be written, more is necessary. An assertion language is also part of the notation. An assertion is a statement or relation, in mathematical symbolism (an extended first order predicate calculus), that allows specification without the details of a procedure. For example, an assertion in the framework can state that associated with a sensor is a radius of sensitivity and that the sensor is not affected by any physical events outside that range. In a model, particular sensors have specifically stated radii, and a consistency condition is that of the assertion to the procedure representing the sensor. An expected benefit of the assertion language is an enhanced ability to analyze properties of DSN.

As is well known, analysis is not likely to determine all (or even many) properties of very complex systems such as DSN. Therefore, the modelling notation has been designed to be executable by a simulation system. This system is called the DSN (software) testbed. Simulation results will be used to augment and test analysis results--and conversely. Through this approach, it is hoped that insights into possible DSN arcitectures and system tradeoffs can be developed.

DSN SCENARIOS

In order to use the testbed simulation system, problems for a DSN to solve must be developed. These problems are called scenarios. A good set of scenarios must exercise a proposed system model both in the normal modes of operations and at the boundaries of its projected capabilities. Below, several scenarios are briefly described. They were selected not for their completeness but because they present problems not adequately handled without a DSN approach.

Low-Flying Aircraft. This is a problem for two reasons: First, terrain features such as hills obscure sensors' views; thus, the set of observations is fragmented in time. Second, radars have severe clutter proolems when aimed low along the horizon. Thus, many false targets present themselves and signal reflections become

troublesome. To solve these problems, a system must be able to combine disconnected observations having a high error rate. DSN should be able to outperform conventional systems since more fragments of information are available from which to construct a track. This is particularly true if multiple sensors cover the area, and, therefore, more readings from different viewpoints are available over a longer time span as the craft moves from one area to another.

Multiple Craft Tracking. This is a problem currently receiving much attention. When more than one craft is in a single sensor's field of view, it may be difficult to distinguish them. Some tracking algorithms also run into serious computation time limitations. A DSN should be able to use information from many sensors over time to better construct a realistic world picture.

Mobile Sensors. There are several manifestations of this problem, e.g., observers with portable terminals and intelligence-gathering craft (ours). In each case, either the sensor's current location must be accurately determined or its output must somehow be used given the inherent inaccuracy of its measurment base. The system may also want to exercise control over the mobile sensor, requesting it to move to a critical region. Thus, intelligence must be used both in interpreting the sensor output and in utilizing the sensor to greatest advantage.

Complete System. In this set of scenarios, higher-level components such as C3 systems appear. This component has access to intelligence outside the sensor sources. The problem is to use the DSN to confirm or deny hypotheses generated from the intelligence and to use the intelligence to generate a more accurate world picture.

Rate of Change. The time it takes to spot and identify craft depends upon the type of craft and the type of sensor involved. For example, a radar should detect an aircraft flying at normal altitudes within seconds. On the other hand, underseas events unfold more slowly, and the environment is more noisy. Thus, it may take minutes, hours, or even days to accurately locate a submarine trying to avoid detection by operating at low speeds. Fast detection situations are relatively simple; it is known immediately whether the data is important or not. In the other case, data that is only potentially useful must be stored for long periods of time. For a DSN, this poses the issues of where to store the data and establishing responsibility for monitoring events over long time periods. It should be noted that the problems of long-term and short-term detection cannot be completely decoupled. For instance, an aircraft flying near a body of water may create phenomena that are observed by an underseas sensor. If the presence of the aircraft is not correlated with the underseas data, large amounts of information could be stored away by a conservative wait-and-see policy.

The Null Case. No set of scenarios for signal detection systems is complete without inclusion of the null case. For our purposes, this means no craft activity in the area of the DSN's sensitivity. The results of such a simulation help to characterize the profile of likely Type II errors--errors of commission.

POSTSCRIPT

Many important possible advantages of DSN systems are discussed above. However, many problems remain. These are described by Y. Yemini, "DSN: An Attempt to Define the Issues," in these proceedings.

To achieve the hoped-for benefits, it appears that DSN systems must be decentralized in addition to being distributed. Methodology for organizing and implementing such systems is not of the off-the-shelf variety. Much work remains to be done. In particular, methods of communicating for decentralized systems that are compatible with the desired flexibility and robustness must be developed. At ISI, we are engaged in producing a testbed so that ideas and strategies for DSN architectures can be investigated and competing concepts compared.

STRAWMAN DESIGN OF A DSN TO DETECT AND TRACK LOW FLYING AIRCRAFT

By

R. Lacoss and R. Walton
Lincoln Laboratory
Massachusetts Institute of Technology
Cambridge, Mass. 02142

I. INTRODUCTION.

A preliminary high level design for a DSN capable of detection and tracking of low flying aircraft and cruise missiles is being prepared. Progress in this activity is reported in this paper. The design incorporates small radar and acoustic array sensors and makes use of packet radio as the basic communication mechanism. The design is for a system which might be developed and demonstrated by the mid 1980s. However, there is at this time no plan to actually develop the system. The idea is that an attempted design of a realistic deployable system is one of the best ways to identify basic DSN problems and to evolve solutions. We expect that in the future the design process will be iterated several times with major system changes resulting from lessons learned during earlier design phases. Repeated designs and system evaluation without commitment to or actual development of a specific sytem should be a cost effective way to develop DSN technology and to eventually develop a specific DSN which is far more optimal than could be achieved by immediate commitment to a specific system development program. The design is a true strawman. It is expected to evolve into a design for a system which could be developed but it is expected that the required changes may be very major indeed.

The design summarized in the other sections of this paper constitutes our current version 1A DSN design. In the coming year the version 1A design will be the stepping off point for system analysis and for software testbed simulations which will be used to further develop the design and demonstrate how the system represented by the design might perform. The simulated systems will be denoted versions 1B, 1C, etc. The plan is that they should all be outgrowths of the 1A design but may differ substantially from it.

Future plans also call for the development and demonstration of an experimental DSN consisting of three or more nodes with real sensors. Such an experimental system, at least partially because of the limited number of nodes, will be somewhat different from the large scale systems represented by the version 1 systems. The design discussed here is not for an experimental system but for a hypothetical full scale system.

II. VERSION 1A DESIGN FEATURES.

The following are major features to be exhibited by the version 1A design or follow-on designs to be demonstrated in a software testbed:

>The System is to Detect and Track Low Flying Aircraft.

The primary task of the DSN is to detect, locate, track and identify low flying aircraft. It is to make such surveillance data selectively available to users who may be within the DSN or outside of it. Subject to physical constraints dictated by sensor capabilities the surveillance data is to be made available in real time with only small delays for processing and communication.

>Use of Multiple Cooperative Sensors of Limited Individual Capability.

The system will make use of sensors which are individually not capable of furnishing the required surveillance information. At the present time attention is focused upon short range radars with very limited azimuth measurement capability and small acoustic arrays which can measure the azimuth of sound sources. Data from multiple sites will generally be required to locate and track a target and, as a target moves through the DSN, the set of sensors reporting new measurements will continuously change.

>Distributed Intelligence is Used for Data Screening.

The tremendous amount of raw data generated by large numbers of sensors requires that computer data screening and analysis replace manual inspection, and that this be done near the sensor because limited communications bandwidth allows only near final results to be communicated.

>Adaption to DSN Configuration and Load Variation.

One potential advantage of a system which makes use of very large numbers of distributed sensors and which also distributes the surveillance function is the ability to adapt to and continue effective operation in the face of failure of various nodes in the system. The node failures may be natural, due to damage resulting from hostile action, or communication failures resulting from jamming. Adaption to changing situations is a generic feature of DSNs and must be included in version 1 designs.

>Packet Radio is Used For Digital Communication.

Packet radio is a suitable communication system meeting the requirements of rapid deployability, moderately high rate of secure digital data transfer, and unlimited system size. It is also one of several important research efforts

in digital communication, and needs evaluation in a variety of different application systems.

>Very Large Amounts of Computer Power are Applied as Necessary.

Large gains in the utility of sensors can be made using sophisticated computer processing of the raw data, and large amounts of computer processing power can now the made cheaply available using recently developed integrated circuit technologies. As a result, a large amount of computing power can be located with each sensor, with potential for improved sensor performance and good results from cheaper sensors.

III. SYSTEM CONFIGURATION OPTIONS.

There are three different general configurations for DSNs. These are area, linear, and honeycomb. The distinction between all of these are indicated in Figure 1. Figure 2 shows the area configuration in more detail. In the figure the basic spacing of 10 km is the nominal spacing for the 1A system. In the 1A system each node has a packet radio, substantial processing capability, an acoustic array, and/or a small radar. The basic 10 km spacing has been selected to be consistent with the communication system capabilities, acoustic sensor capabilities, and terrain masking difficulties for low flying aircraft. Also shown in the figure are nodes on a 50 km spacing grid which are denoted as sector nodes. These are surveillance data collection points and are discussed at greater length elsewhere in this paper.

The 10 km distance, like other parameters appearing in the version 1A design, is neither unchangeable, nor is it readily changed. Changing the value can have substantial global side effects. We have selected a single value and just note possible difficulties which might be encountered in increasing or decreasing the value.

Decreasing the 10 km value results in DSN systems which may be less cost effective. Much of the node cost is electronic, and subject to tremendous decreases in price over the next 10 years. VLSI chips may someday result in the ability to furnish node processing for only a few thousad dollars. In this case much smaller spacing options may become more interesting. This might involve an acoustic only system or the use of extremely low power cheap omnidirectional radars. Decreasing the 10 km value would then increase problems with radars interfering with each other. This should not be a problem for the strawman system since the radars do have some limited amount of directionality. Increasing the 10 km value gets beyond the useful range of acoustic sensors, and introduces masking problems with radar.

The version 1A design effort has concentrated upon the area coverage geometry. Primarily this is because the uniform distribution of nodes will tend to generate a greater communication load than other configurations. Thus if the communication issues for the area geometry can be handled the others should cause no difficulty. The design could be adapted to either the linear or honeycomb stuation. In the case of the linear configuration the overall system would tend to be much simpler. For the honeycomb the system might be slightly simpler but the problem of data association and tracking across sensor free areas would be more difficult. With respect to this last comment note that in the long run an area system should function when large holes are torn in the net. Thus the area system ultimately must treat this problem of association and tracking across sensor free areas.

IV. OPERATIONAL MODES.

Our strawman DSN has both small acoustic arrays and small radars at each node. This provides for operation in a number of possible modes. Following is a brief description and discussion of three such modes. The discussion is in terms of modes of a single system but it should be clear that they could also correspond to systems using different mixes of sensors at the nodes.

>Acoustic Only Surveillance and Tracking.

Detection and tracking can be accomplished using only measurements made by small acoustic arrays located at each system node. Individual arrays are used to get a series of single node detections and estimates of possible target azimuths. Such data is combined from several such nodes both to perform surveillance for new targets and to track targets.

In such an all acoustic mode the system is passive except for the communications. It would use less power than with radars in operation and the elements of the DSN would be less detectable than with the radars in operation. The lower detectability follows because radiated power needed to communicate over 10 km distances is very much less than the radar power needed to detect targets at 10 km ranges. However in an all acoustic mode the quality of present location estimates and predictions of future positions would probably not be as high as would be possible when making use of radar. This is as least partially due to the inherent acoustic propagation time delay to the sensors.

>Radar Only Surveillance and Tracking.

Each DSN node has a small radar which can give good range information but very limited azimuth and no elevation information. As noted above if the system operates using only these radar sensors it would consume more power, be more detectable, but probably be capable of more precise estimates of present and future posi-

tions. The improved locations would follow from multisite combining of range measurements and not from the use of radar azimuth measurements. Since the radars furnish information for both surveillance and tracking the most obvious mode for individual radars is to continuously scan and search for targets. The system can perform tracking by combining reports from the individual radars which are physically functioning in the search mode.

A hybrid mode would be for some radars to scan some of the time and to physically keep beams pointed at specific targets some of the time to accomplish better tracking of already acquired targets. Multiple sites would still be required for good location but range measurements for targets being tracked would not be constrained to one observation per radar scan time in the surveillance mode. The utility of this hybrid mode and the impact upon radar complexity and cost are issues to be considered.

It should be noted that the hybrid mode assumes enough radar redundancy so that some fraction of the radars can be diverted from the surveillance task and redirected to improve tracking information on selected targets. This observation is true independent of whether radars are mechanically or electrically steered since the transmitted radar beam will be directed in only a single direction at any one time.

Our strawman system does not contain a directional acoustic receiver which is physically moved to point at targets. Thus there is no acoustic only hybrid mode which is entirely equivalent to the hybrid radar mode just mentioned. The closest equivalent would be to selectively process data for different directions and frequencies rather than to routinely scan the entire space of interest. This could potentially reduce processing loads but we do not currently know how to do this without compromising performance.

>Simultaneous Acoustic and Radar Surveillance and Tracking.

Making use of both acoustic and radar sensors is the option which generates the richest system alternatives. Rather than try to mention a large number of them we cite only one simple scenario which represents a prime alternative to be investigated.

To minimize system detectability and to conserve power a potentially useful mode of operation is to perform surveillance using acoustic sensors and to operate radars only upon cue for tracking. A system operating in this way would handle a target entering the DSN from the outside the coverage area as follows. Initial detection and rough location is obtained acoustically as the target is entering. The radars in the immediate target area and adjacent areas are triggered into operation. As the target goes deep into the DSN the cueing of radars will be from extrapolation of target tracks. Internal acoustic sites will furnish backup as well as continuous surveillance and target signature information. Our current view is that individual radars will operate in the scanning mode with tracking done from the radar measurements obtained. However the more complicated radars suggested in the context of the all radar hybrid mode above could also be considered.

V. MULTISITE PROCESSING.

Multiple site target detection and tracking functions are to be performed at every DSN node independently of other nodes. Each node will use all the information available to it for this purpose. The input information will consist of sensor reports and tracking reports received from neighbor nodes. In general only reports received directly on a single radio hop from a neighbor will be required or used for basic multisite processing. Figure 3 shows the 12 nodes which we assume would nominally receive a one hop sensor report from a single node. This assumption is discussed more in Section VII of this paper.

The sensor reports generated by a node summarize target information which can be obtained by analysis of the data obtained from the sensors at that node without reference to any other nodes. For acoustic sensors this can be a list of the most likely directions for targets and a summary of spectral characteristics for sound from each of those directions. For radar this can be a list of possible ranges and rough azimuths plus some target doppler information. Tracking reports combine sensor reports from several nodes plus other available tracking reports to give actual target positions and trajectory information in a directly usable system coordinate system. The tracking reports may also summarize other target signature or identification information.

There are a number of reasons for making every node a multisite node. One is the built in redundancy which this achieves. In effect a target may be under surveillance by and be included in the tracking reports of several nodes. Loss of a few nodes, apart from the fact that sensor data from those nodes will be no longer available, will not have a serious impact upon system performance. Another factor is the anticipated computational load of multisite processing and sensor processing. In the current design the computational requirements for sensor processing at each node are very substantial. A belief yet to be confirmed is that basic multisite processing computational load is not substantial relative to the basic sensor processing load. Thus the cost of routinely doing multiple site processing with high redundancy will not substantially add to computational hardware in the system. Another factor involves communications. With each node being a multisite node we are able to design a system in which all sensor

reports are made on a single communication hop. By excluding multiple hop routes for sensor reports we avoid an increase in sensor communication requirements which would occur for multiple hop distribution.

At this time the multisite processing of radar data has not yet been specified and only the general ideas of minimal acoustic multisite processing have been considered. We include here a brief discussion of an approach to multisite tracking for acoustic data. More specific and detailed algorithms and procedures will be specified in the process of implementing simulations of a functional DSN.

First we consider the multisite use of acoustic data for location. Measurements of acoustic azimuth are made at each DSN node every two seconds. The node locally tries to associate successive observations so that it can deliver sequences of azimuth measurements which are associated with a single target. The node cannot locate the target but it can often furnish this association between sequences of observations. Thus the multisite process has available as input a collection of lists of azimuths from several sensors with each list corresponding to measurements of a single target from a single node. Some lists may have weak links and some lists which should be a single list may be broken into two or more due to difficulties in correctly associating sequential observations. The basic multisite task is to use these data to locate and track targets.

As described in more detail in another workshop paper --"Multisite Processing of Acoustic Azimuth Measurements to Locate and Track" by R. Walton--each azimuth versus time curve can be mapped into a possible position curve in the horizontal plane. A surveillance time T, earlier than the present time, is associated with a possible position curve and the mapping from azimuth versus time to possible positions depends upon T. Imagine that T is fixed and that all of the possible position curves are displayed using a different color for each contributing node. It should be relatively easy for a person to locate targets using such a display. The targets are where segments from different nodes intersect. The DSN will require a program to sift this display and find targets. Results reported in the workshop paper by Walton show good results with only 2 nodes and only one target, provided the target is not in certain areas relative to the 2 nodes where its position becomes ambiguous.

Exactly how radar will fit into the picture depends on yet undecided details of the radar and how it is to be used in the system. The strawman radar capability discussed elsewhere in this report supplies good range information but poor azimuth resolution. Each such radar measurement then corresponds to a possible position curve which is a segment of circular arc. This fits nicely into the above picture if the time of the radar observation is the time T. Such radar possible position curves would typically be shorter than those generated by acoustic sensors. This could be used to resolve ambiguities and possible ghosts which could not be completely resolved using only the acoustic data. However, there may be problems when the radar observations do not occur at time T. This is a problem to be investigated in conjunction with continuing design and first functional simulations of a DSN. One possibility is that radar data, as mentioned in our discussion of system modes, will not be used extensively for initial target detection and location but will be used only for refining tracks for targets (and false targets) tentatively detected and located acoustically. This would be the case if individual radars were focused upon specific limited potential target areas rather than being operated in a general surveillance mode.

Independent of just how radars are used there is always the problem of refining the track estimate of a target given the ever increasing amount of data. There are considerable possibilities for track improvement as more data arrives. Most likely some form of Kalman filtering will be used for this purpose.

VI. REPORTING TO SECTOR NODES.

For reporting purposes, DSN geometries are divided into reporting sectors. Each sector has at its center a sector node with which system users interface either directly or through communication links. The reports from tracking programs operating at every node in the sector must be summarized and surveillance information for the sector must flow to the sector node. The flow of reports from the edges of the reporting sector to the sector node is by means of a patterned sequence of broadcasts. During normal operation the reports within a sector are collected with time delays on the order of a second. Figure 4 shows the configuration of reporting sectors for normal system operation. Normal operation is here defined to be operation when all sector nodes are functional. The reporting sectors are defined so that the sector associated with a sector node constitutes all of the area which is closer to that sector node than any other sector. As shown it happens that in addition to the sector node itself the reporting sector contains 30 sensors of which 12 are on the boundary.

Reports flow from the edges of reporting sectors to the sector nodes as follows. Each node produces and broadcasts a report roughly once each second. This report is the best report summary that the node can generate for an area around the node which is known as its report area of interest. Figure 4 also shows the area of interest for a typical node when the DSN is in full operational condition with all sector nodes functioning. Each node's report covers a particular area and the areas for different

nodes overlap very much. In prepairing its report a node uses all reports it has heard as well as its own multisite processing of sensor reports. With reference to Figure 4 it is clear how the sector node can generate its picture of its report sector from the reports it obtains from neighbors. It should be noted that report coverage is very redundant so the sector node can get reports for a particular part of its sector from many paths. This will automatically compensate for malfunctioning nodes, for communication errors, and for missing paths. Of course the sector node will also produce and broadcast its sector report once each second.

Under normal operation as described above each node in the DSN produces and broadcasts a complete summary report once each second for a region of the same size and shape as the report sectors associated with the sector nodes. Each node's report is limited in length to about 1500 data bits. This is enough to accommodate 10 targets. If there are up to 20 targets the extra targets will be reported at the cost of halving the reporting rate by sending half the report each second. Targets in some region may be preferred and sent every second. Beyond 20 targets it will be necessary for the node computers to pick targets not of interest to be discarded from some reports. For example slow targets may be reported 2 seconds out of every 8. The performance of the system is not optimal above 10 targets and the system behaviour is not yet specified above 20 targets.

Saturation of the 1500 bits of a report by excessing numbers of targets is a problem which can be more serious when some sector nodes are not functioning and their report sectors must be absorbed into other report sectors. Figure 5 shows the redefinition of report sectors which results when a single sector node fails. Each neighbor has increased the area of its report sector by one sixth and problems can result. The report area of interest for nodes other than the sector nodes, if unchanged from that shown in Figure 4, are not sufficient to accomplish the flow of surveillance information from the area near the failed sector node to the still functioning sector nodes. The shape of the report area of interest could be maintained and the size increased. However this would mean that the same number of bits would be available to summarize a larger area. The result would be a reduction in the target density which could be handled. The alternative actually selected is to allow different shapes of report area of interest for different nodes. For example nodes near a functioning sector node and on the side toward a failed sector node will have adjusted areas of interest to encompass areas further away towared the failed node without increasing the size of its area of interest. The nature of the communication pattern used to cause reports to flow to sector nodes is such that reports from the area near the failed node will reach functioning nodes with delays on the order of two seconds.

These issues would be more serious if more nearby sector nodes were to stop functioning.

VII. COMMUNICATIONS.

The DSN system will make use of packet radios for communication. However, the DSN application and traffic is quite different from that for which packet radio networks have been designed. The normal PRN application is for large numbers of independent nodes each generating bursty traffic with low average rates. Although there are statistical delay specifications, occasionally very large delays may be experienced and tolerated. Errors and lost messages are universally not tolerated and the system is designed and organized to enforce this as much as possible. In general there is no quaranteed performance for any individual user.

The DSN requirements differ substantially. The basic DSN is an organized set of nodes working cooperatively. It is most likely that high target density in an area will cause DSN communication traffic to be high in that area. Satisfactory performance must be assured at such critical times. In particular some DSN functions will require assured data rates and delays even in the worst conditions. Fortunately the fact that the DSN nodes are not independent should allow us to coordinate their communication and thus provide this essential guaranteed response. In some cases lost messages or messages received in error (which can be treated as lost messages) may be somewhat tolerable.

In this section we review a very simple packet radio model being used for preliminary DSN communications system design and analysis, review the major DSN communication requirements, and outline a strawman approach for using packet radios to furnish the required service. Only area surveillance with a uniform grid of sensor nodes is discussed. This is the most stressing of the DSN system system geometry options from the communication point of view. We have used a regular hexagonal grid with the understanding that it must be possible to adapt to any reasonable regular or irregular distribution of sensors.

VII.1 Assumed Packet Radio Characteristics for Initial Design Purposes.

The Distributed Sensor Network project has decided to use packet radio for communications. In general the characteristics of a Upgraded Packet Radio are assumed if they differ from the current packet radio.

Our model of the packet radio is very simple. We assume that in any given 10 millisecond period a packet with up to 1500 data bits can be transmitted. The actual number of bits in the packet may be as large as 2000 with the extra 500 being error control and other information not considered specifically to be data. We

assume that the error control allows the receiver to detect and discard all packets received in error. Nominally, this packet might be heard by any neighboring node 10 to 20 km away, with error rates discussed below.

We assume that the time a packet is transmitted is under the control of the node's DSN computer. We assume that a node cannot receive more than one packet at a time. We also assume that if two nearby neighbors transmit overlapping packets, then the two packets will collide and neither will be received correctly. (We do not use the potential of Upgraded Packet Radio for receiving the first packet and ignoring the second.)

For the purpose of preliminary system analysis required to confirm that the DSN design might work we have made simple assumptions about communication network connectivity and probability of correct reception of a packet over a single hop. We assume that 70% of all single hop transmission paths of lengh 10 km will be useable. The rest will not be useable due to siting, transmission path effects, and masking. For paths of length 17 km (second nearest neighbors) the probability of useability is reduced to 50%. Longer paths are assumed to not be useable at all. Finally, the single hop error free transmission over a useable path is assumed to be 70%. Such figure should be achievable in practice.

VII.2 General Communication Requirements.

The DSN communication system must furnish several kinds of service. These are: (1)-Sensor reports service, (2)-Surveillance reports and command distribution service, and (3)-Special point to point service. Each of these is discussed briefly below.

>Sensor reports service.

Each sensor node will deliver a sensor report once each second to all of the tracking sites which are within one communication hop. Such reports are a summary of the results of processing recent sensor data. In general a node may take more that a second to update its local world view based only on its own sensors. For example the current design calls for this to be done one every two seconds for acoustic data and once every four seconds for radar. Thus with this situation more than one sensor report will be available to distribute each new world view generated from the sensors at the node.

A sensor report will contain up to 2000 bits. Of these 1500 may be data and the rest communication headers, error detection, time and site information, and other overhead. Given a message of this size each tracking node will receive (6x0.5+6x0.7)x2.0=14.4 kilobits and transmit 2 kilobits every second. This follows from the geometry and the crude link usability model discussed above.

There may be circumstances in which a single hop link from a sensor to a tracking node which should use it will not exist. Tracking computers will be allowed to arrange to obtain data from selected multi hop paths but only in so far as the system has capacity to supply this data without harming the operation of the overall system. This will be done using the special point to point service discussed below. It is generally discouraged. The communication capacity for such multi hop service is part of the capacity required for special service and is not considered to be part of the basic sensor report service. The basic sensor report mechanism is to be one hop.

The detailed use of the 2000 bits in a sensor report is not yet specified. However it is not likely that the number of bits (and thus the traffic) can be substantially reduced. Although the typical number of real targets in range of a sensor will be small--perhaps one or even less--the system must be capable of handling several targets and, as a direct result of trying to report targets with poor signal to noise ratios, will normally deliver several false alarms each reporting period. Thus if only 10 possible targets are reported (sum of actual target measurements and of possible ones which in fact cannot be verified by multiple sites) we find that only 150 bits are allocated to each possible target. In general we can think of the sensor report as a summary of important characteristics of frequency-wavenumber or range-azimuth-doppler power distributions rather than target reports as such. From this viewpoint we could use some 150 bits to describe each of 10 interesting looking peaks in those maps.

>Sector Surveillance and Command Distribution Service.

The DSN trial design includes surveillance computers located every 50 km on a uniform grid. Each of these collects a summary of target information over the whole area within about 25 km of where it is located. It does this by collecting surveillance information from tracking computers within its surveillance sector. Tracking information is passed inward towards sector nodes as described in Section VI.

Somewhat arbitrarily we will require that the DSN be able to report once each second on at least 10 targets in the normal sector surveillance area without regard to where in the area they are located. If more targets are present it may sometime be possible to get one second reports but they will not be assured. In general for more targets in the area the sector surveillance computer can either be selective of the targets it tracks or use more than one second to complete a single surveillance survey of the sector. If sector nodes are not functioning so that sector surveillance areas are redefined and expanded larger delays are allowed. Specifically, approximately one second delay is allowed for each normal surveillance sector which a re-

port must traverse.

Some 1500 bits should be sufficient to represent the surveillance information for 10 targets in the area. If so a node broadcasting its normal area of interest surveillance report as discussed in Section VI will broadcast 1500 data bits with 500 bits of error correction and other packet header information. The concept that each node will broadcast a surveillance report each second results in 14.4 kilobits incomming surveillance reports and 2.0 outgoing at each node as was the case for sensor reports.

In addition to reports flowing into sector nodes there will be a flow of commands from the sector nodes to the nodes within the surveillance sector. This will be low data rate compared to the reports themselves.

>Special Point to Point Service.

Some users will need assured real time point to point service in the DSN. For example, above we mentioned alternate routing when the standard one hop sensor report distribution service is not adequate for a particular tracking computer. Such users should be guaranteed a communication rate and delay. The circuit will have assigned a probability that any particular message will actually be transfered from end to end. That probability will usually grow linearly with the number of hops in the overall circuit. At any node along the route of the circuit the user must be able to extract and add data to messages as long as the outgoing link remains within the rated capacity of the circuit. There will be needs for assured service with various capacity. For example the alternative routing of sensor reports to tracking computers would require 2kb circuits. Here we only discuss the maximum assured capacity which the DSN must offer for users.

A point to point user could be located at a sector surveillance site and need to examine in detail a great deal of the data at a single tracking site in the sector. In this case the rate to the tracking site is small--limited to a few commands--but the rate into the sector surveillance site could be large. The tracking node of interest could change as often as every 30 seconds to keep up with a target flying close to the speed of sound. On occasion such data may also be passed on to a user many surveillance sectors away.

The high rate data of interest might be sensor reports, sections of frequency-wavenumber power maps, range-azimuth-doppler maps, or even raw data. Acoustic data is one likely kind of raw data. It should be possible to at least obtain raw data from a single channel of acoustic data. Such data is gathered at a 2 kilohertz rate with 16 bit samples. Assuming a very nominal factor of two data compression this will give a 16 kilobit per second data rate. Other data of interest might be all the sensor data reports being used by a particular tracking computer. This would amount to at least 8.2x1.5 kilobits per second and probably somewhat more. Another might be a radar clutter map of 100*36=3600 power values. Based on such considerations we require the DSN to be able to deliver point to point service at rates up to 17 kilobits per second.

If 16 khz circuits are to be supported then any node must be able to deal with 32khz total circuit traffic at least. This is the sum of the incoming and outgoing traffic. It can be split arbitrarily between input and output in general and each stream can in fact be any number of smaller capacity circuits. Each of the individual circuits will have its own error rates and quarateed capacity and delays.

Delay characteristics for point to point transmission should be similar to those for surveillance reporting. That is, the delays on the order of one second per surveillance sector traversed are to be allowed.

In general a request for circuit service must be made, the system will decide if it can be supported, and it will be granted or denied. Priority requestes may result in the ungranting of currently quaranteed service.

VII.3 Strawman Communication Organization.

The communication needs for DSN are considerable. As outlined in the requirements section above the message traffic (sum of incoming and outgoing) at a single node can easily be 60 kilobits per second or more and the delays must not be excessive. To achieve this kind of throughput using packet radio will take special effort. Suppose the basic packet radio rate is 200 kbs. Random access or carrier sense access to such a channel could seriously constrain DSN performance. Their are various options to help the situation. One is a system very much like packet radio but with rates increased by almost an order of magnitude by increasing bandwidth. A second is to use a layered packet radio network to effectively increase capacity. This would involve running several packet radio networks simultaneously in the same frequency band but with different codes. Each radio would have multiple decoders and multiple encoders of the signal. A third option is to recognize the structured nature of the DSN system and to make use of this to coordinate node transmissions so that the available channel is very heavily used and self-interference is minimized. This last is the option which is further developed in this section to indicate how a packet radio with a 200 kbs rate might furnish all DSN communications. Minor changes such as having a broadcast mode as well as a point to point mode and allowing packets which do not get retransmitted if not acknowledged will be required.

>Overview of Version 1A Circuits

Version 1A DSN communications is modularized by splitting all communications into separate "circuits". A circuit is a particular conceptual and algorithmic construct for communications. A circuit is a time ordered geometrical pattern of packet transmissions. Roughly it is a statement that specific nodes will transmit packets in some specific order during a given time. The algorithmic details of chosing which nodes transmit when are flexible. But there must be an explicit algorithm for each type of circuit.

It is necessary to assign specific time intervals to each circuit. Some circuits have the potential for interferring substantially with other circuits, and must be separated in time. Other circuits may not seriously interfer with each other, and need not be separated in time. It takes 10 milliseconds to transmit a maximum length 2000 bit packet. Therefore we will divide each second into 100 intervals of 10 milliseconds each, and assign some number of these 10 milli-second intervals to each circuit.

Version 1A defines three kinds of circuits: the sensor circuit, the report circuit, and special (query) circuits. These correspond to the three kinds of services discussed in the requirements section above. Of the 100 10 millisecond time intervals in each second, 25 are allocated to the sensor circuit, 25 to the report circuit, and 50 to the various special circuits.

The sensor circuit moves messages from each node to all its neighbors. There are no acknowledgements, and any data that is to arrive with high probablility must be retransmitted repeatedly.

The report circuit moves messages from each node to sector nodes, which in turn are expected to move the messages to a manned computer center. The report circuit also moves occassional control messages in the reverse direction, from the sector nodes to all other nodes. Acknowlegment is optional for each message. Because of the many paths available to or from sector nodes, an unacknowleged message has quite a high probability of arriving at its destination, though this probability goes down when report circuit traffic becomes very heavy and the number of paths is reduced to accommodate the load. Report circuit messages are broadcast so they may be heard by all surrounding nodes, and thereby are also the means by which each node knows what its neighbors are tracking.

Special circuits are established on request between any two nodes for data rates up to 17 kilobits per second in one direction, and a low data rate in the reverse direction. Usually these circuits are used for querying a node, and transmit data principally from that node to the nearest sector node. Acknowlegment and retransmissions are automatic for special circuits, except that messages can be marked to be thrown away if the backlog in the special circuit exceeds one of several limits (which in normal situations happens very rarely).

Higher level system functions will generally use both report and special circuits.

>Communication Patterns.

Simultaneous broadcasts by packet radios separated by at least 50 kilometers cannot interfer with each other for any DSN purpose. Therefore we want to organize communication so that only nodes separated by at least 50 kilometers will broadcast simultaneously. This is done by breaking the DSN into communication cells with 50 km sides and arranging for no more than one node in a cell to broadcast at any given time. We illustrate the idea here in terms of the reports circuit.

The report circuit performs three functions: it conveys report messages from arbitrary nodes to sector nodes, from whence they are sent to manned control centers; it broadcasts tracking results from each node to all neighboring nodes, as an aid to the tracking programs in neighboring nodes; and it broadcasts control messages from each sector node to all surrounding nodes.

The report circuit is a circuit with a typical broadcast pattern as shown in Figure 6. The nodes within the parallelogram constitute one communication cell. Such cells are repeated throughout the DSN. With rather minimal local coordination the pattern will effectively avoid communication conflicts. The pattern is repeated once every second. As shown the sequence of broadcasts is 1A...1F, 2A...2F, 3A...3F, 4, 5A...5F. Such a pattern can be used to move data towards the sector nodes or away from the sector nodes.

To see the flow away from the sector node consider the cycle starting with 4 and proceeding 5A...5F, 1A...1F, 2A...2F. This can be used to move data away from one sector node toward another in case the first sector node is nonfunctional, or has lost touch with the manned computer center. Because it is capable of moving data away from sector nodes, the report circuit is also used to broadcast control messages originating at manned computer centers and routed through the sector nodes.

The sequence 1A...1F, 2A...2F, 3A...3F, 5A...5F (4 is not used) obviously can be used to move data towards the sector node. Since the cycle is repeated once each second reports from the more distant nodes in the sector will be refreshed each second.

It should be noted that this communication scheme assumes that the DSN can control the time of a broadcast. Also the flow over multiple hops is not accomplished strictly by multi hop communication paths under control of the commun-

ication system. The DSN computers have access to messages in transit and modify these messages before they are retransmitted. For example, in moving data toward the sector node the node at 5C in the pattern has probably heard 1C, 3B, and 3C with enough lead time to incorporate their information in the packet it will transmit.

VIII. SENSORS AND SENSOR DATA PROCESSING.

As noted in several other sections of this paper each strawman DSN node includes a small acoustic array and a small radar. In the following brief paragraphs we summarize the general strawman sensor characteristics and processing load required for those sensors. These sensors and loads are only representative. This is particularly true of the radar option for which there is a need for considerable design and analysis work before a sensor can be proposed with much confidence. Low flying targets in general and cruise missiles in particular do present difficult sensor problems. More strawman sensor and sensor performance details will be found in the papers "Single Node Detection and Target Parameter Estimation" by P. Demko and "Acoustic Sensor Capabilities and Performance" by T. Landers which appear elsewhere in this workshop proceedings.

Each node in the strawman design includes an array of 10 high quality microphones deployed over a plane aperture of 3 meters. Digital data are collected at the rate of 2000 samples per second per channel. Once every two seconds a one second sample of array data are processed to detect targets and estimate the direction from which sound is comming at a large number of different frequencies. Alternate one second intervals of data are not collected and the time may be used from sensor calibration or other purposes. Single node target detection and direction estimation makes use of high resolution frequency-wavenumber signal processing. Detection and estimation depend upon finding increases in the amount of power which appears to arrive at the array as a function of direction and frequency. The single node processing load amounts to about 11 million real adds or multiplies per second.

A small monostatic, coherent pulsed doppler, two dimensional radar has been tentatively selected as the strawman DSN radar. The strawman radar at this time is not a serious proposal for a specific radar but rather is a somewhat arbitrary selection of some possible radar characteristics for the purpose of sizing computational requirements. The radar frequency and other characteristics are subject to substantial changes as a result of further study. Even the type of radar may be changed but it is likely that our present selection would be the most computationally demanding alternative so it has been selected as worst case starting point. The strawman radar is L band (1.3 GHz). It uses a 10 kHz pulse repetition rate to operate out to about 12.5 km ranges. The antenna has about 10 degree azimuth resolution and 30 degree elevation beamwidth. The radar scans 360 degrees using 36 beams displaced from eachother by 10 degrees. Nine beam directions are used each second. State of the art digital processing is used to detect targets in radar ground clutter and to make most effective use of doppler shifts resulting from target motion. Targets are located within 120 meter range cells but with poor direction information. The processing load, about half of which is FFT amounts to 25 million read adds or multiplies per second for a single such radar.

IX. NODE HARDWARE CONFIGURATION.

The hardware configuration for each node in the Version 1A DSN is shown in Figure 7. Neither the sensor nor the packet radio will be discussed further in this section. The other electronic hardware is assumed to be built with currently existing integrated circuits but higher level elements of the system are not now commercially available. We have specified this hardware to avoid compromising system performance due to the use of slower commercial equipment. The specification of considerable computer hardware at each node will give the most flexibility for software options. Even more computer power would be technically feasible but no well defined requirement has yet developed. The option shown is sufficient for the worst case radar and acoustic signal processing at a node with enough spare power for other DSN functions. Ongoing design, simulation, and experimental efforts will be used to determine if the processing capability represented by Figure 7 will actually be required.

Except for a small amount of sensor specific and interface electronics the electronics consist of a main computer, memory, and single instruction multiple data stream processor (SIMD) which could satisfy all known processing requirements for a node. The outline of a design for the main electronics has been completed and the numbers of integrated circuits required for each electronic subsystem has been roughly estimated. For the estimates obtained and shown on the Figure the cost of the entire node built to MIL SPEC would be $120,000 if we assume that system cost is propostional to the number of itegrated circuits and the system cost per circuit is $60. The $60 figure is reasonable at the present time and will probably decrease by a substantial percentage, between 30% and 50%, each year in the immediate future. The design and these estimates do not make use of special developments in very large scale integration or of special custom chips. Such alternatives could substantially reduce cost, size, and power consumption.

The main processor serves for general purpose computing and also as an input/output channel processor for main memory. In its role as a

general purpose processor it will perform all processing at the node which is not more appropriately done by the faster, essentially parallel, SIMD processor. This could include a substantial amount of sensory AI processing. Such sensory AI would be to make more effective use of multidimentional power maps produced by single sensor signal processing as the first step toward target detection and parameter estimation. The strawman design has the main processor running an extension of the PDP-11 instruction set so that it can emulate the PDP-11 with high efficiency. The main differences would be the following two. First, instruction lengths are not the same: the instructions have been reformated to allow 32 bit addressing, 32 registers, etc. Second, instructions must be treated as read only because the main processor design includes an instruction cache which is not updated by data writes.

The main processor, although it is 20 times faster than typical commercial minicomputers in tight loop arithmetic computations, is still not fast enough to handle the several tens of millions of arithmetic operations per second required for full acoustic and radar data processing. For this purpose a special arithmetic processor is employed. This processor is a 10 element SIMD: a single instruction stream, multiple data stream processor with 10 data units and one instruction execution unit. Both acoustic and radar processing at a single node naturally decompose to be efficiently accomplished using such an architecture. The SIMD design takes advantage of currently available high speed 16x16 multiplier chips.

Almost all circuitry is intended to be low power. Micro-cycle times are 250 nanoseconds, and only a little processing is done in each micro-cycle. For example, single precision floating point adds are done in 750 nanoseconds in the main computer, whereas Schottky circuitry would do them in only 200 nanoseconds. The main exception to low power are 16x16 multiplier chips to be used by the SIMD processor. Each of those chips consume several watts of power.

The processor briefly outlined here is not presented as the only practical option for the node electronics. We have pursued a specific option as part of our strawman design only to be able to put on display one option which is feasible and will satisfy our requirements.

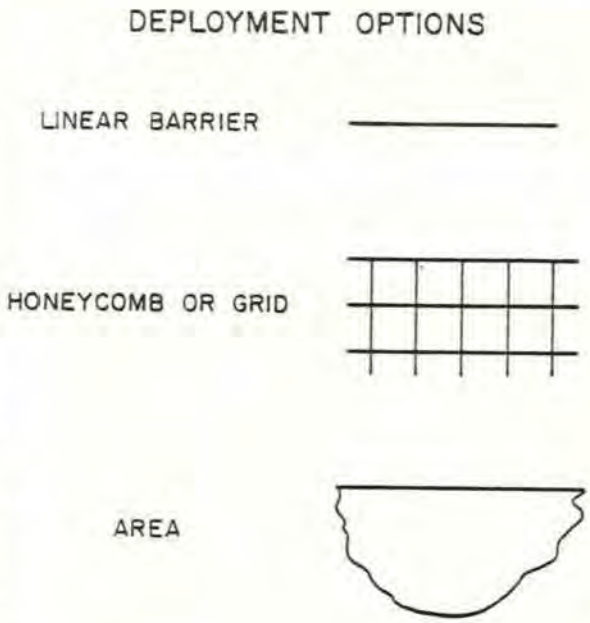


Fig. 1. The three basic DSN deployment configuration.

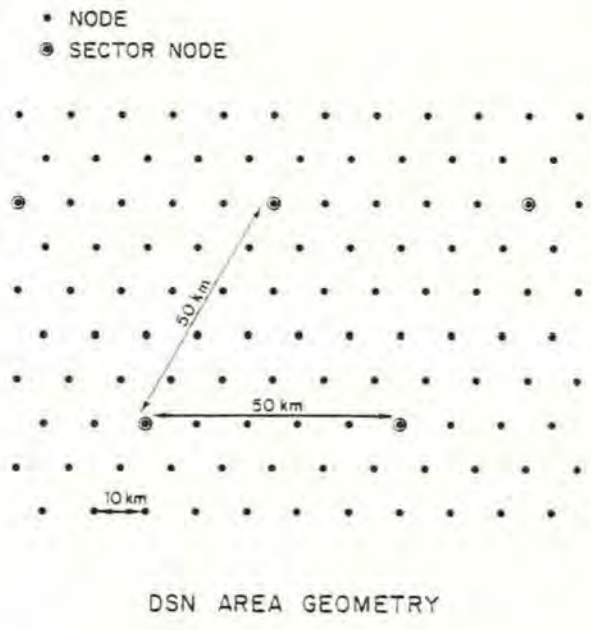


Fig. 2. The strawman area deployment configuration.

AREA GEOMETRY SENSOR CIRCUIT
INFORMATION FLOW FROM A NODE TO NEIGHBORS

Fig. 3. Nominal single hop receptions of a node sensor report.

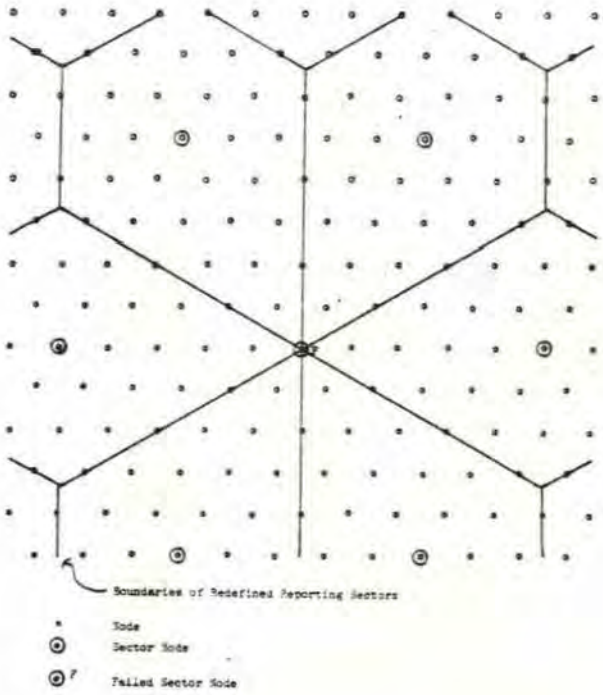


Fig. 5. Redefinition of surveillance sectors when a sector node has failed.

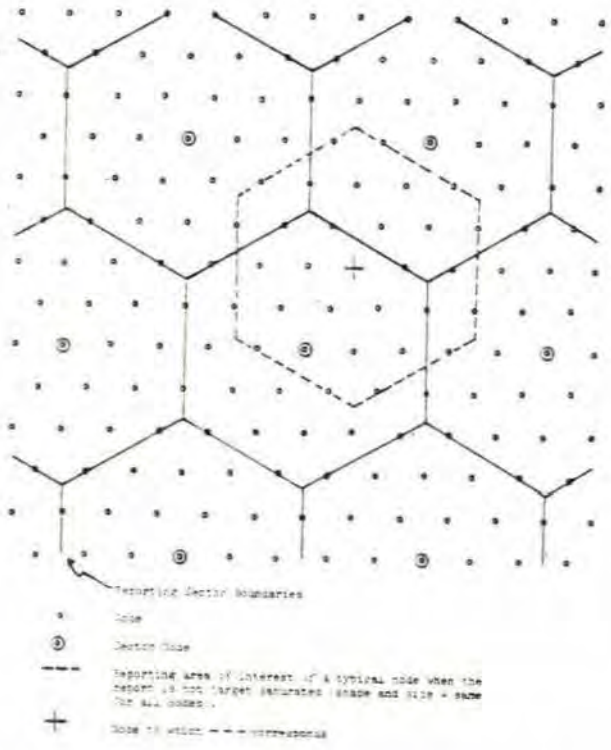


Fig. 4. Surveillance sectors and node areas of interest with all sector nodes operational.

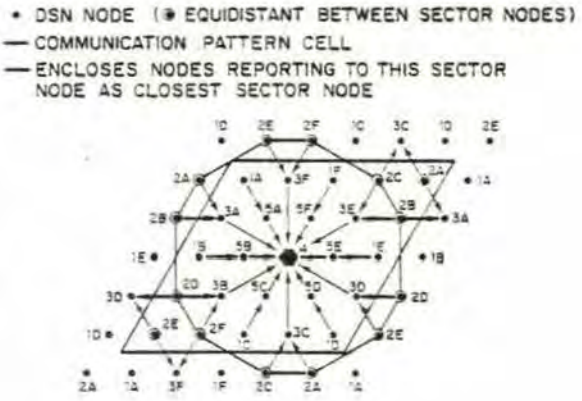


Fig. 6. Report circuit broadcast pattern.

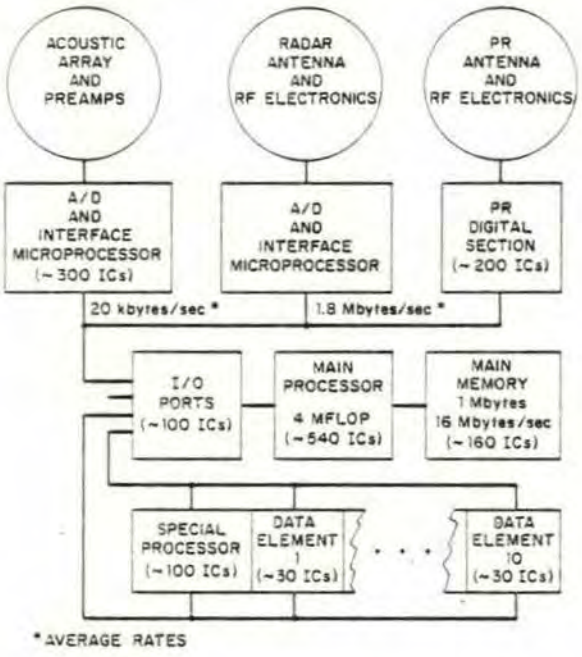


Fig. 7. Strawman node hardware configuration.

DISTRIBUTED SENSORS NETWORKS (DSN):

AN ATTEMPT TO DEFINE THE ISSUES

YECHIAM YEMINI

USC/INFORMATION SCIENCES INSTITUTE

1. INTRODUCTION

This paper draws upon ideas from a number of sources and persons, in particular, lengthy discussions with J. Barnett and D. Cohen, whose invaluable help I wish to acknowledge. However, I bear the sole responsibility for the opinions expressed, the errors made, and the final form of the paper.

1.0 Objective of this paper

It seems appropriate at this early stage of research to take a global view of the DSN problem. This paper presents an initial attempt to obtain this goal; it should be read as a working paper whose main purpose is to trigger responses and focus attention on some of the issues.

The method we have chosen to help define the issues centers on identifying the potential functions of DSN. These issues include intelligent integrated signal processing architecture, developing architecture to support decentralized cooperative processing, and overall design problems.

Finally, the development of DSN, integrating three major technologies (i.e., sensor, computer-communication, and information-processing technologies), will require expertise in many fields. We have compiled an annotated bibliography of some key papers, survey papers and books that may help the reader to gain insights into the scope and focus of some relevant technologies.

1.1 DSN, some terminology.

By a feature-set we mean a set whose elements represent possible values of some target parameter (e.g., position, heading, type of aircraft, etc.). A target feature is a time function assuming values in some feature-set (e.g., the track pursued by a target, the velocity of a target, the spectral formants of the noise emitted by a target, etc.). A target is an object consisting of a set of features and certain relationships between them. A sensor is a device that can observe a function, to be called observation, of some target-features. The observation depends, in addition to the feature observed, upon two sets of parameters: intrinsic sensor's parameters (e.g., the object of the sensor's observation, the rate of sampling, internal noise, etc.) and extrinsic parameters, resulting from the interaction between the observable feature and the sensor's environment (e.g., limited field of view, weather conditions, etc.).

A DSN consists of a set of sensors, a communication network, and a set of processing elements. The sensors may observe targets in a given area; the information collected by the sensors is then communicated to elements that process the observations and communicate between themselves to extract target-features. Note that we classified the network elements according to functions; traditionally, a few of those functions are sometimes incorporated into a single device, the borderline between functions not always clearly marked.

1.2 DSN, an inference-derivation system.

A DSN may be thought of as a large inference-derivation system. The process of inferring is based upon some prior knowledge about relations between target-features and observations; it may consist of both logical and statistical derivations and follow, typically, the Bayesian inference paradigm. That is, the observed data is employed to improve apriori knowledge of targets. The inferred posterior knowledge serves as the new prior knowledge, on the basis of which new observations are to be explained. Accordingly, the state of the inference process at each moment consists of the set of all present beliefs that it holds. This state is modified as new observations or inferences are made.

Loosely speaking, the intelligence of a DSN consists of its ability to recognize, learn, and use relationships between features to derive inferences.

As an example consider a DSN that observes the acoustic emissions and radar reflections from a set of low-flying aircrafts. At the lowest level of recognition, the DSN tries to infer target parameters such as azimuths, ranges, and velocities from the observations. Once these higher-level features are extracted, the DSN needs to relate the parameters extracted from the acoustic noise and those extracted from radar reflection. The next step may be to compare the present estimation to information acquired in the past in order to

extract features of the tracks pursued by the targets. At still a higher level of intelligence, it may be possible to extract information about a possible enemy-plan from a set of target tracks and their composition.

Although higher-level features may be inferred from combinations of lower level features, the inference process is not unidirectional. In fact, higher-level recognition of features can be fed back to lower-level inference derivations to improve their prior information. This typical Bayesian approach to inference can serve to cut down the combinatorially explosive size of processing at the lower-levels. For instance, in an acoustic array the spatially distributed observations of acoustic emissions are temporally and spatially correlated. The correlation function is analyzed spectrally to produce a map of the power spectrum as a function of the vector wave number. The azimuth to the source of emission is found by a maximum likelihood analysis of the map. The process of mapping the power spectrum in the wave-number plane involves an enormous amount of computation. The amount of computation may be radically cut down if the algorithm had some prior "clues" on the nature of the source (i.e., expected direction of arrival, expected speed, spectral signature) from some higher-level recognition process. Such clues can be used to reduce the range of wave numbers to be searched.

1.3 DSN, the integration of communication and processing.

Let us consider some of the results of distributing DSN components (i.e., sensors and processors). The purpose of the distribution is twofold. First, distributing sensing, processing, and communication capabilities obtains a higher degree of reliable coverage and survivability. Second, the function of sensing may best be served by distributing the sensing power. Indeed, since a sensor's capabilities may depend upon its relative location with respect to the target, distributing the sensing capabilities augments the area covered by any single sensor. This is particularly true of target scenarios in which the targets try to avoid detection (e.g., low-flying aircraft, silent submarines). In addition, using a variety of sensors can provide an effective corroboration mechanism between different observations.

The distribution of the observation process implies a distribution of the processing and a suitable communication mechanism between the different parts, posing difficult architectural problems. How should we organize the network? How should we allocate functions, responsibilities and resources? How should we establish useful communication mechanisms? We call these organizational issues problems of decentralized architecture.

We shall reserve the word distributed to denote spatial distribution, while decentralized will denote a mode of organization and operation where no single process possesses the power and knowledge to monitor, control, and coordinate the activities of all other processes.

The two major problems of the DSN project are

1. To develop integrated inference systems to extract target-features from distributed observations.

2. To develop decentralized architecture and algorithms to support the inference system.

In what follows we shall consider some of the issues arising from the two problems. However, before proceeding to a detailed discussion of the issues, let us comment on the significance of the DSN development effort to other related technologies.

1.4 DSN, what contributions can be expected?

It is premature to try to answer this question accurately. Nevertheless it is important to keep in mind some possible scenarios of military information processing in the next decade. Therefore, rather than attempting to answer the question, we will present some possible technological problems to which the DSN research effort may be highly relevant.

In the mid-1980s military effectiveness will depend upon rapid collection, interpretation, evaluation, and dissemination of information. By that time a major portion of all strategic and tactical information will be computer-processable, and automated information processing will replace traditional manual methods. Achieving an integrated information system encompassing the full spectrum of command, control, and communication (C^3) functions will require a major architecture development effort in order to integrate the corresponding C^3 technologies.

The Distributed Sensor Networks (DSN) research will contribute to the development of integrated C^3 information systems in two forms: first, smart sensor networks will form an essential part of future integrated military information systems, and second, DSN offer a microcosm of the future operation of integrated C^3 systems.

Problems of developing real-time, function-oriented, high-level protocols, piecing together partial information to extract intelligence and feeding this intelligence back to alleviate computational processes, designing software for decentralized large-scale processing, and integrating communication and processing to support each other are not unique to DSN, but are an essential part of C^3 systems architecture. It is in this sense that the DSN project transcends its immediate goals and can be expected to contribute to the development of concepts, methodologies, and solutions to general problems of C^3 systems architecture.

2. DSN FUNCTIONS.

The first issue to be addressed is: What specifically do we expect DSN to do for us? The DSN goal is to extract information about targets from observables detected by a set of sensors and then to report it. This goal needs to be defined more accurately.

The description of the goal of a DSN must consist of two parts. First, it is necessary to describe the environment in which the DSN will operate. What type of targets are expected? What are the observables produced by those targets and the relationships between them? What do the sensors observe? What are the characteristics of the geographical environment in which the DSN is to operate? and so on.

Second, it is required to describe the type of information that we wish the DSN to deliver and the mode of reporting. What are the features that we wish to be reported by the DSN? Where should the reports be delivered? When should features be reported? How should an interactive mode of reporting be established (i.e., to permit human operators to request specific types of reports, or to focus the attention of the DSN on features of particular interest)?

In addition, a performance-evaluation scheme should be established to measure the merit of the DSN. What is it precisely that we wish to optimize in the "quality" of reports? Is it possible to direct the DSN through an interactive process to adapt to different external performance objectives (say, as defined by a human operator)?

Work should be carried out to abstract and define precisely the boundary of interaction between DSN and human operators, to establish agreed linguistic primitives for proper interaction, to define the possible performance objectives of DSN, and so on.

Once the expected functions of DSN are well understood, we can proceed to develop the system to provide those functions. The problems that we face consist (as noted in the previous section) of two classes: problems of drawing inferences from distributed observations and problems of organizing decentralized architecture. In the following sections we consider the two classes of problems more carefully.

3. ARCHITECTURE TO SUPPORT INFERENCES FROM DISTRIBUTED OBSERVATIONS.

In this section we shall restrict our attention to the "DS" part of DSN: Namely, what are the issues in developing intelligent processing of distributed observations from a variety of sensors? The constraints of decentralized processing of observations will be considered in the next section.

DSN are large integrated signal-processing systems whose main differences from classical sensor systems, as far as signal-processing is concerned, are the level of intelligence that we expect to extract from signals and the intelligence that is required to produce this information. Intelligence is the answer to the challenge of tracking targets that can not be fully observed. That is, in the absence of full observation of the targets, we have to use intelligence to piece together the little information that we may have. What are the important issues to be addressed in developing architecture to support intelligent inference-derivation systems?

To answer the question posed above we should digress on the problem of inferring from observations. Perhaps the best way to proceed is by considering a detailed example. We shall consider a DSN using acoustic sensors in order to ground the issues in a specific context. However, the issues raised are by no means restricted to the example and can be easily generalized to other DSN scenarios.

3.1 Spatially distributed observations.

3.1.1 Spatially distributed observables.

Consider the case of an aircraft emitting acoustic noise. The level of noise and its spectral composition depend upon intrinsic properties of the sources of emission (e.g., engine type and rpms). The noise emitted is modified by the environment of the source (e.g., atmospheric conditions, geomorphology) to induce a field of acoustic noise over a certain region. This acoustic field is a typical spatially distributed observable.

3.1.2 Observations.

The acoustic field may be sampled by a microphone. The sampled wave provides local information about the observable. The local sample may be statistically analyzed to extract some target features. Specifically,

1. It is possible to test the hypothesis that the observed sample has been produced by a target against the hypothesis that it is an environmental noise.

2. It is possible to estimate the speed of a moving target from the Doppler shift of frequencies.

3. It is possible to classify the target from its spectral signature.

4. It is possible to estimate the azimuth to the target using a directional microphone.

The possibility and accuracy of the above inferences depend upon two factors: the quality of observations and the sophistication of the inferring processes.

For instance, testing whether a target exists against background noises depends upon the level of background noise (e.g., strong winds may sound like targets). However, a sophisticated process could learn about characteristics of the environmental noise from past observations and use this intelligence to obtain more accurate resolution of hypothesis. For instance, it may be possible to learn about the spectral characteristics of the wind and identify new spectral components belonging to other sources, as the sources appear. How should inference algorithms learn from past observations? How could the information learned be used?

3.1.3 Inferring from spatially distributed observations.

In the previous section we considered point observations of acoustic emissions. The main problem to be approached in this section is: What can be inferred from a discrete set of spatially distributed samples?

First, it may be possible to employ spatially distributed observations to improve the signal-to-noise ratio (SNR). In other words, by applying averaging processes to the set of point observations, it is possible to reduce uncorrelated noises (just another application of the laws of large numbers). The noises to be reduced are those produced by the environment and intrinsic sensor noises. The internal noises of different sensors can be kept uncorrelated by using standard methods. The environment is more difficult to handle. If the sensors are far enough apart, the environmental noises are uncorrelated. However, if the sensors are close to each other, the environmental noise may be highly correlated and an averaging process will have an adverse effect on noise level (e.g., the noise of wind in a tightly packed array of microphones is a typical example). It is possible to use better filtering processes to reduce correlated noise. However, this requires that we have information about the nature of the noise, possibly also requiring a filter that learns and adapts to the noise. In addition, a sophisticated noise filtering may be computationally expensive.

Second, it is possible to use the distributed observations to infer spatial properties of the noise source. For instance, by correlating spatially distributed observations of an acoustic wave, it is possible to infer the direction of propagation and thus the azimuth to the target.

Such a capability requires that the processing algorithm be aware of the time and location of the observations to within a high accuracy. The problem of coordinating the time-space dimensions of the observations vis-a-vis the time-space dimensions of the observables is a key organizational problem. The overriding factor in obtaining a suitable space-time synchronization is the speed of communication. The correlation of some observations requires that a large volume of information be accurately communicated at a high speed to meet the requirement of real-time processing. For instance, in an array of microphones, the extraction of azimuth to the source requires that the sampled observations of the acoustic wave be correlated. This process requires a wide communication bandwidth between each observer and the processing algorithm. On the other hand, when the information to be processed consists of a small number of azimuth measurements (from a few arrays), the communication bandwidth demand is small. What are the suitable organizational structures for DSN to meet space-time coordination required by the inferring processes?

3.2 Architecture to support real-time distributed inferencing.

3.2.1 Hierarchical extraction of information--the "two-way pipeline".

As described in the introduction, the Bayesian paradigm of statistical derivation of inferences consists of improving the state of knowledge through iterative feedback. This paradigm implies a "two-way pipeline" inference architecture. Observations are first processed locally to extract some target-features. The information derived may be delivered to a higher-level processing algorithm that combines the pieces of information to extract higher-level associations to be delivered to higher processing levels, and so on. The upstream pipeline is an information compressor turning billions of observational bits into a small report to be delivered to the user.

The downstream pipeline returns information about the present state of target information, to be used as a prior knowledge to support lower-level processing. (Note that this two-way pipeline can map user's instructions and other external high-level information downstream.) In addition, higher levels can identify electronic and other enemy counter measures and instruct lower levels to respond accordingly.

It may be possible to carry out a resource management policy which adapts to the information available on a target scenario. For instance, it is possible to focus the attention of directional sources to those directions from which targets are expected to arrive. It is also possible to turn radars on and off when targets appear or disappear or when they may be endangered by smart bombs.

How should we structure the two-way pipeline? Which inferences are to be processed at which level? How should we use the inferences to achieve intelligent resource management policy?

3.2.2 Inferring in "real-time".

A major consideration in structuring a hierarchical two-way inference pipeline is the space-time structure of the observations. The spatial structure of the communication and processing resources is an overriding factor in deciding the allocation of functions and responsibilities among

network nodes. We shall consider the resulting problems in section 4. However, the requirement that processing should occur in real-time has implications beyond questions of resource management.

Different target-features have different temporal behavior--some change rapidly, others remain invariant. The notion of "real-time" changes from one feature to another. Accordingly, the knowledge-bases and the inferring processes at each level in the hierarchy should reflect the temporal nature of the knowledge.

Another aspect of the "real-time" processing requirement is that old data acquired must often be discarded to permit new data to be processed. In deciding whether to discard data and which data to discard, the inference process must evaluate the relative significance of the data. The value of information about a target-feature is a function of both its time and its content.

How should the inference processes manage their knowledge-bases to reflect the temporal nature of the knowledge and to evaluate the significance of data so as to manage local processing resources effectively?

3.2.3 Communication between inferring processes.

In this section we discuss semantic aspects of the communication between inferring processes. Once responsibilities are allocated to the different levels of the two-way inference pipeline, it is necessary to provide for an efficient mechanism to communicate knowledge between the different levels. The main questions are: What to communicate? and How to communicate?

To answer the first question we should first identify the objective of communication. Loosely speaking, the objective of communication is twofold, to update the knowledge possessed by the respective processes and to direct their attention. Accordingly, it is required to identify which knowledge may be required by which processes. How should that knowledge be represented? How should one inference process support the operation of others? Finally, how should higher level inferring processes dynamically focus the attention of the lower levels?

Another class of problems, arising from the distributed nature of the inference processes, is the problem of data-association. How do processes that reason about a multitarget scenario relate the (possibly different) images of the same scenario that they may have? What global view may be developed and how may it be developed?

These are the major issues as far as the semantics of communicating inference processes is concerned. In section 4.1 we shall consider the more general problem of communication in a decentralized system.

4. ARCHITECTURE TO SUPPORT DECENTRALIZATION.

In addition to issues arising from the need to develop integrated intelligent signal-processing architecture, it is necessary to understand and address issues arising from the decentralized nature of the participating processes. A line of distinction should be drawn between the "classical" notion of "distributed" architecture and our needs. Our main concern is to teach networks how to solve problems (e.g., as described in the previous sections) together. We shall call this "decentralized" computation as opposed to "P & V" problems of classical distributed processing.

Problems of decentralized cooperation are beyond the scope of present day practices of network-protocol development. Our problem starts at the point where present network protocols terminate; that is, once the issues of how to provide the best communication schemes are resolved, how is the communication facility to be used to obtain optimal cooperative problem solving?

Our main concern is to develop protocols for real-time decentralized cooperative problem solving. However, since the issues involved in developing real-time communication protocols are far from being resolved, a major effort is required to bridge the gap between existing approaches to high-level protocols and the communication needs of DSN.

4.1 It is necessary to introduce new types of communication protocols.

The function of communication protocols is to use the communication resources effectively to provide a suitable communication medium for application processes. The application processes in the DSN environment are distinguished by two important characteristics: first, they have to process real-time data in real-time and, second, they require close cooperation in order to proceed with their individual computations. These two characteristics dominate the arena of DSN protocol development.

4.1.1 Function-oriented high-level application protocols.

The main problem is that we have no methodology to design and implement "network-computations". How should computational tasks be allocated? How should the communication mechanism be used to advance the computations? How should we express (linguistically speaking) communicating algorithms? How should we model and design communicating computational processes? How should the protocol carry out an efficient resource management?

4.1.2 Problems of real-time protocols.

As discussed in section 3.2.2, the characteristic feature of real-time algorithms is that the data they process carries a virtual time stamp; it loses

its value as time progresses. The objective of a real-time protocol is not just to deliver data but to deliver as much data-value as possible (here the value has to do with both, the significance of the data for the processes and the time at which the data was produced). This objective implies that such classical protocol objectives as 100% reliable delivery are irrelevant. There is no point in insisting on delivery of outdated messages and tying up communication resources to serve them at the expense of larger delay for recent data. Therefore, a real-time protocol may discard outdated messages and/or give priority service to messages according to their "value".

Such a protocol is distinguished from classical protocols not only in its objectives but also in its possible requirements for implementation. For instance, a classical principle of protocol design is the principle of "transparency". Namely, service of messages by low-level protocols should be independent of their contents. A real-time protocol, on the other hand, may be required to provide service whose quality depends upon the value of the data content of the message. This in turn implies that high-level protocols need to be able to pass content-dependent service instructions to low-level protocols all the way down.

In addition, it is required that the application processes can instruct the communication protocol about how value should be assigned to data. That is, the criteria for comparing the value of different messages are beyond the scope of the protocol.

Therefore, the problems faced by real-time protocol developers are substantially different from existing practices. Many new concepts and methods will have to be introduced to structure a good real-time protocol.

4.2 Organizational problems of decentralized architecture.

Little is known about efficient organizational structures to perform tasks. The problem has been qualitatively studied by economists and recently by control-theoreticians. However, the problem is far from being understood in spite of its significance to many fields of knowledge. The economists have been mainly interested in descriptive models that, due to the complexity of human-organizations, are usually very crude. Control theoreticians, on the other hand, have restricted their interest to systems possessing a sufficiently simple mathematical description and even for those systems the problems of decentralization have not been adequately addressed.

With the advent of computer-communication networks, the problem of "programming networks" and organizing them to perform real-time computations will become significant. Being the first "special-purpose" computer networks for real-time distributed processing, DSN can be expected to contribute significantly to the understanding of organizational problems of large-scale decentralized systems.

4.2.1 Decentralized computations.

How should we organize a system of "myopic" distributed processes to perform a cooperative global problem solving? How should the processes reach agreements? Should control flow between processes? How should they coordinate local decisions to optimize global objectives? How should control flow between processes? How should we secure global convergence of local iterations to a desirable state?

4.2.2 Organizational structures.

How should we establish hierarchies of computational responsibilities? How should we tailor the organizational structure to suit the nature of the task to be performed and the availability of resources? How should processing and communication resources be allocated to support effective decentralized processing? What are the performance parameters and the tradeoffs (e.g., between communication and processing)? How can a large-scale system adapt to a global state to achieve global objectives on the basis of local observations only?

Problems of decentralized processing will demand a major interdisciplinary research effort. The methodologies developed to address the specific problems of DSN can be expected to bear major significance in the general understanding and development of function-oriented organizations for large-scale computational systems.

5. PROBLEMS OF MODELLING AND DESIGN

Here we have a large number of standard problems, some of which may have a specific DSN flavor.

1. Establishing DSN to maximize coverage.

   Topological optimization of sensor locations and distribution of sensor power. Choice of sensor power to optimize coverage. Alternative sensors' configurations and mixtures.

2. Performance and design parameters.

   Identify critical performance and design parameters and their relations. Performance optimization. Problems of reliable operation and survivability.

6. CONCLUSIONS.

Are open for discussion.

7. ANNOTATED BIBLIOGRAPHY.

The bibliography reflects essentially my personal taste. There has been no attempt to produce an extensive list of references. My choice has been

directed mainly by the need to expose different technologies and methodologies. The answers to DSN problems cannot be found in these references, although some fundamental relevant ideas may.

7.1 Signal processing technology

VAN-TREES H. L.

Detection, Estimation, and Modulation Theory, Part I, J. Wiley & Sons, Inc., 1968.

The "Bible" of signal estimation theory. Chapters 2, 3, parts of 4 and 5, contain an excellent introduction to problems of filtering and estimation. The second half of part III contains an extensive discussion of problems of parameter estimation from radar returns.

CAPON J.

"High-Resolution Frequency-Wavenumber Spectrum Analysis", Proc. IEEE, Vol. 57, NO. 8, August 1969.

LACOSS R. T.

"Review of Some Technique for Array Processing", in Exploitation of Seismograph Networks, Ed. Beauchamp, NATO Advanced Study Institutes Series, Series E: Applied Sciences - No. 11, Nordhoff - Leiden - 1975.

Both papers discuss the high-resolution technique for analyzing the map of the power-spectrum function, over the wave-number plane. The technique is the one used to extract source azimuth by cross-correlating the observations of the acoustic (or seismic) field as measured by an acoustic array. The first paper is highly theoretical, the second is more practical and is the best starting point.

BAR-SHALOM Y.

"Tracking Methods in a Multitarget Environment", IEEE Transactions on Automatic Control, Vol. AC-23, No. 4, August 1978.

A good survey of the problem with good references. The problem of Multitarget Detection requires intelligent detection algorithms that resolve the problem of data-association (i.e., associating data from sequential observations). This last problem is described and analyzed in

SITTLER R. W.

"An Optimal Data Association Problem in Surveillance Theory", IEEE Trans. Mil. Electron., Vol. MIL-8, pp. 125-139, April 1964.

7.2 AI software for inference drawing

ERMAN L. & LESSER V.

HEARSAY-II: Tutorial, Introduction & Retrospective View, Department of Computer Science, CMU, Report No. CMU-CS-78-117, May, 1978.

Describes Hearsay-II, a speech understanding system. Of particular interest to the DSN project is the Blackboard Architecture: A communication mechanism between the inference processes (each considered an "expert" in performing certain inferences), displaying the present state of knowledge.

SMITH R. G. & DAVIES R.

Distributed Problem Solving: The Contract Net Approach, Computer Science Department, Stanford University, Report No. STAN-CS-78-647, September 1978.

An approach to distributed problem solving is described whereby processes compete for the services of each other by bidding. The bids are used by the serving mechanisms to establish priority service.

7.3 Decentralized cooperative problem solving

7.3.1 The view of economists and decision theory

MARSCHAK & RADNER

Economic Theory of Teams, Cowles Foundation Monograph 22, Yale University Press, 1972.

The first part of the book is dedicated to an in depth study of the relation between statistical decision theory and rational decision making in organizations. The second part is a mind-provoking study of cooperative statistical decision making. Still unsurpassed in depth.

LUCE & RAIFFA

Games and Decisions, J. Wiley & Sons, Inc., 1967.

A critical introduction to game theory and statistical decision theory. There is an extensive introduction to problems of group decision making, culminating in chapter 14, where some of the major dichotomies are exposed.

7.3.2 The view of control theory

IEEE

Trans. Auto. Control, Special Issue on Large-Scale Systems and Decentralized Control, Vol. AC-23, No. 2, April 1978.

The survey article by Sandel etal. (pp. 108-128) is comprehensive and contains an extensive bibliography (Attention: read the conclusion of the survey). Other papers range between good to uninteresting. The paper by Hassan et al. (pp. 262) on decentralized computation of a Kalman filter bears some significance to the problem of position-tracking. The paper by Ho et al. (pp. 305) illuminates some of the major theoretical difficulties.

7.4 Computer Communication Networks.

McQUILLAN & CERF

"A Practical View of Computer Communications Protocols", Tutorial, IEEE Computer Society, 1978.

The only comprehensive introduction to problems of protocol design. Contains reproduction of many important papers on the subject. The paper by J. McQuillan "The Evolution of Message Processing Techniques in the ARPA network" is particularly recommended.

IEEE

Proceedings, Special Issue on Packet Communication Networks, Vol. 66, No. 11, November 1978.

A fresh collection of invited papers covering the full spectrum of packet-switched Computer Communication networks. The tutorial on protocols (pp. 1346) and the paper on high-level protocols (1371) are recommended as a prelude to DSN protocol development. The description of PRNets (pp. 1468) is essential to the understanding of DSN communication.

KAHN R. E.

"The Organization of Computer Resources into a Packet Radio Network", IEEE Trans. Communications, Vol. COM-25, No. 1, Jan. 1977.

A well organized exposition of the major issues in PRNets.

7.5 Resource management in computer systems.

KLEINROCK L.

Queueing Systems, Vol. II, J. Wiley & Sons, Inc., 1976.

The first volume can serve as a general introduction to queueing theory. The second volume is devoted to applications. Chapter 4 describes a menu of scheduling algorithms and analyses of their queueing performance. Chapters 5 and 6 contain an introduction, as yet unsurpassed, to resource management problems in computer communication networks.

SENSOR TUTORIAL

By
R. Lacoss
Massachusetts Institute of Technology
Lincoln Laboratory
Cambridge, Mass. 02142

SUMMARY

A general survey of sensor alternatives for detecting low flying aircraft and obtaining target metrics will be presented.

There are two distinct kinds of sensors. They are (1)-active, and (2)-passive. Passive sensors measure radiation produced directly by the target of interest. The major radiations of value are (1)-sound, (2)-infared from heat sources, and (3)-altimeter or radar radiations from the aircraft. The primary active sensor of interest is radar,of which there are very many alternatives.

Passive Sensors.

Following are some of the source characteristics which relate to passive sensors. First, all sources have very complicated radiation patterns. The spectrum and intensity of radiated energy will vary very much depending upon the orientation of the observing sensor with respect to the aircraft. That radiation pattern is also difficult to predict in detail with much certainty. As acoustic or infared sources aircraft are continuous radiators. The best a sensor can do is to extract a spectrum and estimate the direction from which the radiation is comming. Crosscorrelations between widely spaced sensors could in theory determine relative times of arrival of signals at the two sensors but radiation patterns and transmission path effects tend to make this impossible. The signals from the aircraft altimeter or radars are generally pulses and a sensor can determine both direction and time of arrival. Of course this source is available only if the aircraft is using the radiating equipment.

Microphones can be used to gather data from which spectra can be measured. The problem is to determine if the spectrum is from an aircraft or is just some other noise in the area. One help for this is to use a form of directional microphone or small array to measure the direction from which sounds are comming. Much more on the detection procedure and the characteristics of acoustic sources and sensors will be found in the papers by Landers and Demko in this workshop.

We also note that sound from a low flying aircraft couples into the ground below the aircraft to create seismic waves which can be measured by seismometers. In general the detection range for those seismic signals will be much less than the range for acoustic signals. However seismometers might be of value to identify non aircraft sources such as land vehicles.

Infared sensors measure the amount of radiation which hits them in the infared region. Several different bands can be used to obtain a simple signature of the radiation. Such sensors are combined with optics to view a very small field of view. This field of view is scanned and a heat picture of the region near the sensor is obtained. Aircraft should be local hot spots in that picture. Thus passive infared can obtain a signature and direction. In general these can be quite complicated sensors and currect technology seems to require cryogenic cooling to achieve the sensitivity for aircraft at several kilometer ranges.

Both acoustics and infared must contend with transmission path effects. For infared the transmission path effects primarily involve attenuation and line of sight limitations. Attenuation is not a major issue for acoustics in the frequency band of interest. Line of sight is some limitation but not nearly so serious. The major acoustic path effects result from the velocity structure of the medium and are discussed in detail by Landers.

The major problem for both acoustics and infared is to detect and identify the target in the presence of all other acoustic or infared noise sources in the area. Given the ability to do that they both give spectral information and direction. However, due to the slow speed of propagation of sound, the measured acoustic direction is where the aircraft was and not where it is.

Active Sensors.

Radar of course operates by radiating a signal which interacts with the target, is reradiated from the target, and detected by the sensor. The radar cross section of a target is a measure of the reradiation characteristcs of the target. It will fluctuate wildly as a function of the orientation of aircraft with respect to both the incident energy and the detecting sensor. The corresponding characteristic for passive sensors is the non isotropic radiation pattern mentioned above.

The signal to noise ratio, and thus ability to detect, a target in free space depends upon the amount of radiated power, the fourth power of the range to the target and the target's cross section. However for low flying aircraft there are additional important considerations. In particular these are terrain masking, clutter, and multipathing. Terrain masking is obvious. Depending upon the topography and how well sited a radar is used the line of sight limit for very low targets may be only a few tens of kilometers or less. This mitigates in favor of relatively small radars. Two kinds of clutter are of interest. Unfortunately the ground and other stationary things attached to

it reradiate signals just like real targets. These signals are called ground clutter and must be dealt with. Other clutter results from moving objects. The most important of these are weather, trees, and birds. Finally, the signal reradiated from a low flying aircraft can skip on the gound to get to a receiver as well as arrive by the direct path. This is called multipathing.

The variety of radars which can be constructed is very great.

First there are options on the radar signal shape. Simple pulsed radars measure the time delay from transmission to the reception of the reradiated signal. Continuous wave (CW) radars measure only the frequency shift of the reradiated signal from moving targets. From this doppler measurement target velocity information can be obtained. Also, use of doppler is a prime way to operate in ground clutter with moving targets. Hybrid pulsed doppler radars can be used to obtain both range and doppler information. There are many other alternatives but these will not be discussed.

Second there is the choise between monostatic or multistatic. In a monostatic radar the transmitter and receiver are colocated. Thus range measurements are the range from the sensor to the target and velocity measurements, or velocity used to determine if a target is moving, are the radial velocity with respect to the sensor. In multistatic (or bistatic if there is only one receiver) the transmitter is at one position and the receivers are at a different position. This is not a common mode for radars. There can be difficult timing and coordination problems for multistatic operation.

Finally there is the issue of directionality. Transmission antennas and receiving antennas generally are quite directional. This tends to conserve power and is very helpful with respect to clutter. Traditionally the directionality is also used to obtain directional information. Unfortunately good directionality requires large antennas and is somewhat at odds with the DSN concept. Fortunately the use of multiple range measurements for target location is generally superior to what can be achieved from a single directional radar.

MULTISITE ACOUSTIC LOCATION

By

Robert Walton
Lincoln Laboratory
Massachusetts Institute of Technology
Cambridge, Mass. 02142

ABSTRACT

Two acoustic arrays 10 km apart are used to locate a sub-sonic aircraft in the horizontal plane. The arrays measure azimuth of sound from the aircraft. If the latter is flying very slowly, it is located at the intersection of straight lines drawn from the sensors in the direction of the azimuths measured at the time the plane is to be located. If the plane is flying faster, the straight lines become curved because of the finite speed of sound, and measurements made over intervals of time must be used instead of measurements made at a single time. But the general picture remains very much the same. The effects of azimuth errors are similar for slow and fast aircraft, though geometrical diagrams relating error magnitude to aircraft location distort as the aircraft velocity changes. Aircraft height, which causes no error for very slow aircraft, causes only small errors for fast aircraft flying below 1 km.

INTRODUCTION

Our goal is to show that an aircraft can be located in the horizontal plane using data from only two acoustic sensors. Each sensor is a microphone array whose outputs are of the form "sound is coming from azimuth (horizontal direction) ϕ at time t." The two-sensor method described below contains ideas basic to any DSN tracking low flying aircraft with acoustic array sensors.

LOCATING AN AIRCRAFT

Suppose we have two sensors, S_1 and S_2, located 10 km apart, and both at the same height, which is zero in our coordinate system. Suppose a plane is flying around somewhere, and the azimuth vs. time curves for the two sensors are as in Figure 1.

First, we want to find the plane at time t, assuming (with no particular reason) that the plane is at height z = 0. Consider just the S_1 data. The sound emitted by the plane at time t must arrive at S_1 at some time $t_1 > t$. Let ϕ_1 be the azimuth reading measured for time t_1. Then at time t the plane is distance

$$R_1 = (t_1 - t)\cdot(\text{speed of sound})$$

from S_1, or at horizontal distance

$$r_1 = \sqrt{R_1^2 - z^2}$$

from S_1. Given t_1, we have found ϕ_1 and r_1, which specify a point in the horizontal plane. The set of all pairs (r_1, ϕ_1) given by all $t_1 > t$ generates a curve of possible positions in the horizontal plane for the airplane at time t. We call this latter curve the possible position curve for S_1, t, and z=0. Figure 2 shows such curves for t = -30, -26, -22, -18, -14, and -10 seconds, z = 0, and both S_1 and S_2. Note that the possible position curve for a sensor originates at that sensor, which is the point on the curve corresponding to $t_1 = t$, unless the available measured azimuth curve begins after time t.

At time t the plane must be at the intersection of the S_1 and S_2 curves for given t and z, assuming that the plane is actually at height z. Thus for each t we get a plane location. Connecting these locations with straight lines gives the computed plane path displayed in Figure 2. The computed plane path in Figure 2 is just about equal to the actual plane path so the actual path is not displayed. The plane is actually flying a straight line at 0.9 Mach and height 0.1 km. The difference between the assumed height z = 0 and actual height 0.1 km results in negligible error.

The possible position curves for t = -22 and -18 have multiple intersections: the extra intersections are labelled G denoting ghost. A crude automatic algorithm has been used in all our examples to distinguish real position measurements from ghosts and trace the computed path.

AZIMUTH ERRORS

We now consider the effects of errors in azimuth measurement. Suppose each such measurement ϕ is replaced by a range of measurements $(\phi-e, \phi+e)$ where e is the maximum possible error in ϕ. Then each possible position curve is replaced by a region shaped like a highway. The boundaries of this region are just the possible position curve rotated by +e and -e degrees around the sensor. Figure 3 shows these possible position regions for t = -30, -18, and -6 with z = 0 and e = 5 degrees. Also shown are the computed path, plus the ghosts for t = -18.

For particular t and z the intersections of the S_1 and S_2 possible position regions determine an intersection region where the plane will be found. This region generally contains all plane locations and ghosts found in the previous section for given t and z. Basically, ghosts are the result of nearly tangentially intersecting possible position curves, and are associated with an elongated intersection region containing

the true postion plus ghosts. Noise in the azimuth readings can cause ghosts anywhere in the intersection region.

The intersection angle of the S_1 and S_2 possible position curves indicates whether the intersection region will be elongated. For mathematical reasons we consider this angle to have a range from 0 to 360 degrees, with 180 degrees being the usual angle for tangential intersection. Angles between 180 - 2e and 180 + 2e degrees will generally result in elongated intersection regions, whereas angles near 90 or 270 degrees will generally result in small rectangular intersection regions. The intersection angle can be computed as follows. From the point at which the plane is located draw three velocity vectors (see Figure 4):

$\bar{v}$ The plane's apparent horizontal velocity.

$\bar{v}_1$ Points at S_1 and has magnitude (speed of sound)/cos θ_1 where θ_1 is the elevation angle of the plane at the given point (depends on z).

$\bar{v}_2$ Similar to $\bar{v}_1$ but for S_2 and θ_2.

Then the direction of the S_1 possible position curve going through the point is $w_1 = v - v_1$ and of the S_2 curve is $w_2 = v - v_2$.

As an aside we indicate why the procedure just given works. At time t the plane is at the given position emitting sound received by S_1 at time t_1 and angle ϕ_1. At time t+dt the plane has moved by an incremental position $dt \cdot v$ emitting sound received by S_1 at time t_1+dt_1 and angle $\phi_1+d\phi_1$. Let w_1 be a vector directed along the possible position curve for S_1 such that moving an incremental position $dt \cdot w_1$ from the original position will also correspond to the same change $(dt_1, d\phi_1)$ in the azimuth curve for S_1. Consider the local rectilinear coordinate system i_r, i_ϕ, i_θ at the aircraft location associated with global spherical coordinates with origin S_1, where i_r is the unit vector directed from S_1 to the given position, i_ϕ is the unit vector in the direction of increasing ϕ, and i_θ is the unit vector in the direction of increasing θ. Then if r_1 is the horizontal distance between S_1 and the plane location,

$$d\phi_1 = dt\ (\bar{v} \cdot \bar{i}_\phi) / r_1$$

$$d\phi_1 = dt\ (\bar{w}_1 \cdot \bar{i}_\phi) / r_1$$

$$dt_1 = dt + dt\ (\bar{v} \cdot \bar{i}_r / \text{speed of sound})$$

$$dt_1 = dt\ (\bar{w}_1 \cdot \bar{i}_r / \text{speed of sound}).$$

These last equations imply

$$\bar{w}_1 \cdot \bar{i}_\phi = \bar{v} \cdot \bar{i}_\phi$$

$$\bar{w}_1 \cdot \bar{i}_r = \text{speed of sound} + \bar{v} \cdot \bar{i}_r.$$

The equations just given imply $\bar{w}_1 = \bar{v} - \bar{v}_1$ where $\bar{v}_1$ is chosen so that

$$\text{speed of sound} = -\bar{v}_1 \cdot \bar{i}_r$$

$$0 = \bar{v}_1 \cdot \bar{i}_\phi.$$

While $\bar{v}$ is the apparent horizontal velocity for the computed path, and not the plane's actual horizontal velocity, we will ignore the distinction here because the difference between these velocities will be small as long as the error in position due to misestimated height z is small. (If the position were accurate both these velocities would be the same, because they give the same rates of azimuth change for S_1 and S_2). Note, however, that an actual vertical component to the plane velocity is equivalent to a radial horizontal component, because it projects on i_r, but this equivalent horizontal component is different for each sensor, and cannot be expressed by changing the apparent horizontal velocity of the plane. So our calculations are not exact unless the plane actually has zero vertical velocity, or unless the assumed height z = 0.

For given z and $\bar{v}$, curves of constant intersection angle can be plotted in the horizontal plane. These constant intersection angle curves have sensor as end points, and are show in Figure 5 for z = 0 and the actual plane velocity v. We call the region in which the intersection angle lies between 180 - 2e and 180 + 2e degrees the region of linear ambiguity, because in that region plane position can only be localized in one dimension.

Comparison of Figures 3 and 5 suggests that the intersection angle is not a very precise measure of how well a plane is localized. The region of linear ambiguity is nonetheless a useful descriptive concept.

Figure 6 shows the computed paths of the four corners of the intersection regions for times including those of Figure 2. The intersection regions tend to lie between the outermost of these four paths. One clearly sees the changes in length of the intersection regions as intersection angle changes. Near the region of linear ambiguity, the corners of the intersection regions exchange positions relative to each other, with their computed paths crossing over each other.

ERRORS IN ASSUMED HEIGHT

Figure 7 displays the computed plane paths for three different assumed heights, z = 0, 1, and 2 km. Not much change is observed as z changes, except in the region of linear ambiguity.

Figure 8 is just Figure 2 but with assumed height z = 2 km instead of z = 0. Note that not only have the ghosts disappeared, but there is also no intersection at all of the possible position curves for t = -18 seconds. Errors in

azimuth can also eliminate intersections.

Figure 9 displays possible position curves for t = 0 and z = 0, 1, and 2. Changes in z only cause changes in the radial distance to the sensor, and therefore change the possible position curve significantly only when that curve is perpendicular to the radial direction from the sensor. Maximum perpendicularity is achieved by fast (nearly Mach 1.0) planes flying nearly toward the sensor. But if such a plane is between the two sensors, it will have a nearly radial possible position curve for the other sensor, and consequently the possible position curve for the other sensor will not change much with changing z. So changing z in Figure 9 moves the intersection point along one of the two intersecting possible position curves.

Changing z induces a radial translation in a possible position curve very similar to the radial translation induced by changing t. Thus in Figure 9, changing z moves the intersection point along the actual path. As a result, misestimating z does not misestimate the plane path, but only the times at which the plane is at various points on that path.

For planes flying between the sensors changing z generally results in a position error parallel to the plane path.

CONCLUSIONS

Time series azimuth measurements from just two sensors, plus an estimate of plane height, can be used to locate a plane within a box, the intersection region of two possible position regions. For approximately half of a typical plane path this intersection region is a nicely behaved rectangular box, but elsewhere the region is elongated and may become U shaped. The major exception to this rule occurs when the plane flies along the straight line through the sensors, passing very near both sensors.

For two sensors 10 km apart and planes flying no higher than 1 km, the effect of misestimating the plane height is usually minimal.

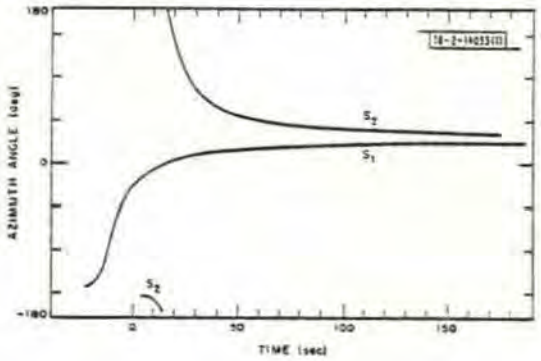


Fig. 1. Azimuth angles vs. time measured by sensors S_1 and S_2 for some aircraft to be located.

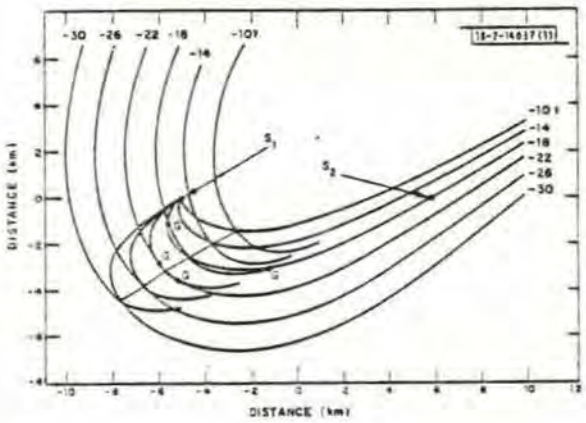


Fig. 2. Possible position for t from -30 to -10 sec. There is one curve for each sensor, S_1 and S_2, and intersections of these two curves are either points on computed plane path (nearly straight line) or ghosts (marked G). Assumed plane height is z = 0, actual height is 0.1 km, plane velocity is Mach 0.9.

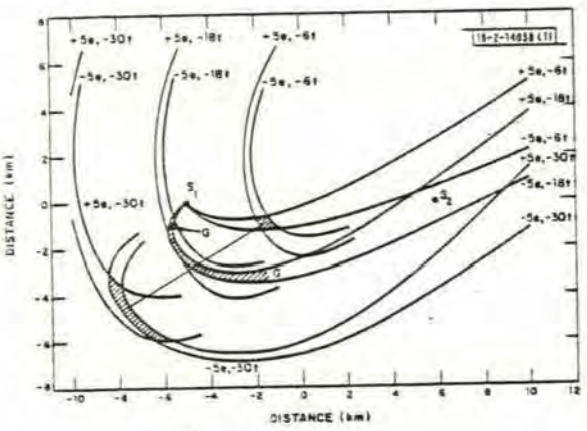


Fig. 3. Possible position regions and intersection regions for times -30, -18, and -6 sec. Computed plane path (straight line) and ghosts (marked G) for these times are also plotted. Possible position curves with assumed errors e of +5° and -5°. z = 0 as for Fig. 2.

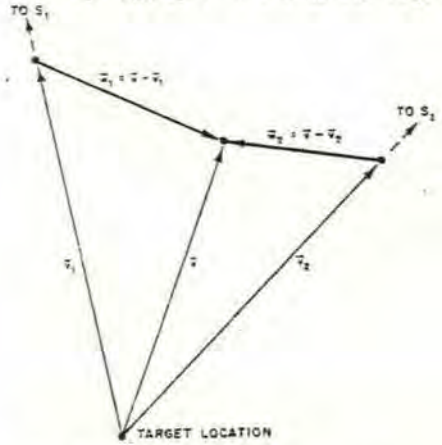


Fig. 4. $\bar{v}$ is plane's apparent horizontal velocity. $\bar{v}_1$ is directed towards S_1, $\bar{v}_2$ towards S_2. If assumed height z = 0, magnitude of $\bar{v}_1$ and $\bar{v}_2$ is speed of sound. $\bar{w}_1$ is directed along S_1 possible position curve, and $\bar{w}_2$ along S_2 possible position curve.

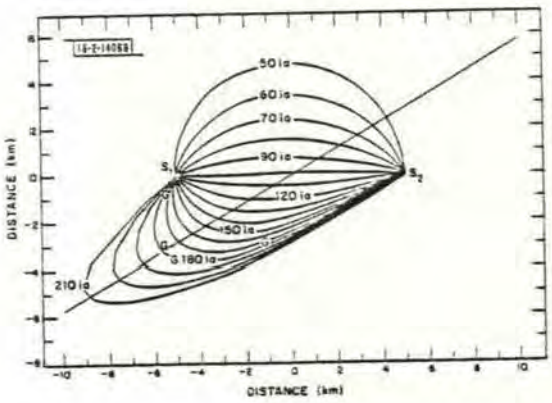


Fig. 5. Curves of constant intersection angle, for angles from 50° to 210° in 10° increments. Plane's actual velocity is used with assumed height z = 0. Computed plane path and ghosts are displayed. Region of linear ambiguity lies between 170° and 190° curves.

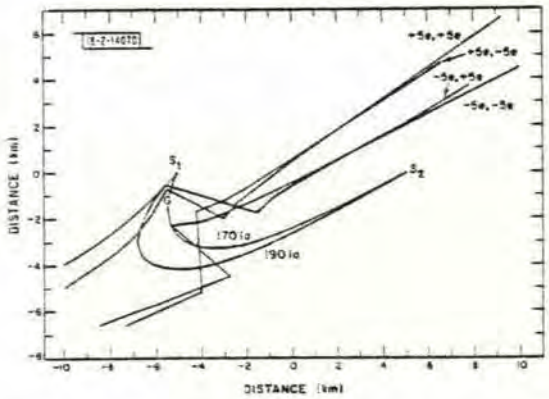


Fig. 6. Computed paths of corners of intersection regions for errors e of +5° and -5° in azimuth angle, assumed height z = 0, and times from -50 to +50 sec in 4-sec increments. Intersection regions generally lie between outermost paths. Region of linear ambiguity is also shown.

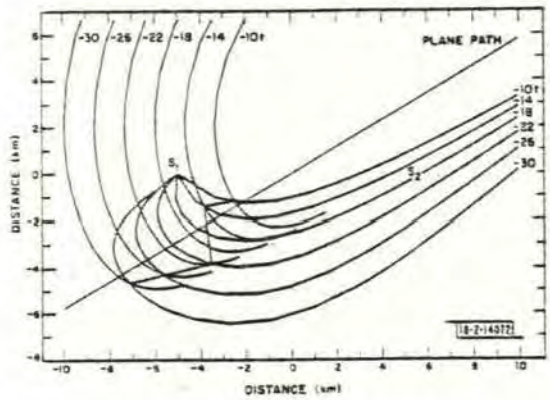


Fig. 8. Possible position curves for assumed height z = 2 km and times from -30 to -10 sec. Actual plane height is 0.1 km. Actual and computed plane paths are displayed. At t = -18 sec, possible position curves do NOT intersect at all.

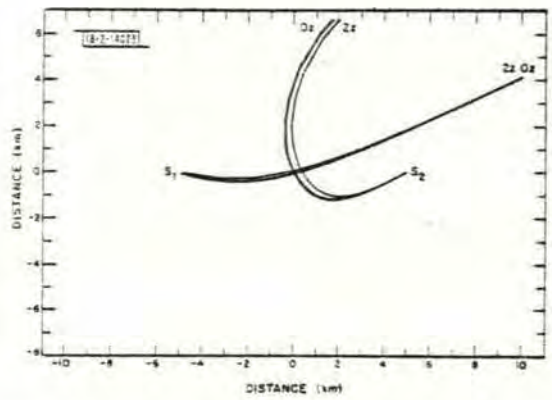


Fig. 9. Possible position curves for time t = 0 and assumed heights z = 0, 1, and 2 km.

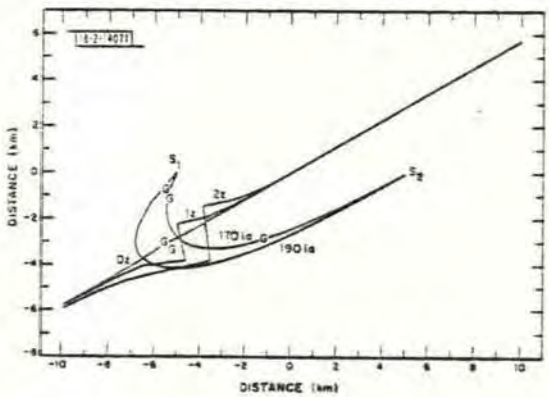


Fig. 7. Computed plane paths for assumed heights z = 0, 1, and 2. Plane location is determined at times from −50 to -50 sec in 4-sec increments. Region of linear ambiguity is also displayed.

LONG RANGE ACOUSTIC TRACKING OF LOW FLYING AIRCRAFT

T. E. Landers and R. T. Lacoss
Lincoln Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02142

Abstract

Data from a 5 meter array of microphones has been collected and analyzed in the band 20-200 Hz to find the accuracy to which the azimuth of low-flying subsonic aircraft can be tracked and to measure signal levels as a function of range. Flight paths were radar tracked and recorded on digital tape. Propagation characteristics were calculated from measured meteorological data and were used to help interpret observed signal level variations. The aircraft included helicopter, prop-driven and jet aircraft which flew 30 km reversed linear tracks at altitudes of 70 m, 300 m and 1 km. Azimuth was measured by applying maximum-likelihood frequency-wavenumber analysis to the digitized data from the fractional wavelength array of microphones. Measured azimuths, after the appropriate moving source corrections, were found to be accurate to within a few degrees at ranges well beyond 5 km. Filtering and array gain allowed us to obtain good azimuth estimates for aircraft signals at least 10 dB below the background noise level.

Introduction

During June 1977, an experiment was conducted at Fort Huachuca, Arizona, for the purpose of collecting acoustic and seismic array data for low-flying aircraft travelling at subsonic velocities. This experiment differed from all previous work known to us in that the exact position of the aircraft as a function of time was obtained by recording radar tracking information on digital tape.

The radar tracking and recording service was provided by the United States Army Electronic Proving Ground (USAEPG), Fort Huachuca, Arizona. In addition, USAEPG arranged for aircraft, communication facilities, power, balloon flights to gather meteorological data, surveillance radar, and general physical support. The acoustic and seismic measurement and recording equipment and the necessary personnel to set up and operate it were provided by the United States Army Corps of Engineers Waterways Experiment Station (WES). WES also provided the analog-to-digital data conversion services at their facility in Vicksburg, Mississippi.

In addition to source location requirements, the experiment was designed to include as complete a set of the various aircraft source types as possible and known atmospheric acoustic transmission characteristics. There are essentially four different aircraft acoustic source types, and one of each class was employed in the experiment. The aircraft were: an A-7 jet, a Mohawk OV-1D twin-engined turboprop, a Beechcraft U-8 twin reciprocating-type prop, and a UH-1 helicopter. To adequately cover the possibility of atmospheric propagation problems, the experiment was designed such that each aircraft flew reversed tracks 30 km in length at altitudes of 70 m, 300 m, and 1 km above the recording site, and a circular track of radius 4 to 5 km about the recording site. The tracks were repeated at various times when predicted meteorological conditions might have caused significant differences in acoustic propagation characteristics. To firmly establish acoustic propagation characteristics at those times, two meteorological balloon flights were conducted at the expected extreme acoustic conditions, early morning and mid-afternoon. The aircraft flight paths were radar tracked and controlled and recorded on digital tape. Synchronized acoustic data were analog recorded at the sensor site.

The acoustic-seismic sensors and recording van setup is shown in Fig. 1, and the array setup is shown in Fig. 2. The array was designed to provide unaliased yet complete and resolvable wavenumber coverage appropriate to the frequency range 5 to 300 Hz for phase velocities of the speed of sound and greater. Referring to Fig. 2, the instruments beginning with GR were General Radio type 1560-P5 microphones, BK were B & K type 4165 microphones, DR were CBS type 6065 three-component gradient microphones, ERT was a General Radio type 1961-9601 electret microphone, and S were three-component Mark Products L-4 geophones. The seismometers were placed approximately 15 cm below ground level on cement piers. ERT was placed at 3 m above ground level, and all of the other microphones at 1 m above ground level. A GR microphone was placed approximately 1/2 km NW of the array to obtain large range signal coherence information.

During field operations, aircraft acoustic and seismic data were recorded on analog tape. An additional amount of time was spent recording ambient noise conditions during various parts of the day and night and conducting instrument calibrations. Intervals of the data representing each aircraft type were digitized at 5000 Hz. Figures 3 through 6 show examples of digitized data for each aircraft type on the various instruments. We note that, as expected, blade-rate harmonic-type signals dominate for prop-driven aircraft and the overall signal coherence is high for all aircraft types. We also observe good S/N ratios in the data. The OV-1D data were taken during bad surface wind conditions (gusting to 25 knots). However, the acoustic noise from the wind is well out of the signal band and the inband S/N is still quite high. We note that acoustic signal detection characteristics are not necessarily severely affected by high ambient noise fields if the data have sufficient dynamic range and signal and noise have different spectral or spacial characteristics. More detailed studies of detection range, classification, tracking, and location are addressed in the following sections.

Analysis and Evaluation of Experimental Data

Preliminary results based on a review and analysis of a part of the experimental data are summarized in the following sections. In order to interpret the results properly and to understand the potential use of acoustic sensors, one must be aware of certain significant moving-source and atmospheric-propagation effects. These well known effects as they apply to our particular situation are reviewed in the sections preceding the analysis and evaluation.

Moving-Source Effects

Consider that at time t a sound source, moving at constant velocity v, is at a distance vt from the closest point of approach (cpa) to an observation point O. Let the distance between O and cpa be r so that the distance of the source from O at time t is

$$D(t) = \sqrt{r^2 + v^2t^2} \quad .$$

The sound arriving at O came from the source when it was at some earlier position, say a distance of R from O. The time required for sound to travel R (i.e., R/c, where c equals the speed of sound) will equal the time the source took to move from R to D. Let that be a distance L. Since the source moves at v, then we have $L/v = R/c$ so that $L = MR$ where M, the Mach number, is defined at v/c. The geometry of this situation is shown in Fig. 7. From the figure we have

$$R^2 = r^2 + (MR - vt)^2$$

and so, for subsonic v, the acoustic range is given by

$$R(t) = [-Mvt + \sqrt{(1 - M^2)r^2 + v^2t^2}]/(1 - M^2) \quad .$$

Using this relationship, the following relationships are found[1]: the true azimuth,

$$\gamma(t) = \tan^{-1}(-vt/r),$$

the acoustic azimuth,

$$\phi(t) = \tan^{-1}[(MR - vt)/r],$$

the perceived frequency when the source is emitting a frequency f_o,

$$f(t) = f_o/[1 - M \sin(\phi)],$$

the acoustic range rate,

$$\dot{R}(t) = -v(f/f_o)\sin(\phi),$$

the acoustic azimuth rate,

$$\dot{\phi}(t) = -rv(f/f_o)/R^2$$

and the far-field pressure envelope

$$p(t) = (f/f_o)^2/4\pi R.$$

Figures 8 and 9 show the behavior of these functions of M = 0.75 and M = 0.12, respectively, which are the maximum and minimum values used in the experiment. We note the following properties of these functions which apply to a greater extent for larger Mach numbers and a lesser extent for smaller Mach numbers. As shown in the figures, the acoustic position is delayed relative to the actual position in proportion to the Mach numbers. At large equal distances, the pressure on the incoming path is greater than the pressure on the outbound path by a factor of $(1 + M)^2/(1 - M)^2$. For example, a source traveling half the speed of sound radiates four times the power forward as backward. Additionally, the peak pressure comes before the vehicle is at cpa. Finally, we note that large changes in azimuth, excess pressure, and doppler (f/f_o), take place primarily in the region when the vehicle is within about $\pm r/c$ of cpa.

Meteorological Effects

In support of the experiment, two balloons were launched to gather sound speed, wind velocity, and relative humidity data. Balloon flights were scheduled during the expected most extreme acoustic conditions. The purpose was to obtain an appropriate data base for determining the importance of sound refraction and attenuation over the observed detection range and signal variations. The first balloon was launched during the mid-afternoon (approximately 3 p.m. local time) when the normal lapse rate would have a high positive temperature gradient approaching the ground surface. During such conditions, the sound velocity profile is monotonically decreasing upwards causing sound energy to be continuously refracted upward at every level of the troposphere. Under such conditions, detection ranges should be less than average. Additionally, during this period small- and large-scale thermal convection is expected to have non-negligible acoustic effects by increasing the background acoustic noise level (wind noise) and introducing asymmetry in propagation characteristics. This phenomenon, due to wind velocity gradients will alter the effective sound velocity profile as a function of propagation direction. Figure 10 shows the meteorological data collected during this time span. The wind direction is not shown although that data was also obtained from the balloon tracking data. The sound speed is that calculated directly from the temperature data. Figure 11 shows the east-west geometrical propagation characteristics (based on ray tracing[2]) for sources at 200, 900, and 3000 ft above ground level. Only the critical distances are indicated and not the ray paths. The calculation predicts shadow zones at 2.1 km west and 1.8 km east for the source at 200 ft altitude; i.e., all rays that start out more horizontal or upward than the critical rays are turned upward before reaching the ground. Similar statements are true for the sources at the higher altitude with the shadow zone distances given in the figure. Ray theory also predicts that some energy in the field from the source at 900 ft altitude is

ducted in an easterly direction. Given changes in the meteorological conditions, or changes in topography, such energy might be detected at large ranges. It is noted that detection range (assumed proportional to the shadow-zone range) is directly proportional to the source altitude for these meteorological conditions.

The second balloon was launched during the early morning (approximately 3 a.m). At this time, the normal lapse rate should be underlain by a high negative temperature gradient approaching the ground surface. Under these conditions, the sound-speed profile has a zone of low values near the surface. Energy entering that zone will be refracted toward the ground, increasing the effective detection range. During periods when such velocity conditions persist, better than average detection ranges should be encountered. Upper level winds during this time period may also have severe propagation path effects. However, there is not much small-scale lower level turbulence at that time and consequently low background noise levels should be encountered. Figure 12 shows the balloon data collected during this early morning flight, and Fig. 13 shows the ray-tracing results based on these data. Just opposite to the mid-afternoon results, the lowest source has the largest detection range. Again, winds produce asymmetry and some ducting occurs, but overall much larger propagation distances can be expected during these conditions.

In summary of the theoretical ray-tracing results, we expected geometrical detection ranges of at least ±10 km for all the sources during good acoustic periods and of at least a few kilometers during unfavorable acoustic conditions. Ambient noise conditions due to surface winds, rain, etc., will of course make detection more difficult but should not significantly change the above effects for systems of sufficient dynamic range. In reality, larger detection ranges should be expected from energy that is diffracted and scattered into the shadow-zone region. Lastly, when wind gradients are on the order of normal sound-speed gradients, asymmetrical propagation conditions will exist.

Observed Detection Ranges

Direct analog plots of most of the acoustic data were obtained in the field. Unfortunately, the time base on these plots is known only to within a few tens of seconds, or equivalently the aircraft location is known only to within a few kilometers. Additionally, plotter noise starts at about 30 dB below full scale. The result is that when using the plotted analog data, estimated acoustic range may be in error by about a kilometer and maximum signal dynamic range will be about 35 dB. Figures 14 to 18 show the amplitudes of detected signals on these analog plots as functions of acoustic range. The acoustic range was derived from the radar tracking data and the moving source effect formulas given previously. The solid curved lines on the plots indicate normal geometrical spreading for a homogeneous atmosphere. The vertical lines show the geometrical shadow boundaries determined from the meteorological data. Finally, the peak noise level is given by the horizontal lines.

The data presented in the figures can be summarized as follows: in all cases, with only 35 dB of effective dynamic range, the aircraft were acoustically detectable out to at least 10 km, although not always on both the inbound and outbound paths. Amplitudes die off at about the geometrical rate, appropriate to a homogeneous atmosphere, up to the calculated geometrical shadow boundary for the actual atmospheric conditions. In the diffracted region, signal levels are diminished as expected, though still at detectable levels. These observations are consistent with the predictions based on the measured meteorological condtions. Finally, we note that even though the ambient or background noise level is often considerably greater than the signal level, the detection level is still high since the noise and signal energy are well separated in frequency.

Spectral Observations

Figures 19 through 25 show spectra computed at 1-sec intervals for each aircraft type during selected parts of their flight paths. Figures 19 to 23 show the UH-1 helicopter beginning at 4 km inbound, 4 km outbound, and circling at a 5-km distance, respectively. The velocity of the aircraft was about 40 m/sec, therefore each segment represents a change in position of 40 m. The basic blade-rate frequency for the UH-1 was 11 Hz. Figures 22 and 23 show similar spectra for the U-8, and Figs. 24 and 25 show spectra for the A-7 and OV-1D, respectively. Fundamental blade-rate frequency for the U-8 was 180 Hz and shaft rate 60 Hz. The OV-1D blade-rate frequency was 70 Hz. Each spectrum clearly shows that the signal is composed of a combination of fundamental and harmonic frequencies, with the exception of A-7 spectra which have the expected broadband appearance. We note that the fundamental (or lowest harmonic in the data band) is not necessarily the dominant peak and further, and more important, the overall spectral pattern can be highly dependent on the position of the source with respect to the receivers. In general, such results indicate that target classification (as to jet, turboprop, or helicopter) should be straightforward. However, tracking and multisite target identification will be more difficult to achieve. As to Fig. 25, we note the Doppler shift as the OV-1D passes cpa and the low-frequency (but out of the signal band) wind-noise interference during the later portions of the trace. Figure 26 shows the improvement gained with simple high-pass filtering.

Array Processing

Conventional and maximum likelihood high-resolution frequency wavenumber array analysis techniques were applied to digital data for the

array configuration shown in Fig. 27. The purpose was to establish the feasibility of making useful azimuth measurements using a small acoustic array. It is important to distinguish between accuracy and precision in acoustics since, while an acoustic wavefront crossing an array may be such that a very precise measurement of its direction and velocity can be made, the direction may not very accurately point to the source from which the sound originated. Inaccuracies of considerable magnitude may be present, due to crosswinds, topographic reflections, horizontal temperature gradients, and in general, the complicated acoustic nature of the troposphere. Figure 28 shows the impulse response of the array contoured in 1-dB intervals in wavenumber space. The circle labeled C is the wavenumber corresponding to the speed of sound, and any propagating sound fields crossing the array will have their power within this circle. Figures 29 and 30 show high-resolution and conventional wavenumber spectra for a UH-1 at 8 km acoustic range. A measure of precision is the width of the 1-dB contour which is a function primarily of the array configuration and chosen frequency. In this case, the width is about 12°. The accuracy is dead on. The conventional spectrum, although less precise, is also dead on in locating the true direction of the source. Figures 31, 32, and 33 show the results for, respectively, a U-8 at 5 km, an A-7 at 8 km, and an OV-1D at 1.1 km. In each case, the accuracy is well within the precision and on the order of at most a few degrees. Figure 34 shows the observed acoustic azimuth obtained from the acoustic array processing as a function of the acoustic range obtained from the radar data for a complete A-7 track. The known acoustic azimuth, again from the radar data, is shown by the solid line. While it is premature to judge that acoustic direction will generally be determinable to the accuracy found here, these results indicate that such is possible.

References

1. P. M. Morse and K. U. Ingard, Theoretical Acoustics (McGraw-Hill, New York, 1968).

2. R. J. Thompson, "Compiling Sound Ray Paths in the Presence of Wind," SC-RR-67-53, Sandia Laboratory, Albuquerque, New Mexico (1967).

3. J. Capon, Proc. IEEE 57, 1408 (1969).

The views and conclusions contained in this document are those of the contractor and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the United State Government.

This work was sponsored by the Advanced Research Projects Agency of the Department of Defense.

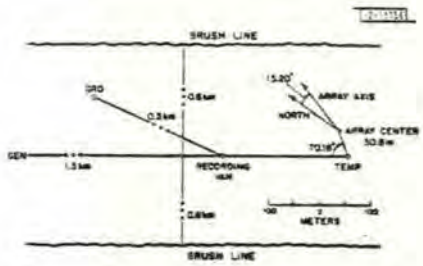


Fig. 1. Acoustic-seismic sensor and recording van setup.

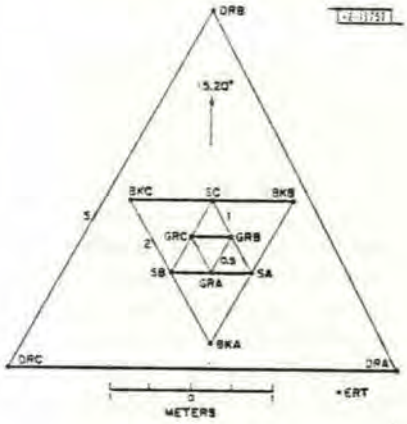


Fig. 2. Acoustic-seismic array setup.

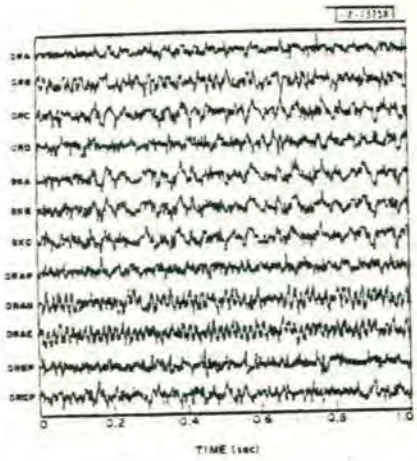


Fig. 3. Segment of U-8 acoustic digital data at an acoustic range of 5 km inbound.

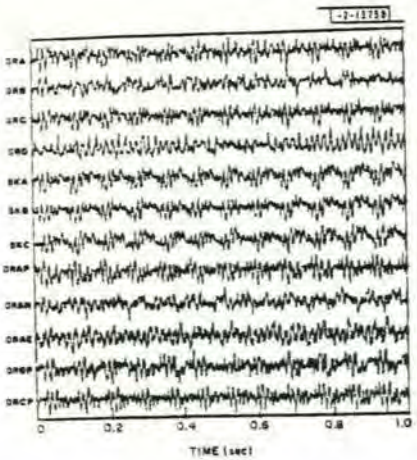


Fig. 4. Segment of UH-1 acoustic digital data at an acoustic range of 8 km inbound.

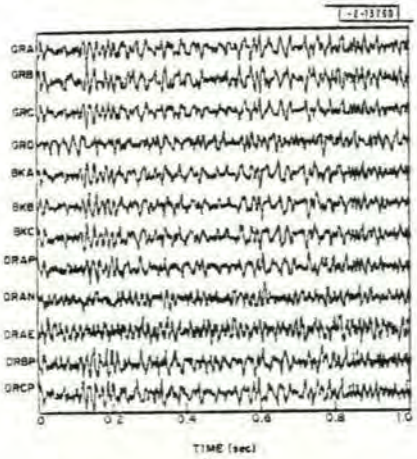


Fig. 5. Segment of A-7 acoustic digital data at

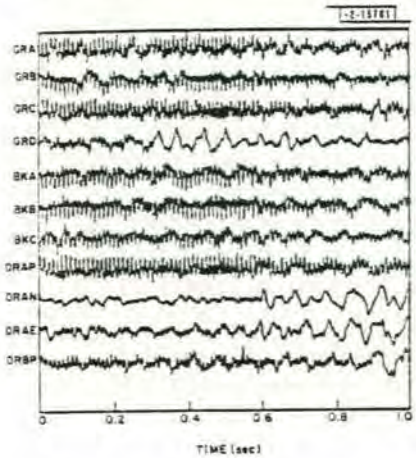


Fig. 6. Segment of OV-1D acoustic digital data at an acoustic range of 1.2 km inbound under severe surface wind conditions.

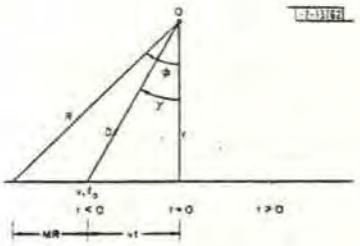


Fig. 7. Geometry for the analysis of kinematic parameters for a constant-velocity source.

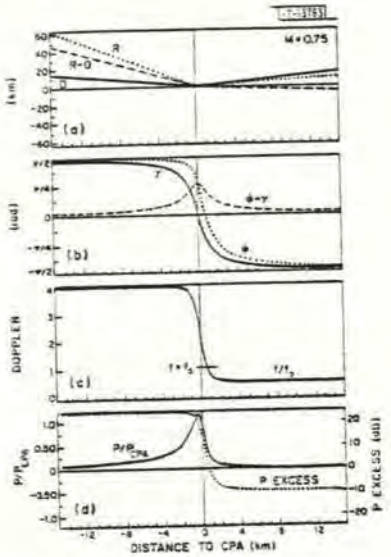


Fig. 8. Kinematic and dynamic characteristics of sound received from a Mach 0.75 flyby. The characteristics are shown as a function of the aircraft distance from cpa. Negative distance is for the incoming path. (a) Accoustic range R, actual range D, and their differences. (b) Acoustic azimuth ϕ, actual azimuth γ, and their difference. (c) Doppler f/f_o. (d) Ratio P/P_{cpa} of pressure to pressure at cpa and, P_{EXCESS}, the excess pressure above pressure that would be observed if the source were stationary.

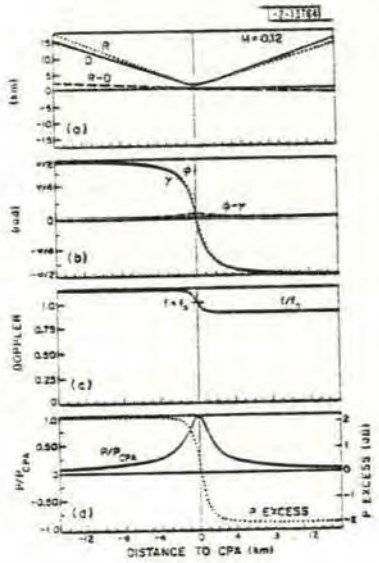


Fig. 9. Same as Fig. 8 but for Mach 0.12.

Fig. 10. Mid-afternoon meteorological balloon data.

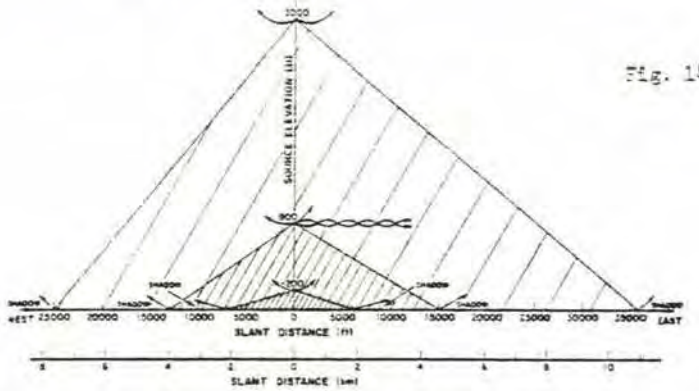


Fig. 11. Primary critical points and their geometrical shadow zones for afternoon conditions.

Fig. 12. Early morning meteorological balloon data.

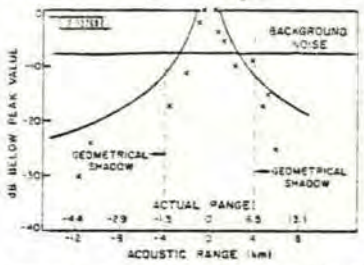


Fig. 14. Detection data for A-7 (altitude = 450 m, 0 dB = 87 dBa).

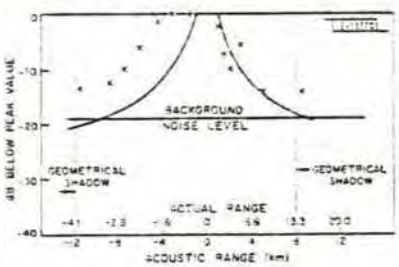


Fig. 13. Detection data for A-7 (altitude = 1020 m, 0 dB = 82 dBa).

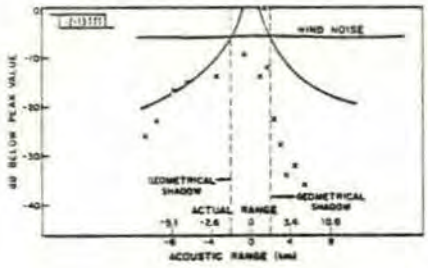


Fig. 16. Detection data for Ov-1D (altitude = 75 m, 0 dB = 88 dBa, windy conditions).

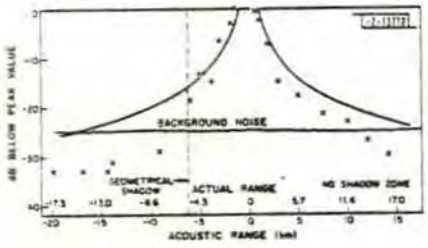


Fig. 17. Detection data for UH-1 (altitude = 300 m, 0 dB = 81 dBa).

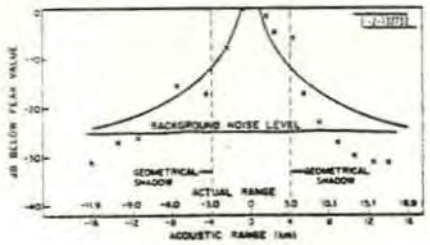


Fig. 18. Detection data for U-8 (altitude = 300 m, 0 dB = 68 dBa).

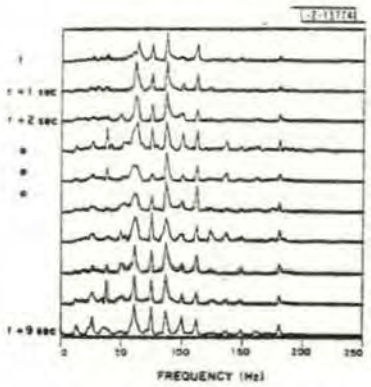


Fig. 19. Power spectra on a linear scale for a series of 1-sec (1 sec = 40 m position change) time intervals. Target is a UH-1 inbound with the first spectrum corresponding to a range of 4 km.

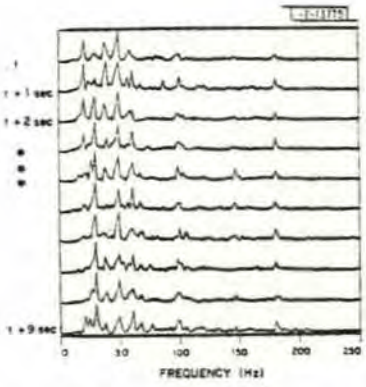


Fig. 20. Same as Fig. 19 for the UH-1 at 4 km outbound.

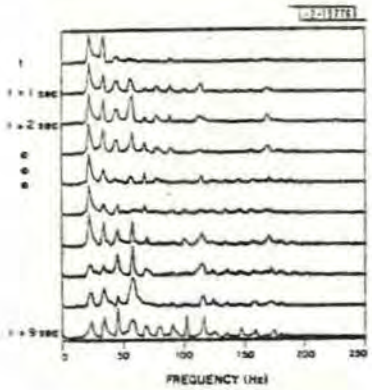


Fig. 21. Same as Fig. 19 for the Uh-1 circling at 5 km.

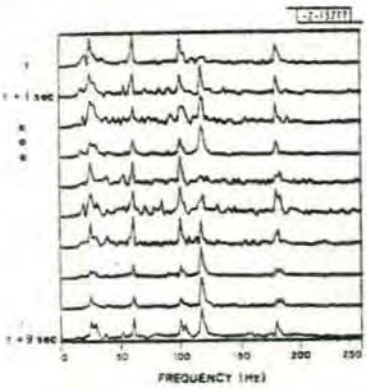


Fig. 22. Power spectra on a linear scale for a series of 1-sec (1 sec = 80 m position change) time intervals. Target is a U-8 inbound with the first spectrum corresponding to a range of 4 km.

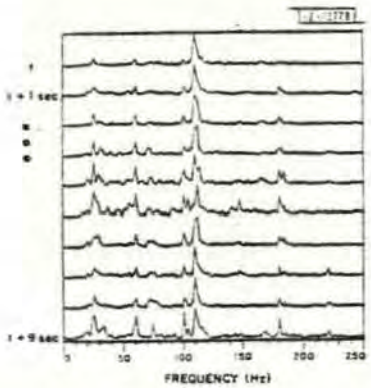


Fig. 23. Same as Fig. 22 for the U-8 at 4 km outbound.

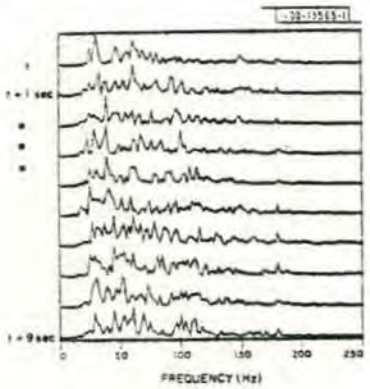


Fig. 24. Power spectra on a linear scale for a series of 1-sec (1 sec = 210 m position change) time intervals. Target is a A-7 outbound with the first spectrum corresponding to a range of 3 km.

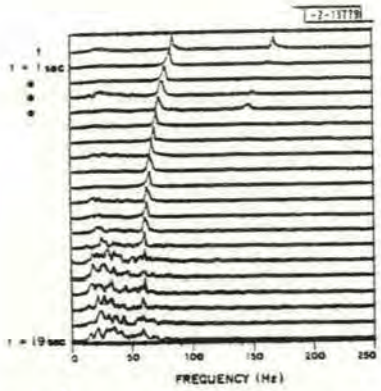


Fig. 25. Power spectra on a linear scale for a series of 1-sec (1 sec = 110 m position change) time intervals. Target is an OV-1D inbound at an initial range of 1 km.

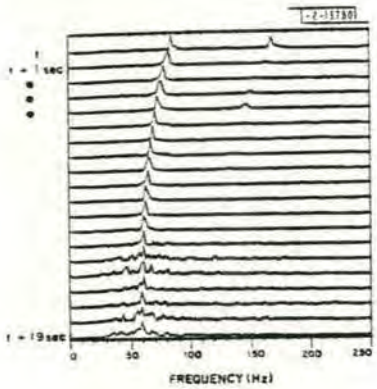


Fig. 26. Same as Fig. 25 but data preprocessed with a 50-Hz high-pass filter.

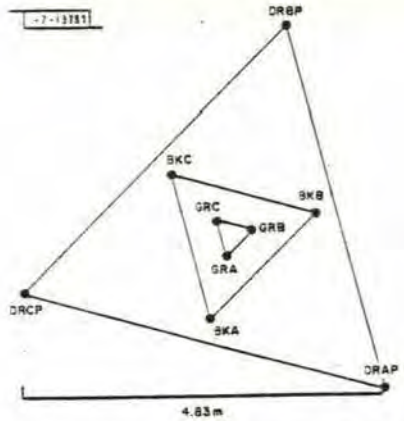


Fig. 27. Sensor stations used for array processing.

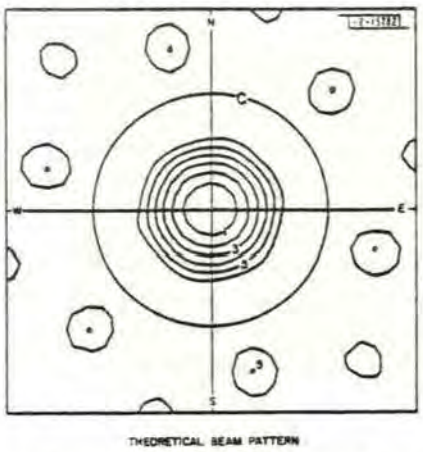


Fig. 28. Wavenumber impulse response of the array of Fig. 27.

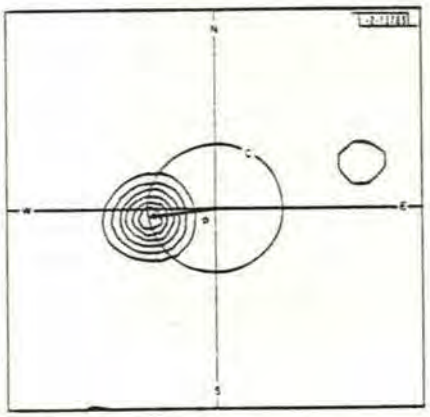


Fig. 29. High-resolution wavenumber spectrum for the UH-1 at 8 km. True azimuth is 0.

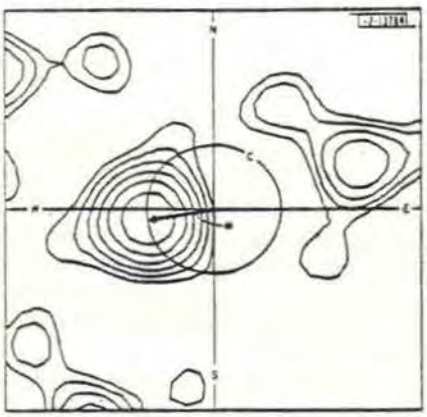


Fig. 30. Conventional wavenumber spectrum of UH-1 at 8 km. True azimuth is ϕ.

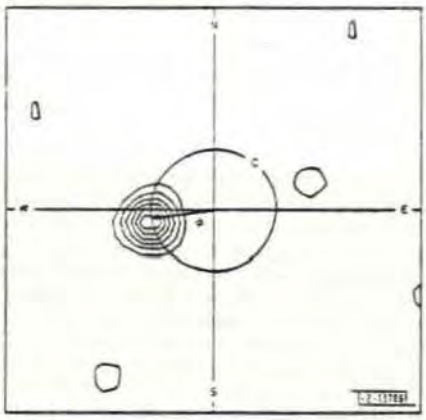


Fig. 32. High-resolution wavenumber spectrum for the A-7 at 8 km. True azimuth is ϕ.

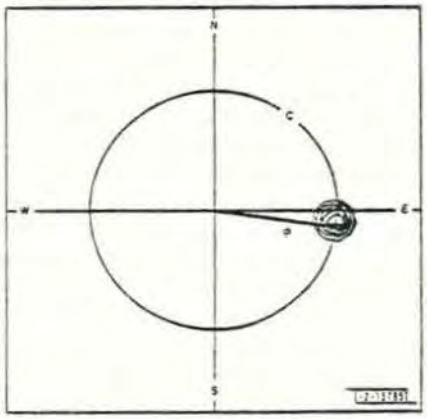


Fig. 31. High-resolution wavenumber spectrum for the U-8 at 5 km. True azimuth is ϕ.

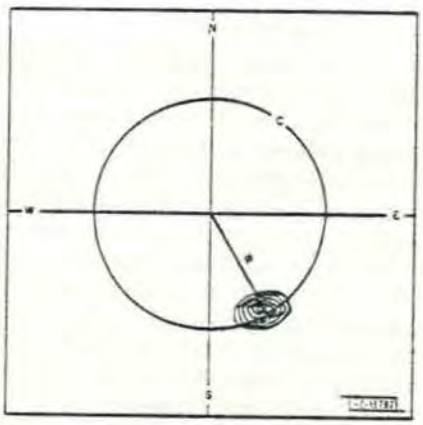


Fig. 33. High-resolution wavenumber spectrum for the OV-1D at 1.1 km. True azimuth is ϕ.

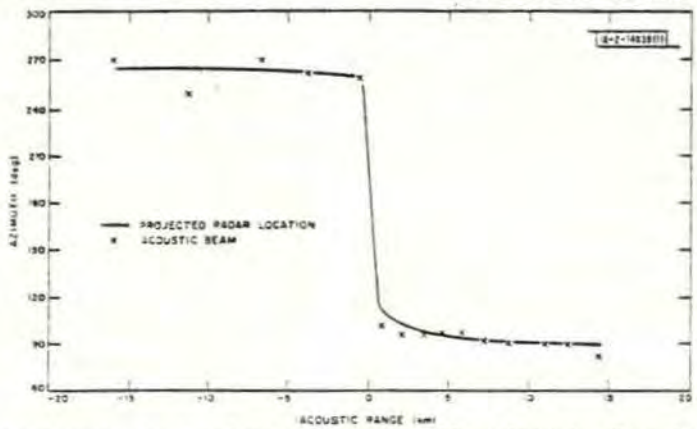


Fig. 34. Azimuth detections over the complete A-7 track. 200 m altitude. Negative ranges are inbound.

Reprinted from:

PROCEEDINGS OF THE IEEE, VOL. 66, NO. 11, NOVEMBER 1978

High-Level Protocols

ROBERT F. SPROULL AND DAN COHEN

Invited Paper

Abstract—High-level protocols (HLP's) are the high-level languages of distributed systems. In a resource-sharing network, HLP's link processes working on a common application. The design of an HLP is decomposed into three components: language, coding, and transport. The language expresses the commands and data passed between processes. It is designed to provide standardization and device independence, in order to use a small number of HLP's to address a range of applications implemented on a variety of computer systems. Coding converts the language into digital messages. Finally, a transport system is used to transmit the messages from one process to another—experience with HLP's has shown that different HLP's require different transport behaviors. This paper describes some examples of HLP's (ARPA network voice and graphics protocols), and argues that modern techniques for expressing structure and control in programming languages should be applied to analogous problems in communication among application processes in a network.

I. Introduction

HIGH-LEVEL PROTOCOLS (HLP's) are the high-level languages (HLL's) of a distributed computing environment. They are the means by which a communications network and processing resources in the network are harnessed in an orderly way to accomplish some task. Using different resources in an application on a single computer is by now natural, aided by operating systems, high-level languages memory management, and other well-established tools. But a resource-sharing computer network offers to an application the resources available throughout the network [18], [29]—to harness these distributed resources requires the collaboration of several computers and many computational processes, and, therefore, a set of conventions for harmonious communication that extends throughout the distributed environment. HLP's are these conventions.

The term "high-level protocol" attempts to distinguish protocols designed to control the computing processes involved in an application from "low-level protocols" designed primarily to control communication processes. Two computer processes require communications in order to transmit collections of bits from one to the other. But it is the HLP that assigns an interpretation to the bits, and thus allows each process to control the other. An analogy may be helpful. To call a department store on the telephone is simply to establish communication. But when the caller says, "Please send me two pillowcases, catalog number X802, and bill my account," he is using an HLP that controls ordering goods. Thus the HLP is implemented "on top of" the low-level communication protocols.

Manuscript received February 25, 1978; revised June 9, 1978. This work was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC-15-72-C-0308, ARPA Order 2223, and under Contract F44620-73-C-0074, monitored by the Air Force Office of Scientific Research.

R. F. Sproull is with the Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

D. Cohen is with USC/Information Sciences Institute, Marina del Rey, CA 90291.

Of course, HLP's that control computer processes need precise definitions of commands and responses, and cannot rely on human intelligence to provide an interpretation. (The paper by Pouzin and Zimmermann [26] in this issue introduces the notion of "protocol" and the concept of "layering" by which an HLP is implemented in terms of lower level communications protocols. Their term "application control" is equivalent to our "high-level protocol.")

The services available in a resource-sharing network are sufficiently diverse that several different HLP's need to be designed. For example, HLP's have been developed to control job entry, interactive terminals, file systems, graphics displays, resource allocation, and voice transmission in distributed environments. But why are HLP's required to control these services? Is not the communication provided by a network sufficient? The need can be demonstrated by an example. At first, using a file storage service seems to be a simple matter of sending a file to the service as a serial stream of bits, and subsequently retrieving the same stream. But additional attributes of the file must also be saved and retrieved: a file name, some information about how the file is to be protected from unauthorized access, accounting information to permit charging for the storage of the file, and so forth. Moreover, a file service must provide more operations than storage and retrieval: enumerating names of files stored, deleting or renaming files, moving files from one storage medium to another, etc. The HLP must select among these operations, provide the information needed by them, and report errors encountered as the operations progress, in addition to simply transmitting file data.

A resource-sharing network becomes increasingly valuable as more HLP's that offer services are designed and implemented. Thus an overall objective of protocol development is to seek methods and tools that simplify HLP design and implementation. Already, experience with HLP's has uncovered common design techniques, some of which are reviewed in this paper. But techniques for specifying HLP's and for implementing them remain primitive. We argue that methodology and tools for designing and implementing HLP's can be improved by applying notions developed for HLL's—in effect extending the HLL into a distributed environment. Just as the HLL frees the programmer from many tiresome details of programming a particular computer, the extension frees the programmer from the tiresome details of dealing with network communications.

To construct such an argument, a better understanding of actual HLP's is required. The remainder of this introductory section illustrates the range of HLP's used in a resource-sharing network. Section II builds a detailed understanding of the ingredients and requirements of HLP's by defining a model and characterizing some very different protocols in terms of the model. Section III reviews the issues faced in the design of HLP's, and common techniques used in the designs. Finally,

0018-9219/78/1100-1371$00.75

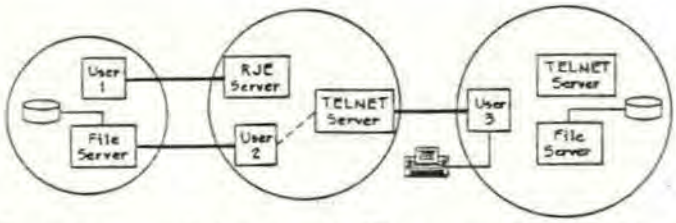


Fig. 1. A resource-sharing network. "User processes" obtain services from "server processes," under the control of HLP's. Heavy lines linking processes in the illustration denote the logical connections that carry HLP. These connections are unrelated to the physical organization of the network.

Section IV shows how many of the needs of HLP's can be addressed by HLL facilities.

A. The Role of HLP's

HLP's are used in two slightly different ways in networks. In the first use, the basis of resource-sharing networks, HLP's control the *services* offered by resources in the network. Various sites in the network implement *server processes* that offer a specific service to any *user process* that communicates requests to it (Fig. 1). In these settings, activities are instigated by the user process; the server is a willing slave. Examples of important services are as follows.

a) Remote job entry service: A user process submits to the server a file of text that controls running batch jobs (job control language). Later, the user process may retrieve from the job entry server the "listing" text produced as a result of the job.

b) Remote terminal service: A user process can "log in" to a timesharing system service, and subsequently send and receive characters as if it were a terminal on the system. We shall use the term "TELNET" for this service, adopting the name of the protocol used in the ARPA network [7].

c) File service: A user process can send data files to the service for storage and subsequent retrieval. The file service may be provided with conventional disk equipment, or may have special file-storage resources such as a mass memory.

d) Mail service: A service accepts text messages addressed to individual people. Later the individuals can interrogate the service through a user process to receive any mail that has been left for them.

e) Resource-sharing service: This service accepts requests from user processes to allocate resources of the computer system on which the service executes (jobs, processes, files, I/O devices), and makes these resources available to other processes in the network [36].

Each of these services has a corresponding HLP for accessing and controlling the server process. These services are analogous to the services that are offered on a single computer by an operating system. A mechanism for storing files, for example, is a vital service provided by an operating system. Extending this service into a resource-sharing network allows greater sharing of the file data, as well as economical storage of files with specialized hardware. Thus HLP's are not entirely new phenomena—similar concepts have existed within operating systems—but are now being revealed as they are extended into networks.

A second use for HLP's links in a symmetric fashion two processes that are part of a single application. Such a link may be the result of the division of a software system into separate modules. Or it may be that the communicating parties themselves are symmetric, as is the case in the network voice protocol described in Section II-D. In these cases, the user–server distinction is not relevant. Thus applications may use HLP's both to access services and to communicate among parts of an application.

HLP's offer many new opportunities to distribute applications. An application can be divided into processes, each of which can be executed on the computer best suited to its needs: signal processing tasks are accomplished on fast parallel machines; symbolic computation on computers with large address spaces suited to list processing. Processes that communicate directly with a human by voice, graphical, or textual information can travel with the user, communicating with collaborating processes in the network by packet radio links [19]. Distributed applications can evolve as requirements change, by modifying HLP's, adding new services, distributing the computing activity among a wider set of resources, etc. The ease with which HLP's can be designed and implemented will control the extent to which these opportunities can be fulfilled.

B. Standardization

Achieving the standardization required to offer services on a variety of computers is a major goal of HLP's. Before the appearance of large computer communication networks, HLP's were largely a private matter. A programmer who wanted to design an application consisting of several processes and wanted to provide a method for communicating among them, had only to establish some personal conventions for the interprocess communication. Networks spoiled this simple approach. Communication protocols became a matter of public decision in the hope that a single standard design could serve a wide community of computers and users; interconnecting a variety of computers unleashed a barrage of problems due to different representation conventions in different machines (word lengths, character sets, and the like) and in different operating systems (files, naming, and protection mechanisms); networks forced slower communication among processes than did tightly coupled systems and seemed to magnify problems of synchronizing processes.

The importance of standardization is demonstrated by the "n^2 problem." Imagine a network of n computers, each of which offers a *mail server* that uses a distinct HLP. Each computer will require a separate user program to send and receive mail for each protocol—a total of n programs on each computer or n^2 in all. By contrast, if the mail servers all implement a *standard* HLP for sending and receiving mail, only n user programs, one on each computer, are required to obtain access to all mail services in the network. The high cost of writing and maintaining software makes protocol standardization essential. The techniques and conventions that permit standardization over a range of computing equipment are a major ingredient in the design of HLP's.

II. Three High-Level Protocols

This section describes three HLP's as a prelude to a more general discussion of issues involved in designing and implementing HLP's. These examples are couched in terms of a model of three components of an HLP: language, coding, and transport. The first example, a *plotter protocol*, illustrates the model. The two remaining examples, a *graphics protocol* and a *voice protocol*, are actual HLP's that demonstrate a range of HLP design techniques.

A. The Components of an HLP

Designing HLP's seems at first to require a large number of decisions, ranging from communication formats to application requirements. The design problem can be decomposed into three aspects of HLP's: the *language*, the *coding*, and the *transport* mechanism.

Language: The language describes the functional intent of the communications exchanges between processes: What does one process "say" to the other? What responses does the other process generate? How does one process cause the other to alter its *state*? The language may include *commands* or other *control* information, and will usually include *data* as well. The designer of a language to offer a service must anticipate how other processes will use this service, and provide within the language the means for those processes to request and receive the service they need.

Coding: Any communication between two processes must be coded into a collection of bits that the processes know how to generate and interpret. For example, if the language requires transmission of a character string, it may be coded into a sequence of 8-bit bytes according to the ASCII standard, an integer value into a 32-bit two's complement binary representation, and so forth. Just as data structure declarations in an HLL impose structure on an otherwise homogeneous memory, coding imposes structure on a previously homogeneous communications path.

Transport: The communication path is provided by the transport mechanism, the means to collect bits from one process and deliver them to another. If both processes reside in the same computer, this mechanism is provided by the operating system or by the language environment in which the processes execute (e.g., "pipes" in Unix [28]). If the processes execute on separate machines, the transport may be provided by a network. We prefer to avoid stating which processes execute in which computer and consequently which transport is provided by networks and which by other means. We may be concerned with performance properties of the transport mechanism (e.g., speed, delay, buffering, errors), but not with the communications methods themselves. As a consequence of this view, we must assume that HLP's cannot depend on the processes sharing other resources, such as memory or files: the protocol must provide *all* of the interprocess communication. We shall use the term *message* to refer to the individual packages of bits that are transmitted from process to process. A network will typically use additional messages to help the transport mechanism operate smoothly: these lower level protocols are concerned with flow control, error recovery, and the like (for more discussion of lower level or "host–host" protocols, see [1], [2], [26]).

When designing and describing HLP's, it is important to separate these three aspects of interprocess communication (IPC). The language can be likened to HLL constructs involving control and data: procedure calls, coroutine calls, etc. Coding is analogous to choosing a computer representation based on collections of bits: integers, text characters, or computer instructions. The design of the transport mechanism is concerned with issues such as bandwidth, delay, and error recovery, and with the low-level protocols that provide the transport. If these three aspects are not kept separate, and we think of HLP's as configurations of bits flowing along communications lines, we will become hopelessly confused. Imagine trying to describe how a PDP-11 works by describing only bit patterns and signal transitions on the Unibus!

Fig. 2. Two processes using the plotter protocol. The application sends messages to the plotter process that completely describe the figures to be plotted. The two processes may execute on separate computers linked by a network, or on a single computer that provides message communication between the two processes.

The three components are not independent, but interact in ways that must be addressed by the HLP designer. Transport bandwidths or errors may affect the choice of coding techniques. Language choices often affect coding, and can also impact the design of transport methods. In the sections that follow, some of these interactions are demonstrated by the three example HLP's.

B. A Plotter Protocol

We shall illustrate in this section a simple HLP for controlling a pen plotter. Such a protocol could be used to offer "plotting services" in a distributed system: one or more "plotter server" processes are provided, each controlling plotter hardware in response to commands and data provided by the plotter protocol. In order to make plots, an application process such as a curve-fitting package establishes communication with a plotter process that will provide the plotting services. Fig. 2 shows the logical arrangement of these processes.

Language: The language for the plotter HLP is designed to allow the application program to produce drawings by issuing commands to the plotter process. The application program needs to be able to start a fresh plot, to raise and lower the plotter pen, and to draw straight lines. These commands in the language might be provided by procedures that are called by the application program:

BeginPlot	Procedure to place a new sheet of paper in the plotting area.
EndPlot	Procedure to signal that a plot is complete.
Up	Procedure to raise the plotter pen so that it no longer touches the paper.
Down	Procedure to lower the plotter pen so that it will mark on the paper.
LineTo(x,y)	Procedure to cause the plotter pen to move from its present location in a straight line to the point (x,y). The Cartesian coordinates specify a location on the paper, in units of centimeters from the lower left corner.

Notice that these procedures show no effect of the presence of a network–the same procedures could be used to control a plotter connected directly to the application program.

Coding: In order to transmit to the plotter process the commands invoked by the application process, each command must be encoded as a sequence of bits. For our example, we shall use 8-bit ASCII characters as the basis for the coding. A command is designated by a single character corresponding to the procedure name (B, E, U, D, L). The (x,y) arguments to the "L" command can be represented as two four-digit numbers in units of 1/100 cm. Thus the protocol sequence

B U L 0100 0100 D L 0200 0100 L 0200 0350 L 0100 0100 E

will plot a triangle with vertices (1,1), (2,1), and (2,3.5) cm. Of course, another encoding could be devised that requires fewer bits to code the same language. In any case, very little

computation is required to encode and decode the plotter language.

Transport: The transport system must deliver the individual 8-bit bytes to the plotter process. If a packet-switched network is used for transport, it is sensible for the application process to pack several dozen bytes into a message before the message is transmitted. However, the EndPlot procedure must force the transmission of the last message, even if it is only partially full, to insure that the final "E" code is delivered to the plotter process.

For the plotter protocol, a general-purpose interprocess transport mechanism such as TCP [2] is ideal. The transport reliability obtained from such a mechanism is essential for the plotter protocol: errors in the data, or disorder in the sequence of messages decoded by the plotter process will have disastrous effects. However, the protocol requires neither extremely high transmission rates nor extremely short transmission delays, because most plotters are quite slow.

Discussion: The plotter protocol illustrates two important aspects of HLP's: *standardization* and *device independence*. A resource-sharing network may offer several plotter server processes, all of which use a standard plotter protocol. Standardization is required for all components of the HLP: language, coding, and transport. The method of packing bytes into messages and the low-level protocols used to achieve reliable transport of messages must be standardized. Coding is standardized by the ASCII character standard and the definition of character sequences used by the plotter protocol. Finally, the concepts of the language are standardized: Up, Down, and LineTo are but one way to control a plotter, but they are chosen as the standard. The language standard imposes a method for controlling the plotter that may have far-reaching ramifications in the structure of the application program.

Standardization begets generalization: if a single standard is to be designed, it should be able to cope with as many similar uses as possible. This property is often called "device independence," although a physical device is not always involved. The idea is to make the protocol independent of particular hardware details by abstracting the elements common to the control of a range of devices. The plotter protocol described above is somewhat device independent: plotters with various resolutions (e.g., 100 steps per centimeter, 173 steps per centimeter) can clearly be used—the plotter process converts the standard units of 1/100 cm into the coordinate system used by the hardware it controls. Electrostatic plotters or storage-tube displays, even though they have no pen to be raised and lowered, can also be correctly controlled by the protocol because the only effect of protocol commands is to describe visible lines. However, colored pens on a plotter cannot be handled properly by our protocol—it is not general enough to handle more than one color.

The protocol illustrates a style of HLP design that derives the language from a standard subroutine interface. Each procedure call causes a record of the procedure name and arguments to be encoded and sent through the network. The receiver decodes the messages and usually calls corresponding procedures that actually drive the plotter. The network thus provides a *binding* mechanism by which calls to the five procedures in the application program are converted into calls to procedures in the plotter process. Ideally, the network is "transparent" to the application program—its presence cannot be detected!

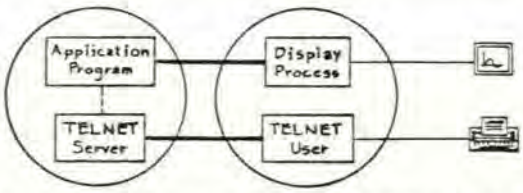


Fig. 3. An application using two HLP's. A person uses the TELNET service to control a curve-fitting application running on a time-sharing system. This program, in turn, uses a high-level graphics protocol to control a display that shows the curve-fitting results.

The plotter protocol is extremely simple compared to most real HLP's, which attempt greater device independence and which require bidirectional communication between processes. The following two sections describe two actual HLP's developed by the ARPA network community: a network graphics protocol (NGP) and a network voice protocol (NVP). Because these two descriptions are rather long, the reader may prefer to skip to Section III, which contains a summary of techniques used in designing HLP's.

C. A Network Graphics Protocol (NGP)

The objective of an NGP is to provide a standard way for an application program to control an interactive graphics display. It is a natural extension of the idea of the TELNET protocol which allows an alphanumeric terminal connected to one computer to act as a terminal to another. A user with a graphics display and computer capable of obeying the NGP would have access to a potentially large number of network resources that use graphical interaction (Fig. 3). Because users with different display devices will want to use these services, the NGP design must be independent of the particular device being used.

The plotter protocol developed in the previous section could be used as an NGP: it can control the generation of line-drawing images on a display. If a storage-tube display is used, the commands to the hardware are almost identical to those provided by the protocol. But an NGP needs greater generality, including the ability to handle a wide variety of display hardware, to provide fast interaction, and to cope with different kinds of graphical input devices.

The methodology we find appropriate for developing an NGP is to first design a device-independent graphics package (or copy an existing design) and then adapt it to network use. The device-independent design wrestles with the functional capabilities of the system and the model that the application program uses to invoke them. The network adaptation then need address only coding and transport.

The particular NGP described in this section was developed for use by the ARPA network [33], [14]. It is based on a "general-purpose graphics package" [31], [40], which attempts to cover a range of applications requiring interactive line-drawing displays.[1] We cannot, in this paper, take up the many issues that surround the design of such a package (see [24], [25]). However, we will discuss how network communications interact with the design. A summary of the protocol is given in Table I.

Language: The functions of a graphics package can be divided into three categories: those for defining an image to be

[1] Other protocols have been proposed for graphical needs that lie outside the NGP's objectives: carefully formatted text [32] and sampled images such as might be captured with a TV camera [22].

displayed, those for accepting responses from the user via graphical input devices, and those that inform the application program of important properties of the display. The discussion of the NGP language is likewise divided into considerations of output, input, and inquiry.

Language—Output: One of the most critical requirements of an interactive graphics system, and one that might be compromised by a network, is the speed with which the display can be altered—this is the crux of many interactive dialogues. This consideration forces us to abandon the plotter protocol: to make a small change to a display of 2000 lines would require transmitting a description of the entire image (roughly 40 000 bits, even with a fairly efficient coding). Furthermore, such a technique burdens the application program by making it recompute the image from its application data structure.

An effective technique that solves these problems is the *segmented display file*. Rather than manipulate one large description of the image, the graphics system maintains a set of independent segments that together comprise the display. Each segment is assigned a numeric "name" for identification, and contains a list of graphic primitives (lines or characters) that are part of the image. Each segment also has a flag (on,off) that indicates whether the primitives described in the segment are to be displayed on the screen. The system provides facilities for specifying the contents of a specific segment, and for altering the flag. To create a segment named 8 that contains a description of the triangle in the plotter example, the application might call the following procedures in the graphics package:

OpenSegment(8)	Says primitives are to become segment 8.
MoveTo(1,1)	Primitives defining triangle.
DrawTo(2,1)	
DrawTo(2,3.5)	
DrawTo(1,1)	
CloseSegment()	Ends construction of segment 8.
PostSegment(8)	Sets segment 8 flag, so it will be displayed.

The calls simply build a display file that a hardware display processor interprets to refresh the display. Changes to the display file as a result of these calls are reflected as changes in the image on the screen.

The segment thus becomes the unit of information that is easily changed with each interaction. It is not difficult to write most application programs so that the segments correspond to logically separate parts of the display in such a way that display alterations are often limited to a small fraction of all the segments. If information within a segment must be changed, the segment is entirely rebuilt with calls such as those above. Or the segment may be deleted if its contents are no longer needed. Or the information in a segment can be retained for possible use later. This technique is useful for recurring imagery such as messages and menus.

The graphics package also provides an "update" function. The application program calls this function to identify points at which the display *must* accurately reflect all changes that have been requested by calls on the graphics system. These points usually directly precede requests for new user input: the user must view an up-to-date display to formulate his input. This function plays an important role in device independence, providing good service to storage-tube displays by reducing the number of screen erasures [31].

On a single computer, a device-independent graphics package of this sort is implemented as shown in Fig. 4(a). The package

TABLE I
SUMMARY OF ARPA NETWORK GRAPHICS PROTOCOL. EACH HAS THE SAME FORMAT, A COMMAND FOLLOWED BY ARGUMENTS. THE ARGUMENTS ARE LISTED INSIDE BRACKETS (( )). DETAILS OF THE PROTOCOL ARE FOUND IN [33]

```
Segment control:
  seg.open <seg.name>
  seg.close
  seg.post <seg.name>
  seg.unpost <seg.name>
  seg.kill <seg.name>
  seg.append <seg.name>
  end.batch.of.updates

  seg.dot <x.coordinate> <y.coordinate>
  seg.move <x.coordinate> <y.coordinate>
  seg.draw <x.coordinate> <y.coordinate>
  seg.text <text.string>

  sel.intensity <count>
  sel.type <count>
  sel.character.orientation.discrete <count>
  sel.character.orientation.continuous <fraction>
  sel.character.size.discrete <count>
  sel.character.size.continuous <fraction>

  change.attribute <seg.name> <attribute>
   _possible <attribute> arguments:
          <sel.highlight.attribute> <on/off>
          <sel.hit.sensitivity.attribute> <on/off>
          <sel.intensity.attribute> <count>
          <sel.screen.select.attribute> <mask>
          <sel.position.attribute> <x.coordinate> <y.coordinate>

  seg.readback.seglist
  seg.readback.seg <seg.name>

Positioned text:
  ptext.open <ptext.name> <count> <x.coordinate> <y.coordinate>
          <x.coordinate> <y.coordinate> <count>
  ptext.kill <ptext.name>
  ptext.set <ptext.name> <mask>
  ptext.scroll.up <ptext.name> <string.number> <string.number>
  ptext.scroll.down <ptext.name> <string.number> <string.number>
  ptext.move <ptext.name> <string.number> <ptext.name> <string.number>
  ptext.edit <ptext.name> <string.number> <pos1> <pos2> <text.string>
  ptext.modify <ptext.name> <string.number> <pos1> <pos2> <ptext.feature>
  ptext.remote.edit <ptext.name> <ptext.string>

Input functions:
  input.enable <technique.number> <disable.condition> <report.sequence>
  input.disable <technique.number>
  input.report <technique.number> <technique.summary>
  input.reset.system
  input.drag.sel <technique.number> <seg.name>

Inquiry:
  inquire
  inquire.response <count> <response.phrase> _
   _sample <response.phrase>s are:
          <i.implemented.commands> <mask>
          <i.screen.coordinates> <x.left> <y.bottom> <x.right> <y.top> <count>
          <i.screen.size> <text.string> <text.string>
          <i.screen.number> <count>
          <i.terminal.name> <text.string>
          <i.terminal.type> <terminal.type.value>
          <i.intensities> <count>
          <i.line.type> <count>
          <i.characters> <mask>
          <i.character.orientations> <count> <fraction> _
          <i.character.size> <count> <character.size.description> _
          <i.available.input.technique> <technique.number> <text.string> _

Miscellaneous:
  escape.protocol <text.string>
  synchronize <count>
  reset
  error.string <text.string>
```

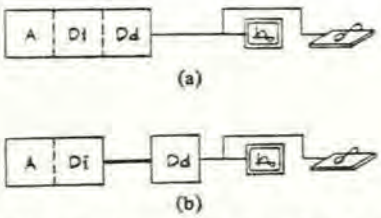


Fig. 4. The organization of a device-independent graphics package. (a) On a single computer, the software is divided into three modules: the application (A), the device-independent module (Di), and the device-dependent module (Dd). (b) The system can be adapted for a network implementation by using communications to provide the linkage between Di and Dd.

is divided into two modules, one device independent (Di), and one device dependent (Dd). Each call by the application program (A) is subjected to some Di processing, and then usually results in a call on the Dd module. The Dd module is responsible for building the segments themselves in a form that can be interpreted by the particular display processor in use. The Di module can be used with many different Dd modules with the same procedure interface—in this way, the application program can operate a variety of different display devices.

Adapting this configuration to a network environment is easy (Fig. 4(b)). The module resides with the application program and the Dd module runs in the computer that directly controls the display hardware. Network communications are used to achieve the effect of the Di module calling procedures in the Dd module. Because the procedure interface between the Di and the Dd parts is standardized to allow alternative Dd modules, the standardization is easily retained in the network protocol. Thus the segmented display file, a mechanism devised to solve problems of interactive graphics programs in single computers, turns out to be adapted easily to networks.

Language–Input: The NGP provides access to a variety of graphical input devices, for it is the interaction afforded by these devices that makes graphics an important tool in many applications. Unfortunately, achieving device independence for input is much harder than for output, in part because of the enormous range of input devices. An additional problem is that effective use of input devices requires generating on the display screen various *feedback images*, such as cursors, to allow the user to coordinate his input with images already displayed. The range of feedback techniques is even wider than the range of input devices!

An approach to device independence is to concentrate on the interactive techniques being used rather than on particular input devices. The display hardware and Dd module cooperate closely to provide rapid feedback to implement the technique. After the interaction is complete, the Dd module reports to the application program a terse summary containing essential details of the entire interaction. For example, three techniques in the repertoire are the following.

1) Positioning: The user is expected to identify a spot on the screen. He uses a coordinate input device to steer a displayed cursor to the desired spot, and then depresses a button to indicate "this is the point I mean." The summary returned to the application program contains only the coordinates of the cursor when the button was struck.

2) Stroke collection: The user is expected to draw a trace on the screen by steering the cursor at the same time the button remains depressed (the button is usually a switch on a stylus input device). The display shows the resulting trace as feedback, which presents an effect similar to drawing. The summary reported to the application program is a list of points that lie on the trace.

3) Pointing: The user is expected to identify an item already displayed on the screen. One way to implement this technique is to steer a cursor over the item and push a button. Alternatively, a light pen can be used to identify the item. The summary reported contains a specification of which item was identified.

This entire approach focuses on input *techniques* rather than on *devices* alone. It may limit the flexibility of the application program in using some device in a clever way. However, the flexibility is available instead to the Dd module, which can implement the techniques in various ways, and can perhaps customize the implementation for a particular user.

Language–Inquiry: Not all aspects of the interface to the Dd module are completely standardized; some are parametrized. For example, some displays allow lines to be distinguished by their brightness, although devices differ in the number of brightnesses provided. A standard that assumes only one brightness prevents the use of this technique entirely. One that assumes a fixed number of brightnesses will cause confusion on a display that provides but one. A solution to problems of this sort is to let the application program or Di module *inquire* about features of the display hardware, and thereafter drive the display accordingly. Examples of parameters that can be handled this way are: number of brightnesses, number of line types (e.g., dotted, dashed), physical screen dimensions, and the kinds of input devices available.

Some inquiry results can have a profound effect on the operation of the application program. Many features of the protocol are optional and may not be implemented by the hardware or by the Dd module; inquiry establishes which features are absent. Another sort of inquiry reveals the type of display being used: is it a storage tube or a refreshed display? The application program may choose entirely different interaction styles for these two display types, because changes to storage-tube images are much slower than changes to refreshed images.

Coding: The coding for the NGP is very simple: we encode procedure calls from the Dd module to the Di module as a sequence of 8-bit bytes. The first byte identifies the procedure. Subsequent bytes provide arguments; each procedure has an associated format for arguments. The protocol was designed to avoid floating-point numbers; only integers are used.

Information returned by the Dd module to the Di module is structured symmetrically, as if the Dd module were calling procedures in the Di module to answer queries, to report interaction results, and so forth.

Transport: The coding generates a stream of bytes to be transmitted, just as the plotter protocol does. Because most of the encoded procedure calls do not return answers, the bytes can be packed into messages; messages are transmitted when full. It is occasionally necessary to transmit partial messages: for example, the "update" call forces all information encoded by Di to be transported to Dd so that the display will change. It may also be important to transmit quickly commands that alter the handling of input techniques.

Summary: We have seen that developing the ARPA NGP was primarily a matter of designing device-independent graphics package. As the package modules are designed, we find that some are device-independent and others depend on the details of particular hardware. Thus a natural division of labor between two computers is to execute device-independent mod-

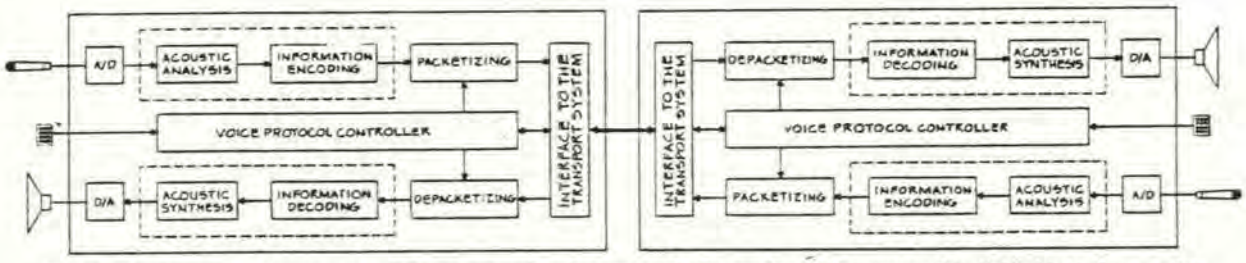


Fig. 5. The structure of processes using the network voice protocol. Processing of the protocol is divided into three parts: sending speech data, receiving speech data, and controlling the transmission. Portions enclosed by dashed lines are "vocoder dependent."

ules and the application program on one machine, and place the device-dependent modules and display hardware on another. A network provides communication between the device-independent and device-dependent modules in a straightforward way. (If we were willing to standardize the procedure interface between the application program and the device-independent module, this interface too could become a point of separation by a network.) Thus, methodology, design, and documentation of the NGP are those of a graphics package, rather than of an *ab initio* protocol design. In this setting, the network becomes a relatively unimportant artifact of the implementation: it offers interprocess communication among modules of a software system.

D. A Network Voice Protocol (NVP)

The objective of an NVP is to control the transmission of a digital representation of voice signals through a packet-switched network in real time. The transmission may be part of a man-to-man or man-to-machine communication. A protocol can provide the effect of two-party or conference calls, including user interfaces such as dialing and ringing. Or it can provide a voice message service by communicating from man to a computer capable of first recording the voice data, and and later responding to a request to repeat the data to the intended recipients. A voice protocol differs from a standard telephone connection in that applications may use the network to transmit additional digital information along with the voice data.

A voice protocol must provide natural communication between humans. On the one hand, this objective seems to require extremely good performance. Poor reproduction of the speech signal may prevent recognition of the speaker; long transmission delays may impede a dialogue; small amounts of distortion may be keenly perceived and interfere with natural communication. On the other hand, a human can tolerate many errors. Occasional noise is compensated for by the redundancy in human speech; catastrophic garbling is repaired by dialogue—the utterance is repeated.

A voice protocol should try to provide good speech quality even though data transmission rates may be severely limited by the network. Fortunately, techniques are available for encoding voice as low-bandwidth digital data; these techniques are called *vocoding techniques*. Vocoding uses real-time signal-processing algorithms to encode and decode the voice. Vocoding algorithms tend to require substantial computing to achieve both acceptable reproduction quality and low data bandwidth. High compression rates also tend to increase sensitivity to acoustic noises and communication errors. Speech quality, data rate, and processing all interact: achieving higher quality or less processing requires more transmission bandwidth. One of the motivations for a device-independent NVP is to simplify experimentation with different compromises.

The needs of a voice HLP differ from those of a graphics HLP in several ways. The dominant concern in an NVP is the vocoding scheme used, often because special fast processing is required to vocode in real time. Language issues are relatively minor. A voice protocol requires a real-time transport system: delays, and especially delay distributions, are critical to offering real-time service; errors, however, can be tolerated. By contrast, the NGP can tolerate longer delays and greater variation in delay for "interactive" service, but cannot tolerate transmission errors.

An experiment in voice protocols has been undertaken by four sites in the ARPA network [4], [5]. Sites with differing hardware for implementing network control and vocoding algorithms were deliberately chosen in order to develop a protocol to accommodate such differences.

Language: The language for the NVP has three functions. First, it must cope with different sending and receiving hardware by achieving device independence in some way. Second, it must support user interfaces such as ringing a telephone, waiting for an answer, and hanging up. Third, it must control the transmission of a voice signal in real time. The protocol separates these functions into two distinct parts: a control protocol and a data protocol. These are separate protocols that link separate processes (see Fig. 5). The data processes are concerned only with transmitting speech data; the control processes provide the other two functions of the language.

Some of the NVP language is devoted to controlling the conversation and to exchanging information about its status. For example, the caller process tries to initiate a call with "Calling ⟨who⟩ from ⟨whom⟩," where each field addresses one of several telephone lines connecting to the NVP equipment. The answerer may reply "Ready" or "Goodbye, I am busy." If all goes well, the answerer may say "Ringing," and eventually "Ready" if the call is answered. These exchanges are simply designed to allow caller and answerer to know each other's state: ringing, ready, terminated. (See Fig. 6 for a summary of the important states of the NVP, and Table II for a summary of the protocol.)

A more interesting part of the control protocol is the negotiation of various parameters of the coding process. Although the parameters are mentioned below, we shall illustrate the negotiation process here. One of the two processes (caller or answerer) is the *negotiation master*. This process makes proposals for parameter values that the other may accept or reject. A proposal takes the form "Can you do ⟨what⟩ ⟨how⟩?" The ⟨what⟩ and ⟨how⟩ fields are coded from a small set (⟨how⟩ is actually a list of ways of doing ⟨what⟩ that are acceptable to the master). For example, the master asks "Can you do ⟨speech

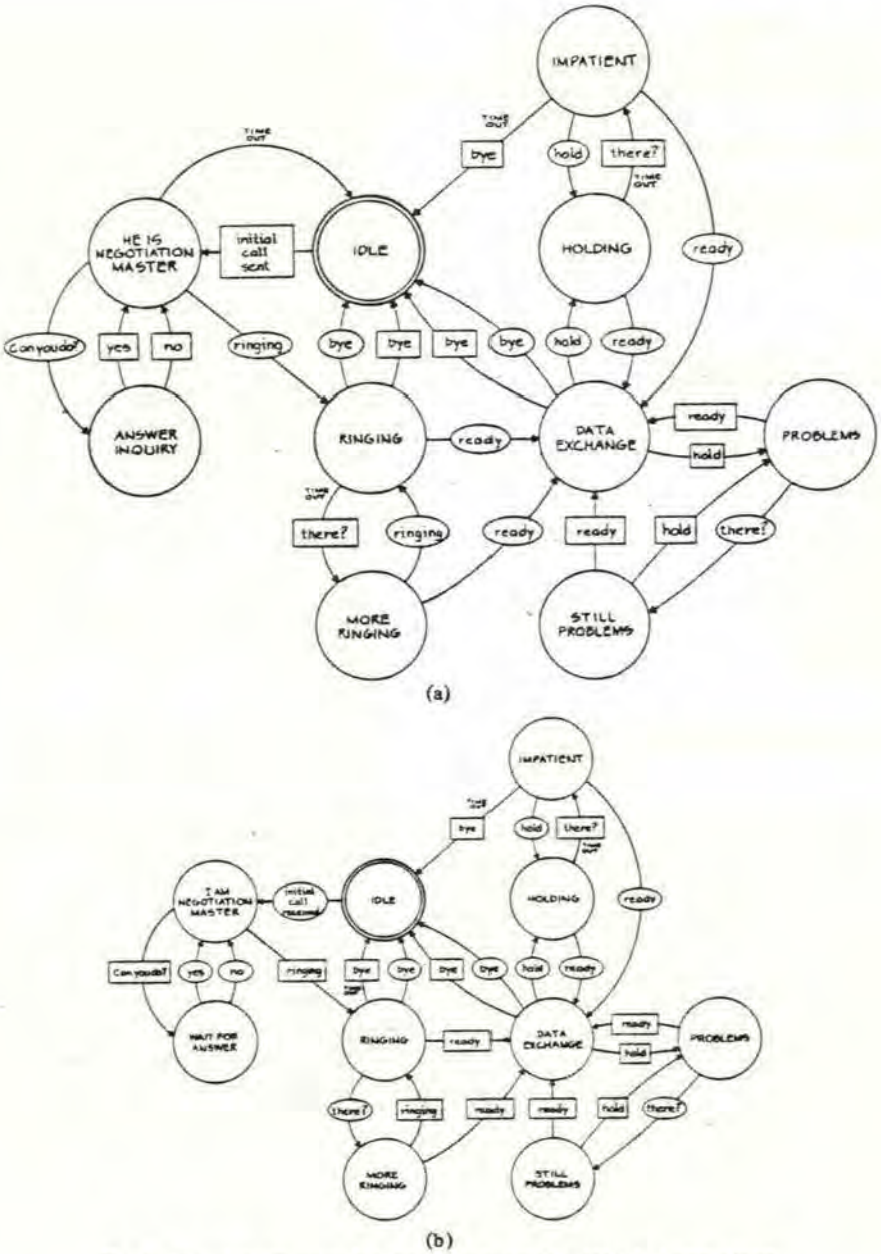


Fig. 6. The state diagram for the network voice protocol. Large circles indicate states. Transitions between states occur when a message is transmitted (arc with box) or received (arc with oval). Note that the caller and callee state diagrams are nearly symmetric.

signal sampling every⟩ ⟨62 microseconds⟩ or ⟨100 microseconds⟩?" The other process, the *negotiation slave*, may respond in one of two ways. A positive response is "I will do ⟨speech signal sampling every⟩ ⟨62 microseconds⟩." A negative reply says "I can't do ⟨speech signal sampling every⟩ in any way you suggested," but may optionally contain a suggested ⟨how⟩ that is acceptable. Although at any instant one of the processes is the master and one the slave, either may request renegotiation to exchange roles.

An additional *version* provision makes the negotiation more convenient. If a large number of parameters must be agreed upon, negotiating each one is tedious. We define a version to be a table of (⟨what⟩, ⟨how⟩) pairs that gives settings for a large number of parameters. Then the single negotiation "Can you do ⟨version⟩ ⟨2⟩?" will establish many parameter settings.

This negotiation scheme is designed to be extremely robust. Messages lost or delivered out of sequence will not garble the negotiation because each response repeats the question (⟨what⟩) as well as providing an answer.

Coding: Strictly speaking, the voice protocol is concerned only with controlling transmission of voice data, and not with the techniques used to generate the data. Nevertheless, good vocoding techniques, providing acceptable quality speech with modest transmission bandwidths, are the key to an NVP. A vocoder may be presented with 16 000 10-bit samples of speech waveform each second, and be required to encode this information in as few as 3500 bits. Clearly, such encodings cannot preserve all of the information in the original signal, but must strive to make errors that have only small perceptual effects.

TABLE II
SUMMARY OF ARPA NETWORK VOICE PROTOCOL MESSAGES. DETAILS OF THE PROTOCOL ARE GIVEN IN [4]

```
Control messages:
  calling <who> <whom> <response-link>
  goodbye <code>
  negotiation.inquiry <what> <list.length> <how.list>
  positive.negotiation.response <what> <how>
  negative.negotiation.response <what> <try.this.value>
  ready
  not.ready
  inquiry
  ringing
  echo.request <id>
  echo <id>
  renegotiation.request <master.request>
  renegotiation.approval <master.request> <ok>


  examples of <what> is negotiated:
          <vocoding>
          <sample.period>
          <version>
          <max.msg.length>
          <LPC.degree>
          <CVSD.time.constant>
          <samples.per.parcel>
          <LPC.acoustic.coding>
          <LPC.info.coding>
          <LPC.pre.emphasis>
          <LPC.tables.sel>

Data messages:
  <time.stamp> <skipped.parcels.flag> <parcel.count> <parcel> ...
```

Although a great many signal coding techniques are available [12], the ARPA NVP is currently experimenting with two: continuously variable slope delta modulation (CVSD) and linear predictive coding (LPC) [23]. These two techniques are very different; CVSD aims to produce an output waveform that *looks* close to the original, whereas LPC tries only to produce an output signal that *sounds* like the original. Experience gathered in recent years has made LPC the more popular technique for low data-rate vocoding.

Both vocoding schemes operate within a similar framework. An interval of speech is sampled and encoded by calculating the values of a number of parameters. These values are then packed into *parcels* for transmission through the network. The negotiation protocol is used to establish the type of vocoding used (LPC or CVSD), the sampling rate, the length of the period encoded in each parcel, and other details of the coding.

CVSD vocoding requires only modest processing, but does not achieve very good compression. Typically, 62 ms of speech is encoded into about 1000 1-bit parameters that are packed into a parcel for transmission, thus requiring about 16-kbits/s transmission rates. Some additional reduction in data rate comes from detecting inactivity of the speaker due to breathing, intersentence pauses, or listening to the other party. These periods of silence can be coded with very few bits; both CVSD and LPC coding take advantage of this reduction.

LPC employs two coding steps to reduce the volume of speech data. First, an *acoustic* vocoding is applied to a short period of sampled speech signal, and yields about a dozen parameters describing the signal. Second, an *information* coding step converts numeric values of these parameters to a smaller range of values and packs them into a parcel for transmission. The parcel is the unit of information flow, containing about 60 bits generated every 19.2 ms. Several parcels are packed together (up to about 14) into messages for transport through the network. Thus only 3500 bits/s are generated. This number can be reduced even further by use of "variable rate LPC" that takes advantage of acoustic redundancies (e.g., long vowels) to reduce the information content of the parameters.

The LPC acoustic vocoding algorithm fits a model of the human vocal system to a 19.2-ms interval of speech and delivers parameters describing the fit. The inverse process is applied to regenerate the speech, but it does not yield a signal identical to the original. The result is expected to *sound* like the original even though it does not *look* like the original.

The information coding step is simple compared to the LPC calculations. Although the LPC parameters vary over a wide range, the full precision need not be transmitted. The range of the parameters is broken into 32, 64, or 128 intervals with roughly equal probability of occurrence; parameter values are mapped via tables into either 5, 6, or 7 bits for transmission. The receiver inverts this nonlinear transformation.

Transport: The NVP requires real-time transport of parcels. Unfortunately, none of the standard transport facilities designed for HLP use in the ARPA network [1], [6] could provide such service; the NVP designed its own real-time protocol using only the facilities of the IMP-based communication subnet.

The real-time protocol is very simple and is applicable to real-time uses other than speech. Parcels, generated at a constant rate, are collected into messages. Each message is time-stamped with the time of creation of the first parcel in the message (the units of time measurement correspond to the time interval represented by a single parcel). During periods of silence, parcels are simply not generated, although any partially filled message will be transmitted promptly as the silence begins. (The treatment of silence is in fact somewhat tricky; see [3].) The time stamps allow the receiver to resequence messages that arrive out of order and to meter the reproduction of speech from the parcels.

Irregularities in message delays in the network cause trouble [13]. Suppose the first message experiences 340-ms delay and the receiver begins processing the first parcel of this message as it is received. If the next message experiences 360-ms delay, the receiver will finish the last parcel of the first message 20 ms before the next message arrives. What is it to do? The receiver could have delayed processing the first message somewhat in order to allow more leeway against subsequent longer network delays. This leeway can be allowed to grow as longer delays occur. However, if the *variation* in the network delay is high, the receiver's delay may become so long that an interruption by the listener is delayed awkwardly or that the speaker notices a pause when he stops speaking and begins listening. Low variation in network delay is a clear requirement for an NVP.

Very few conventional transport services are required by an NVP. No retransmission to remedy errors is required because a lost message is not catastrophic; both coding schemes are designed so that a parcel is decoded without knowledge of previous messages. Moreover, retransmission would introduce highly variable delays that cause worse perceptual damage than the loss of the message. The only effect of a lost message is an audible error for the period represented by the missing data. Because of the real-time nature of the NVP, there is also no need for user-enforced flow control: sender and receiver are processing messages at identical rates.

Results: The ARPA NVP was first used successfully in

December 1974 for real-time voice communication through the ARPA network from coast to coast. It has been implemented at five sites and has been used regularly since then. It has been used experimentally with LPC vocoding at rates between 1.8 and 5 kbit/s, and with CVSD at rates between 8 and 18 kbits/s. Although only a few sites have participated in the ARPA NVP, dissimilarities among the hardware and software environments have thoroughly exercised the device independence aims of the protocol. Some of the initial implementations used one-of-a-kind experimental devices such as the Lincoln Laboratory TX-2. A typical site uses a minicomputer attached to the network and to a fast arithmetic unit used for vocoding algorithms (e.g., PDP-11/45 and FPS AP-120A).

The NVP has been extended to handle the "higher level" functions of storing voice on files and of conferencing. In both cases, the data and negotiation protocols remained unaltered while extensions were made to the control protocol [5]. In conferencing, for example, control messages are used to establish which speaker "has the floor," and to whom speech should be sent. Additional extensions to transmit voice to a speech recognition system or from a speech synthesis system are feasible because LPC is the basis for some approaches to these goals [27]. All of these extensions are accommodated with only minor changes to the protocol because the basic NVP framework and vocoding required no alterations.

Although the voice application seems to be an unrepresentative HLP, we have seen that it offers a new perspective on several HLP functions. It shows a simple elegant negotiation mechanism. It shows the effort that must sometimes be devoted to data compression. It demonstrates a real-time end-to-end protocol. And most importantly, it places an unusual set of requirements on the transport system: low delay variation, minimal flow control, and no need for extreme reliability. Diverse needs such as these require the transport system to be flexible.

III. Designing and Implementing HLP's

This section assembles the various techniques used to design and implement HLP's by referring liberally to the three illustrative protocols of Section III, and also to other HLP's. This compendium is a prelude to an examination of the relationship between HLP's and HLL's. Our observations are grouped into the categories of language, coding, and transport.

A. Language

The language for an HLP is clearly the most important part of the design. Meeting device independence and standardization objectives is a major challenge; it is difficult to abstract the common properties of a range of devices and applications in a design that has a simple form. It is often difficult to avoid letting quirks of inflexible operating systems or of special hardware completely destroy an otherwise simple protocol. The network is not an extrinsic contributor to the problem: it adds a few new difficulties to those encountered in designing a nonnetwork system involving multiple asynchronous communicating processes. Nevertheless, the presence of the network makes communication along such diverse computer hardware and devices possible; thus it is the reason for much of the pressure for device independence and standardization.

Device Independence: Inherent in device-independent designs is a tension between generality and standardization. As more capabilities of devices or a wider range of applications are encompassed by the design, standardization becomes more difficult. The NGP designers, for example, wanted to allow dynamically moving images to be displayed, provided that the display hardware made some provision for applying geometrical transformations to the display file rapidly enough to see motion. The attempt failed because the designers could devise no way to standardize the range of transformation facilities provided by display manufacturers. However, if aspirations of device independence were reduced, and the NGP were required to operate only displays of a certain type, dynamic images could be accommodated easily. Thus an attempt at generality was curbed by standardization problems.

Generality can also lead to complexity. Overly general designs are difficult to implement, may take a long time to devise, and often require more computing resources to process. One way to reduce this problem is to divide the facilities of an HLP into quasi-independent groups of functions with different degrees of generality. Any particular implementation may offer only a subset of the groups. The NGP, for example, recognizes four groups of commands (status and inquiry, segmented display files, positioned text, input interactions). Entire groups are optional—facilities for input interactions, for example, need not be provided. Within a group are both mandatory and optional commands. If the options are not used, a very simple implementation results. If implementers need the added features, the NGP specifies a standard way to provide them.

At the other extreme are protocols that are not very general, but are extremely simple. These unambitious HLP's can offer substantial service. A file transfer protocol that deals only with text files, or a graphics protocol that is little more ambitious than our plotter example, is remarkably useful. A design that provides a special feature in a device-independent way is worthless if it is never used!

HLP's, whether simple or general, usually provide some measure of device independence. Two approaches are used to deal with different devices.

1) The "virtual device approach." A standard is designed that specifies the operations that a process *must* implement. The plotter protocol is an example. The plotter process may be able to forward some protocol commands to the plotter hardware directly; others it may have to "simulate" in some way. For example, if the standard protocol specified colored lines, the plotter process might replace them by dashed lines if no colored pens are available. This approach derives its name from the similarity with "virtual memory:" a computer that provides a million words of virtual memory need not have that much primary memory, but a collection of mapping and paging processes simulate the effect of a million words of primary memory.

2) The "parametric approach." In this approach, the process preparing to send protocol commands will *inquire* of the plotter process information about its capabilities: How many colors can be plotted? What is the size of the plotter surface? What resolution? Given the answers to these questions, the sending process can arrange to use only features of the plotter that are available. Although this technique still requires standard ways to specify all possible plotter commands and standard ways to inquire, the burden of simulating in software those capabilities lacking in the plotter is shifted to the sender.

One benefit of the parametric approach is that the application program can be advised of the plotter's properties and can take advantage of the information. For example, if color is not available, the application program can simulate it with line textures that do not conflict with textures already used by the

program. The technique also allows coding to be parametrized: for example, the plotter process could specify how many digits it requires for each coordinate value.

An important generalization of the parametric approach, called *negotiation*, makes the inquiry activity symmetric. Each process may inquire about properties of the other, often by asserting its request for a certain behavior: sender says "Can you do four colors?" and plotter replies either "Yes, I'll do four colors" or "No, I cannot do four colors." This technique allows each process to adapt to the other—they may end up being each parameterized according to some of the other's wishes. Once again, the method for negotiating, and the eventual commands that flow, must be standardized.

The basic difference between these two approaches concerns how the burden of achieving device independence is divided between the two communicating processes. The parametric approach, used by the NGP, NVP, and TELNET [7], essentially delays the specification of certain parameters until the two communicating parties are connected.

The methods used to achieve device independence can profoundly influence the structure of application programs that use the HLP. For example, the plotter protocol and the NGP are both graphics protocols, but a program must use them very differently, because they use fundamentally different techniques to make changes to the display. An application written to use a segmented display file could not be adapted easily to use the plotter protocol. Moreover, the parametric approach to standardization may induce the application to operate differently for different parameter values. For example, an interactive graphics application may drive a storage tube very differently than a refresh display (which reflects display changes more rapidly), even though the program uses the same device-independent methods to draw lines and text or to receive input.

Combining Protocols: An application may use several separate HLP's to connect its constituent parts. The different protocol uses are often independent, and use separate logical communication paths. But sometimes HLP's can unwittingly compete for resources. A good example is illustrated in Fig. 7(a), in which a user employs a TELNET protocol to control a graphics program, which in turn uses a graphics protocol connection to control a display. Most users prefer to see text and graphical output presented on the same screen (Fig. 7(b))—TELNET and the NGP thus compete for screen space. They also compete for keyboard input. But neither protocol is aware of the other's needs. The NGP design group considered allowing the NGP to control the placement of TELNET text on the screen, but abandoned the idea when it became clear such control would require synchronizing the display process (Dd) and the TELNET process (Tu) whenever screen placement changed. This problem was attacked seriously in a subsequent graphics protocol designed to allow careful control of "windows" on the screen, some of which contain TELNET text, and some graphical information [32], [35].

Robustness: An HLP provides a vital communication link in an application distributed over several computers, a link whose failure may be costly. If a person spends an hour using an interactive data-analysis program that uses the NGP to display results, and a communication error causes so much confusion that the program aborts, a great deal of work is lost. Providing the reliability to prevent these failures requires attention at all levels. The application program can save a user's state periodically in case of catastrophe; network transport facilities

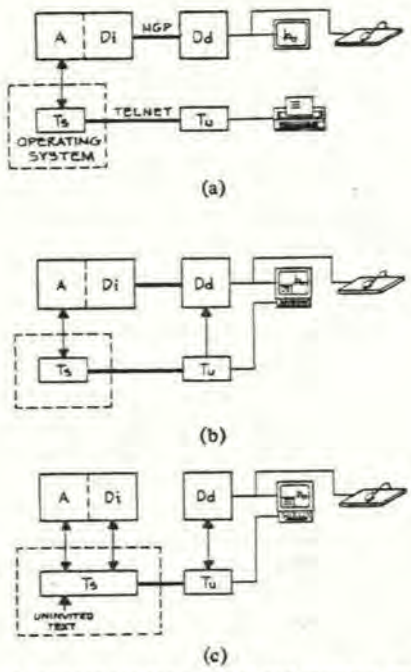


Fig. 7. Logical connections for graphics and terminal protocols. (a) Independent connections, with the display and terminal physically separate. (b) Independent connections, but "terminal text" is presented on the display. (c) A single connection for both graphics and terminal protocols, showing the insertion of "uninvited text" by an operating system.

can provide some measure of error detection and correction [1], [2], [18] and, finally, the HLP language can take steps to improve reliability, such as tolerating lost or duplicated messages. Redundant coding can be used to help detect errors. To avoid loss of synchronization in a stream of commands and arguments, a common technique is to occasionally identify the beginning of a command, for example, by marking a message to indicate that a command appears as the first item in the message.

Specification: Implementations of HLP languages usually depend on the proper interpretation of a rather detailed specification. The detail results chiefly from the requirement that all implementations of the HLP communicate with an identical language, rather than from complexity in the HLP itself. Indeed, the designer of an HLP seeks a simple elegant solution that offers the necessary services and has a straightforward implementation.

A precise HLP specification is difficult to write. Although it is similar to a specification for a "software interface" routinely used to detail the design of a module of a software system, it differs in two respects. First, it must be understood by HLP implementers accustomed to different programming languages, operating systems, conventions, and nomenclature. Second, because the implementer may not be familiar with the application addressed by the HLP, the specification must provide considerable implementation guidance. We shall argue in Section IV that notions from HLL's can be used to improve HLP specifications.

Layering: Layering helps to avoid HLP complexity: one HLP can be implemented *on top of* another. One sort of layering is exploited by all HLP's; they use some services of low-level transport protocols to provide basic communications services [1]. The plotter and NGP examples rely on this layer to provide *reliable* transmission.

HLP's can also be layered atop other HLP's. Protocols for conferencing and voice recording use the NVP as a base. An HLP that uses a coding based on character strings may be implemented on top of an HLP such as TELNET that transmits character streams between different computers.

Sometimes the opportunity to layer one HLP on top of another is impeded by poor protocol design or a shortsighted implementation. The NGP, for example, might have been layered on top of TELNET, so that the two connections of Fig. 7(a) could be reduced to one (Fig. 7(c)). This simplifies the implementation of Dd and Tu, requiring only one process. It also simplifies implementation of careful control of terminal text, as the single connection avoids the need to synchronize the display parameters in Dd to the arrival of text from Tu. However, the single connection approach was rejected by the ARPA NGP as too error-prone. Because an operating system may unexpectedly generate text to be printed on the terminal (e.g., "System going down."), it may interfere with graphical information encoded as text on the Ts–Tu connection. Clearly, the TELNET protocol and operating systems could provide a means for transmitting encoded information that would not be subject to disturbance. Such a facility, which would allow an HLP to be implemented within the TELNET protocol, was not available when the NGP was designed.

B. Coding

Coding can be a simple matter or a very complex one indeed. It is simple if we need only transmit integers or symbols from a well-specified vocabulary (e.g., an integer that identifies a command). Straightforward codings are easy to encode and decode; the NGP and NVP control messages are examples of simple codings. NVP data messages are simple encodings of parameters, although the parameters are extracted from speech samples by a complex vocoding algorithm. Although greater investment in processing can be used to compress the coding, no HLP's known to the authors use sophisticated minimum-entropy information coding techniques. This may be a result of greater attention to the less well understood problems of language and transport than to coding efficiency. Moreover, large efficiency gains can often be achieved by appropriate design of the protocol language: the segmented display file concept in the NGP offers a greater reduction in data transmission than would minimum entropy coding of entire images.

Coding problems arise when we try to encode objects that have different interpretations in different computers: "floating-point numbers" or "text characters" are examples. What happens if a "character" has no representation in the character set of the computer to which it must be transmitted? What happens when a floating-point number lies outside the range of numbers representable on the destination computer? Although it is possible to characterize precisely the acceptable ranges, it is extremely difficult to design a protocol that works properly when a sender wishes to transmit an object that cannot be interpreted by the receiver. Generally, these problems are handled by guaranteeing interpretations only on restricted subsets, such as "ASCII standard characters," or even "upper case characters."

C. Transport

An important lesson about transport systems has emerged from work on HLP's. No single abstract view of the transport facility applies to *all* HLP's; different protocols need access to transport facilities of different types. Transport facilities should be designed in a modular way to provide different *types of service* to different HLP's. The graphics and voice protocols described in this paper illustrate the range of transport services required. The NGP depends on orderly transmission of streams of bytes, and consequently requires many supporting transport mechanisms offered by IPC protocols [1], [2], [37], [38]: connections, flow control, sorting out-of-order packets, duplicate packet detection, and loss prevention. Small sacrifices in bandwidth or irregularities in transmission delay are unimportant. By contrast, the NVP requires a stable transmission delay, but can tolerate errors, lost packets, or packets discarded due to local network congestion.

Other HLP's may fall at points in between. For example, some do not need to establish network "connections," but can be designed around inquiry–response exchanges, sometimes called DATAGRAMS [26]. A protocol to look up telephone numbers might simply address a single message containing a person's name to the retrieval system; it expects a single answering message in response. If the sender receives no answer, he can repeat the request. A surprisingly large variety of protocols can be cast in this framework.

The grain of communication required of the transport system also varies among HLP's. Some will want to transmit streams of bits or bytes; others may compose indivisible messages to be delivered as atomic entities. In implementing a byte stream in a packet switching network, the transport system will attempt to buffer bytes until a message fills up. But how often should packets be sent? If a full message is assembled, presumably it should be sent immediately, subject only to flow control practices in the network. But what if a message is only partially filled, and has for several seconds seen no data added to it? Should it be sent, even though only partially full? Perhaps the receiving process is idle, and could begin working on the new data. Or perhaps the protocol is being used as part of an interactive dialog, and the user needs to see the effect of the data contained in the partial message. Or perhaps the message is partially full only because it is the last message of a long transmission (e.g., when transmitting an entire file).

The question used to resolve these questions is: on what unit of information can the receiving process take significant action? If we want the receiver to act each time a new byte is generated by the sender, we must ensure that no encoded information is buffered at the sender for very long, i.e., that partially filled messages are transmitted at short intervals. If, on the other hand, we are concerned only with finishing a long transmission as quickly as possible, using full-size packets for efficiency, we can expect the sender to indicate that transmission of the very last (partially filled) packet is required even though it's not full.

The transmission strategy used to make these decisions is determined in part by application process. Two facilities should be provided.

1) The sending process states a time interval that is the maximum time any chunk of information is buffered before sending (assuming network flow control conventions allow transmission). This time may be very long if timely transmission is unimportant, as in a file transfer, or it may be related to the amount of data accumulated for transmission [3]. (Note that such a scheme is an essential aspect of "terminal output"—characters must be periodically transmitted for the

user to see. Unfortunately, many operating systems that provide interfaces to network transport mechanisms provide this timely transmission facility *only* for terminal output; they do not implement the technique for other HLP network traffic.)

2) The sending process notifies the transport mechanism when it is important that buffered information actually be transmitted.

The needs of different HLP's are sufficiently varied to induce different behaviors in the way transport is provided: messages with time criticality of 100 ms can be treated differently than messages with 1-min delivery requirements. An HLP should make known to the transport system the *type of service* it desires. For example, NVP communications will request uniform delay. Interactive streams desire short delays, modest bandwidths, and high reliability. Transmitting a file requires high bandwidth, but can tolerate long transmission delays. In this case, packets could be routed far afield, taking advantage of idle communications lines, incurring long delays and numerous hops, but still provide large bandwidth because many packets are taking different long routes concurrently.

Most of the issues described in this section concern the implementation of transport facilities in an operating system and not the design of a communications network. As a consequence of layering, important issues in network design (naming, addressing, message sizes, internetwork operation, routing) rarely influence HLP design. An exception is our plea that the communication system respond to "type of service" requests from applications.

In summary, we believe 1) no single transport facility is universal, 2) transport systems that are assembling messages from smaller items need to transmit partial messages in response to time constraints and precise instructions from the sender, and 3) the transport needs of the application should be specified to the packet-switching network in order to provide the type of service desired.

D. Implementation

For HLP's to be useful in a resource-sharing network, implementations must be devised for many of the computers in the network. The ease with which these implementations can be constructed is often a major goal of an HLP design. Although most HLP's could be implemented easily, programming environments of many computers unfortunately complicate implementations. Many operating systems incorporate network access as an afterthought and therefore provide cumbersome facilities. Many have poor support for cooperating processes, which are used frequently in HLP implementations. And finally, tools for debugging asynchronous processes are usually poor. In a network, the problem is exacerbated by the inability to examine the state of the cooperating process or to repeat the conditions that caused an error. Even stopping the program being tested may be difficult: the other process may detect prolonged inactivity and abort!

Implementation and debugging can be aided by measures taken in the design of the HLP. Commands to reset both processes to an initial state are essential; facilities to report errors precisely can be provided. Sometimes explicit debugging aids can be built into an HLP implementation—a version of the NGP display process that types out a trace of all protocol commands received and sent greatly speeds debugging. Layering of protocols also eases implementation. If an HLP is implemented on top of a transport system deemed to transmit streams of data reliably, many of the possibilities for timing-related errors are removed from the HLP.

IV. High-Level Protocols and High-Level Languages

The finale of this survey of issues surrounding the design and implementation of HLP's relates the structure of HLP's to that of HLL's, and suggests techniques used in HLL's that can be applied fruitfully to problems in HLP's. In addition, HLP's point to some areas that should receive increased attention in HLL development.

A. Why an HLP Protocol Is Like An HLL

HLL's and HLP's are both abstract, but precise, definitions of a computation or communication that can be translated into the primitive operations actually provided by a computer or communication system. HLL's were born of a desire to express algorithms in a language natural to the application; a compiler fills in many details as it translates the program into the machine language of a particular computer. Similarly, an HLP is most clearly expressed in a "language" that relates closely to the application; the tedious details of coding and transport are not the primary concerns of the HLP designer and implementer.

One of the powerful ideas in HLL's is the concept of *type*: a method for defining abstract objects such as integers, characters, arrays, lists, text strings, records, sets, etc. A compiler, together with *type definitions*, chooses a way to represent these objects within the physical memory structure of the computer. With this aid, the programmer is freed from the details of encoding various objects in memory, and concentrates on harnessing these objects to ease the construction of his program.

The concept of type is just as prevalent in HLP's. Usually a message is divided into "fields," sequences of bits that are assigned particular meanings. But to the HLP designer, these fields are simply encodings of types used in the application program: integers, characters, etc.

Data structure declarations thus play a key role in both settings. They structure memory for the HLL; they structure messages for the HLP. A concrete example from the plotter protocol will further illustrate this point. The protocol is a sequence of *commands* (a type); each command specifies a *function* (a type) and possibly some *arguments* (of various types). Using PASCAL [17], we would write:

```
type PlotterFunction = (fBeginPlot,fEndPlot,fUp,fDown,
      fLineTo);
type PlotterCommand = record
  case function: PlotterFunction of
    fBeginPlot: () ;
    fEndPlot: () ;
    fUp: () ;
    fDown: () ;
    fLineTo: (x,y : int)
  end;
type PlotterProtocol = file of PlotterCommand
```

PlotterFunction is a type that can take on five values corresponding to the names fBeginPlot, fEndPlot, fUp, fDown, and fLineTo. The key type is PlotterCommand, which is a *variant record* with an entry giving the function and possible arguments—it thus specifies the structure of a single command

in the plotter HLP. The type PlotterProtocol is a sequence of such commands.

The application program might generate the plotter HLP by calling five procedures corresponding to the five commands. For example, the procedure LineTo builds a PlotterCommand record consisting of the function name (fLineTo) and two argument values. The record is then "transmitted" by appending it to the PlotterProtocol sequence:

```
var c : PlotterCommand;
var pp : PlotterProtocol;

procedure LineTo (tx, ty : int );
  begin
    c.function := fLineTo; (* Build command *)
    c.x := tx; (* Insert arguments *)
    c.y := ty;
    pp↑ := c; put(pp); (*Add command to sequence *)
  end
```

Notice how type declarations and record-construction statements help specify coding and transport details.

The paradigm illustrated above is a common one in HLP's: transmitting records consisting of a command, followed by values of arguments. It is no coincidence that these records correspond to the specifications of typed objects in a procedure call; the declaration of the procedure LineTo and of the fLineTo variant of the PlotterCommand record are strikingly similar. The plotter protocol and the NGP, which use this paradigm, are essentially transmitting procedure names and arguments to the receiving process, which calls corresponding procedures. Both protocols could be expressed in terms of a "procedure call protocol," an HLP that arranges to achieve this effect [39].[2] HLP's with this flavor arise quite frequently when a single application program consisting of several modules is distributed in a network. It is precisely this methodology that we advocated for developing the NGP.

Not all HLP designs are conveniently expressed within a framework as specialized as a procedure call protocol. The NVP, for example, does not fit well into this category. However, the concept of structuring communication into records by using appropriate type declarations remains extremely valuable. These type declarations must of course, be shared by the sender and receiver to insure both parties apply the same interpretations to messages.

Although the idea of *type* is perhaps the most important, other aspects of HLL's correspond to HLP features. When two modules are *linked* together, communication methods are established between the modules (usually through procedure linkages), much the way a connection establishes a communication between two processes. The act of *binding* the two modules together might be compared to the negotiation or inquiry used to inaugurate communication, though most HLL binding is considerably less flexible than HLP parametrization. The *data abstraction* techniques in HLL's foster methods for designing software systems that are similar to the abstractions required to achieve device independence or to define standards. Finally, HLL's are increasingly concerned with methods for specifying precisely the interface to software modules, in part so that uses of the module by other programs can be checked carefully. This problem of precise description is also key to HLP's; an accurate specification of the HLP for a service must be available to any programmer writing programs that use the HLP.

[2] Strictly speaking, a "procedure call" requires that control not return to the caller until the procedure has been executed. In the plotter and NGP protocols, it is desirable to let the calling process continue execution so that several "procedure call" records can be packed together into one network message. If the procedure being called returns an answer, the calling process must of course wait for the answer.

B. How HLL's Can Help HLP's

Our sketch of the relationship between HLL's and HLP's shows that many ideas developed for HLL's address corresponding needs in HLP's. However, to be fully useful in implementing HLP's, some of these ideas must be developed further.

1) Type definitions: Currently, HLL's use type definition facilities that are too loose and dependent on details of the computer on which they execute. For example, an "integer" will designate 36 bits on some machines, 32 on others, 16 on still others, etc. This definition is too vague to be useful in a heterogeneous network, i.e., one containing computers of many different sorts. Instead, we need to define integers precisely, e.g., "24-bit 2's complement integer." Also, most structural definition facilities will not allow us to split a field over a word boundary in the computer. Such a facility may be needed when interpreting a standard protocol, even though it might be folly within any given machine. Thus programming-language techniques for defining structure on word-oriented storage should be extended and applied to defining structure in bit-oriented protocol streams.

2) Interprocess communication (IPC): Many newer HLL's are beginning to offer facilities for IPC, some using the idea of transmitting typed records as we showed above [20]. However, none provides automatically the vital link between the variant record and the procedure call. The programmer can write the obvious encoder, as demonstrated in the PASCAL example, and a matching decoder that collects arguments and dispatches to call the proper procedure, but these are bulky and tedious to write. This sort of IPC is common in networks, but is neglected in programming systems. Some efforts have been devoted to this problem [11], [16], but the need has not been widely recognized.

3) Linking: HLL's must be prepared to check types used in IPC when connections are made. Ideally, a process offering an HLP service could distribute (in a standard form!) a definition of the protocol to receive service. Such a definition could be checked against the protocol definition of a potential partner when a connection is initiated—this is similar to checking caller/callee parameter declarations when linking separately compiled programs; we insist that definitions match before we allow communication.

4) Asynchrony: We need better ways to document message sequences and message exchanges. As in any multiprocessing system, we must worry about deadlock and synchronization issues. They may be especially tricky if we use a transport mechanism that sometimes delivers messages out of sequence or occasionally loses them. Assertions provided with the program, and flow analysis undertaken by a compiler, can help with certain communications problems: can a particular message be packed with others to form a larger package to transport through the network, or must it be sent immediately because a reply is required? Efforts in this area are beginning [8], [34].

5) Documentation: There is also a need for a definition at a more functional level: What are the services of the protocol?

What effect does each command have? How do the commands interact? What are possible responses to particular messages? It seems clear that a computer program, perhaps expressed as abstract operations of a process that implements the protocol, is a better documentation tool than is prose. The data abstraction languages now emerging [15], [21], [30], are working toward such definitions. Even using existing HLL's, a simple exemplary implementation of the NGP might be a much better definition of its properties than is a bulky document.

Ideally, these goals can be addressed in an integrated fashion. A new project of Feldman, Low, and Rovner [9], [10], tackles some of them with the programming language PLITS, which integrates messages and message structures into an HLL and a distributed environment DSYS, which provides the binding and debugging facilities needed to develop applications that use several communicating processes.

Developments that truly integrate protocols into programming languages and tackle interprocess communication among processes written in different programming languages would open up many new opportunities for users of HLP's. Programs using the protocols would be easier to implement and easier to change. The protocols themselves could be changed more readily, and could be described by precise definitions as well as lucid implementations.

V. Conclusion

The emergence of resource-sharing networks and the HLP's that exploit them has aroused public interest in standards. Although a standard protocol is substantially harder to design than is a set of private communication conventions, the reward is enormous: the ability to communicate effectively with a vast number of computer resources. The desire to generalize an HLP—to increase its scope to handle as many uses as possible—is countered by increased difficulty in achieving a standard. The search for simple effective standard HLP's is still in its infancy—we can expect that HLP development will continue, and can hope that further principles and methodologies will emerge to help design these protocols.

Although they have received increased attention because of the growth of computer networks, problems of standardization and generalization are not intrinsic to networks. In fact, the presence of a network often confuses the job of designing a framework to provide service to an application with the job of transmitting information in a network. Breaking the problem into issues of language, coding, and transport is a help. More importantly, the application must be understood well in simple environments, free from standardization constraints, before a standard protocol design is attempted.

The outlook for streamlining specification and implementation of HLP's is closely tied to the development of HLL's. Existing HLL's provide facilities that can be applied to simplify these tasks. The important next step is for the language designer to realize that programs written in his language will need to communicate with other programs written in other languages, on computers with different conventions. Protocol definitions must be shared among these environments in order to propagate standard protocols. Although it is a difficult goal to standardize even a simple form of interprocess communication over a wide variety of computing equipment and programming systems, ignoring this opportunity will curb the potential of resource-sharing networks. Ultimately, HLL's could be enhanced to the point that techniques used to implement HLP's will be expressed as naturally as arithmetic expressions are now.

Acknowledgment

The authors are grateful to V. Cerf, B. Kahn, and J. Shoch for helpful comments on drafts of this paper.

References

[1] C. S. Carr, S. D. Crocker, and V. G. Cerf, "Host-host communication protocol in the ARPA network," in *AFIPS Proc.* (Spring Joint Computer Conf.), pp. 589–597, 1970.
[2] V. G. Cerf and R. E. Kahn, "A protocol for packet network interconnection," *IEEE Trans. Commun.*, vol. COM-22, p. 637, May 1974.
[3] D. Cohen, "Issues in transnet packetized voice communication," in *Proc. 5th Data Communications Symp.* (Snowbird, UT), pp. 6, 10–13, Sept. 1977, IEEE Catalog Number 77CH1260-9C.
[4] —, "Specification for the network voice protocol," USC/Information Sciences Inst., Marina del Rey, CA, ISI/RR-75-39, DDC AD A023506, Mar. 1976.
[5] —, "A protocol for packet switching voice communication," in *Computer Network Protocols*, A. Danthine, Ed. Liege, Belgium, Feb. 1978.
[6] W. R. Crowther, F. E. Heart, A. A. McKenzie, J. M. McQuillan, and D. C. Walden, "Issues in packet switching network design," in *AFIPS Proc.* (National Computer Conf.), p. 161, 1975.
[7] J. Davidson, W. Hathaway, J. Postel, N. Mimno, R. Thomas, and D. Walden, "The ARPANet Telnet protocol: its purpose; principles, implementation and impact on host operating system design," in *Proc. 5th Data Communications Symp.* Snowbird, UT, Sept. 1977.
[8] J. A. Feldman, "Synchronizing distant processes," Computer Science Dept., Univ. Rochester, TR 26, Oct. 1977.
[9] —, "A programming methodology for distributed computing (among other things)," Computer Science Dep., Univ. Rochester, TR 9, Sept. 1976.
[10] J. A. Feldman, J. R. Low, and P. D. Rovner, "Programming distributed systems," *Computer Science Res. Rev.*, Univ. of Rochester, 1977–78.
[11] J. A. Feldman and R. F. Sproull, "System support for the Stanford hand-eye project," in *Proc. 2nd Int. Joint Conf. on Artificial Intelligence*, 1971.
[12] J. L. Flanagan, *Speech Analysis, Synthesis and Perception*. New York: Springer, 1972.
[13] J. W. Forgie, "Speech transmission in packet-switched store-and-forward networks," in *AFIPS Proc.* (National Computer Conf.), p. 137, 1975.
[14] D. Gojanovic, "Some experience with network graphics protocols," in *Computer Network Protocols*, A. Danthine, Ed. Liege, Belgium, Feb. 1978.
[15] A. Goldberg and A. C. Kay, *Smalltalk-72 Instruction Manual*, Xerox Palo Alto Research Center, SSL-76-6, 1976.
[16] G. Hamlin, "Configurable applications for satellite graphics," Ph.D. dissertation, Univ. of North Carolina, Chapel Hill, 1975.
[17] K. Jensen and N. Wirth, "PASCAL, user manual and report," in *Lecture Notes in Computer Science*, G. Goos and J. Hartmanis, Eds. New York: Springer, vol. 18, 1974.
[18] R. E. Kahn, "Resource-sharing computer communications networks," *Proc. IEEE*, vol. 60, pp. 1397–1407, Nov. 1972.
[19] —, "The organization of computer resources with a packet radio network," *IEEE Trans. Commun.*, vol. COM-25, pp. 169–178, Jan. 1977.
[20] B. Lampson, J. Mitchell, and E. Satterthwaite, "On the transfer of control between contexts," in *Lecture Notes in Computer Science*, G. Goos and J. Hartmanis, Eds. New York: Springer, 1974, vol. 19, p. 181.
[21] B. Liskov, A. Snyder, R. Atkinson, and C. Schaffert, "Abstraction mechanisms in CLU," *CACM*, vol. 20, no. 8, p. 564, Aug. 1977.
[22] J. Maleson, J. Nabielsky, and R. Rashid, "Rochester image protocol," Computer Science Dep., Univ. Rochester, informal note.
[23] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*. New York: Springer, 1976.
[24] W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*. New York: McGraw-Hill, 1973.
[25] —, "An approach to graphics system design," *Proc. IEEE*, vol. 62, p. 471, Apr. 1974.
[26] L. Pouzin and H. Zimmerman, "A tutorial on protocols," this issue, pp. 1346–1370.
[27] D. R. Reddy, "Speech recognition by machine: A review," *Proc. IEEE*, vol. 64, pp. 501–531, Apr. 1976.
[28] D. M. Ritchie and K. Thompson, "The UNIX time-sharing system," *CACM*, vol. 17, no. 7, pp. 365–375, July 1974.

[29] L. G. Roberts and B. D. Wessler, "Computer network development to achieve resource sharing," in *AFIPS SJCC Proc.*, vol. 36, p. 543, May 1970.
[30] M. Shaw, W. A. Wulf and R. L. London, "Abstraction and verification in Alphard: Defining and specifying iteration and generators," *CACM*, vol. 20, no. 8, p. 553, Aug. 1977.
[31] R. F. Sproull, "Omnigraph-Simple terminal-independent graphics software," Xerox Palo Alto Res. Center, CSL-73-4, 1973.
[32] —, "InterLisp display primitives," Xerox Palo Alto Res. Center, 1977, informal note.
[33] R. F. Sproull and E. L. Thomas, "A network graphics protocol," *Comput. Graphics*, vol. 8, no. 3, Fall 1974.
[34] C. A. Sunshine, "Survey of protocol definition and verification techniques," in *Computer Network Protocols*, A. Danthine, Ed. Liege, Belgium, Feb. 1978.
[35] W. Teitelman, "A display oriented programmer's assistant," Xerox Palo Alto Res. Center, CSL-77-3, 1977.
[36] R. H. Thomas, "A resource sharing executive for the ARPAnet," in *AFIPS Proc.*, NCC, p. 155, 1973.
[37] —, "MSG: The interprocess communication facility for the national software works," Bolt Beranek and Newman, Rep. 3483.
[38] D. C. Walden, "A system for interprocess communication in a resource-sharing computer network," *CACM*, vol. 15, no. 4, p. 221, Apr. 1972.
[39] J. E. White, "A high-level framework for network-based resource sharing," *AFIPS Proc.*, NCC, p. 561, 1976.
[40] P. A. Woodsford, "The design and implementation of the GINO 3D graphics software package," *Software Practice and Experience*, vol. 1, p. 335, Oct. 1971.

Tutorial Survey of Algorithms For Locating and Identifying Spatially Distributed Sources and Receivers

M. Morf, B. Friedlander and J. Newkirk
Information Systems Laboratory
Stanford University
Stanford, CA 94305

We present a short tutorial survey of algorithms for locating and identifying spatially distributed sources and receivers. The emphasis is on methods that are either considered to be very basic or lend themselves potentially to distributed computations, the main objective of this work. We shall also very briefly outline our own approach to this set of problems.

1. Introduction

In many practical problems it is necessary to determine the location of signal (noise) sources from measurements provided by one or more sensors. Typical applications include:

- Acoustic surveillance systems (e.g., sonar detection of low flying aircraft),
- Seismic arrays for seismic exploration, monitoring earthquakes and nuclear explosions, or detecting vehicle movements,
- Antenna arrays for radio astronomy or electronic surveillance (e.g., direction finding),
- Multiple radar systems for detection and tracking.

The diversity of applications involving the target location problem makes a general unified treatment of this subject quite difficult. To provide some focus for our discussion we will use the following sample problem:

> Consider a small number of sensor sites (perhaps ten) distributed over a specified area. A number of targets are present in the area and their location is to be estimated based on the data collected by the sensors. The sensors measure signals which are either emitted by the target (the passive case) or reflected by it (the active case which requires target illumination). By processing the signals provided by the sensor, information about target bearing and/or range can be determined.

Sometimes a single sensor is not capable of measuring either range or bearing, as for example with omnidirectional passive sensors. Combining data from a group or array of sensors, however, makes it possible to find the desired information. A sensor array of this type may be located at a single site, in which case we consider it as one unit, or it may be distributed among many sites.

In other cases the sensor sites can provide different types of target related data, in particular:

(i) Range only (ranging radar, active sonar)

(ii) Bearing only (optical, infrared sensor, direction finder)

(iii) Bearing and range (search, tracking radar)

(iv) Target velocity (Doppler radars, MTI)

Different data types lead to different location estimation techniques. For example, range only or bearing only measurements are related to target association techniques (section 2.2). Bearing and range data is usually associated with tracking algorithms for moving targets.

Data from a single omnidirectional passive sensor is treated by time-of-arrival methods (section 2.1) or beamforming and array processing techniques (section 2.4). The estimation method also depends on the type of signals provided by the sensor site: coherent/noncoherent, "raw" or filtered data (linear processing), data after detection (nonlinear processing), etc.

Classical methods of processing sensor data have generally been of the centralized type, that is, all of the sensor data was collected at one site and then processed. An alternative is to process much of the data at the collection site and to send only the relevant data to either a central site or (more generally) to the appropriate user. In Section 3 we shall discuss the different types of distributed processing and their advantages.

In the last section we will describe our own approach to the development of distributed algorithms for the estimation of position, location and other characteristics of sensors and sources, (Morf et al.). We will give a short description of sample algorithms of a fully distributed nature that have desirable features. We shall also outline several of the nonclassical approaches to solving these problems.

2.0 Techniques for estimating target location

This section provides a brief summary of the solution techniques associated with the target

This work was supported by ARPA under contract MDA 903-78-C-0179.

location problem. Only the basic ideas are presented; the details can be found in the references.

2.1 Time-of-Arrival Estimation

A signal emanating from a remote source and measured in the presence of noise at two spatially separated sensors can be modeled as

$$x_1(t)=s(t)+n_1(t) \tag{1a}$$

$$x_2(t)=As(t+D)+n_2(t), \tag{1b}$$

where $s(t)$, $n_1(t)$, $n_2(t)$ are assumed to be stationary, independent, random processes. One common method of estimating the time delay, D, is to compute the cross correlation function

$$R_{x_1x_2}=E\{x_1(t)x_2(t-\tau)\} \tag{2}$$

It follows directly that

$$R_{x_1x_2}(\tau)=AR_{ss}(\tau-D) \tag{3}$$

where $R_{ss}(\cdot)$ is the signal autocorrelation function. An important property of autocorrelation functions is that $R_{ss}(\tau)\le R_{ss}(0)$. Thus, the peak of $R_{ss}(\tau-D)$ will occur at $\tau=D$. This provides us with a way for finding the delay by calculating the estimated cross-correlation function. If $s(t)$ is a white noise source, $R_{ss}(\tau-D)=\delta(\tau-D)$, and the peak will be sharply defined. In general, $R_{ss}(\cdot)$ will be "spread out" which tends to broaden the peak, making it more difficult to pinpoint the actual delay. Furthermore, when multiple targets (and multiple delays) are present, the "tails" of the autocorrelation functions for different targets will be overlayed and more difficult to separate. Thus, it is desirable to preprocess the sensor measurements x_1, x_2 so that after crosscorrelation sharper peaks will result, as in Fig. 1. In the absence of measurement noise this can be done by passing $x_1(t)$, $x_2(t)$ through a "whitening" filter for $s(t)$, hence the correlation of s is removed. When noise is present, the filter has to take into account both signal and noise spectra.

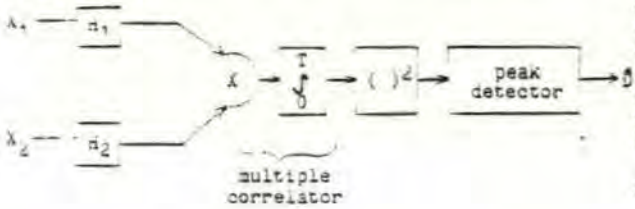


Fig. 1: A time-delay estimator.

Different choices for the pre-filter are possible depending upon the performance criterion chosen by the designer: the likelihood function [Hann and Tretter], the deflection function [Knapp and Carter], etc. It should be noted that several estimator structures besides the multiplier-correlator estimator have been developed.

All of the information about the target location are encoded in the relative time-delays of the various sensors. To see this consider the following:

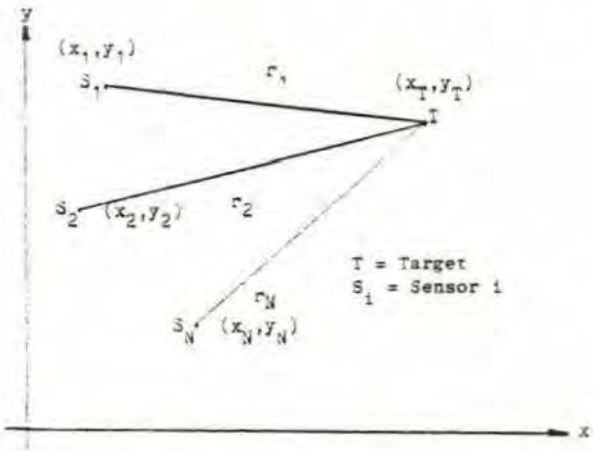


Fig. 2: Target-sensor geometry

$$D_{ij}=\frac{r_i-r_j}{c}, \tag{4}$$

where D_{ij} is the relative delay between sensor i and sensor j, and c is the propagation velocity.

$$r_i^2=(x_T-x_i)^2+(y_T-y_i)^2, i=1,2,\ldots N. \tag{5}$$

It can be shown [Schmidt] that for $N\ge3$, the set of equations (4),(5) can be rewritten as a <u>linear</u> set of equations for x_T, y_T, where the coefficients are known quantities (i.e. written in terms of D_{ij}, x_i, y_i). This set of equations can now be solved to determine the target location.

The discussion above indicates that one way of solving the target location problem is to first estimate the time-of-arrival delays and then to compute the location based on the geometry of the problem [Hann]. It is possible, of course, to combine these two steps and develop an estimator directly for the target coordinates (x_T, y_T) or, as is more commonly done, for its bearing and range. This leads to alternative estimator structures, typically using the maximum likelihood approach [Bangs and Schultheiss], [MacDonald].

2.2 Target Association Techniques

A special type of problem arises when multiple sensors which measure range but not azimuth (or vice versa) are used to estimate target location. If only a single target is present, its location is found by multilateration. For example, if azimuth measurements from several sensors are available, one has only to compute the intersection of the various lines-of-sight.

The situation becomes more complex when multiple targets are present. This is illustrated by Fig. 3.

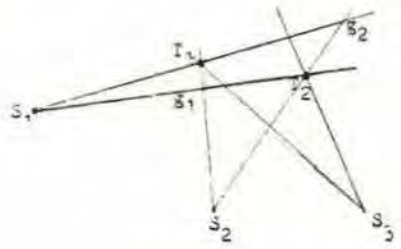


Fig 3: The target association problem

Each sensor is assumed to have detected the two targets T_1, T_2. However, these detections are not properly associated. It is not known which measurement of each sensor corresponds to which target. Thus, it is necessary to associate targets with sensor measurements before estimating the target locations (in fact the association and location problems are addressed simultaneously). Note also that if there are more targets than sensors, ambiguities ("ghost" targets) may result. In Fig. 3, if S_3 was not there, the measurements of S_1 and S_2 would be consistent with the assumption that the targets are at S_1, S_2 rather than at T_1, T_2.

Several schemes have been proposed to solve the target association problem, and they are briefly described below.

List Forming

Pick a pair of sensors and compute all the intersections of their lines-of-sight to potential targets (i.e. directions in which they detected something). These intersection points are potential target locations. Now pick a third sensor and check whether its lines-of-sight pass through any of the intersection points. If not, delete these points from the list of potential targets. By proceeding this way with the other sensors, the list will finally include only those target locations which are consistent with all the observations. It should be emphasized that this is a highly simplified description of more realistic list forming algorithms.

Back-projection or Space-Search

The space to be searched is divided into cells of a size corresponding to the system resolution. The number 1 is added to those cells of the space which lie along the line of sight of a given sensor detection. This process is repeated for all lines-of-sight of all sensors. As can be seen from Fig. 4 the target locations can be identified as those having the highest number (= the number of sensors) written in them. Note that relatively high values can be obtained at locations other than those of the real target ("ghost" locations).

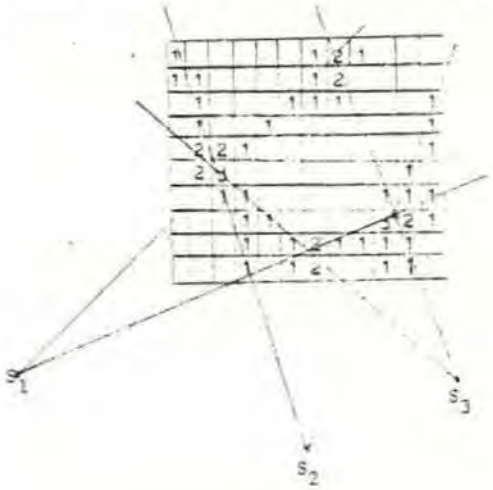


Fig. 4: Association by back-projection

Image reconstruction techniques

It is possible to view the sensor measurements as line integrals through an image consisting of "points of light" at the target locations. The line integrals are over the infrared emissivity map (for IR sensors) or the radar reflectivity map (for ranging radars). The problem of reconstructing images from their line-integral projections has been extensively treated in literature [e.g., Brooks and Di Chiro, Horn]. Recently it was shown how these techniques can be applied to the target association problem [Friedlander et al., Denton et al.] by reconstructing the "brightness map" of the area under surveillance and identifying targets as the "bright spots". It should be noted that the image reconstruction method requires that the sensors provide the actual energy measured in each direction (range) and not just target/no target information. The detection takes place after processing the information from all the sensors; in list forming and space-search, only the results of the detection performed at each individual sensor are passed on.

2.3 Spectral estimation

Multiple-sensor measurements can be considered as samples of a time-space function $y(t,\underline{\rho})$ where $\underline{\rho}$ represents a point in 3-D space. The notions of (temporal) correlation function and (temporal) spectral density can be extended to time-space functions of this type. We define a random field $y(t,\underline{\rho})$ as stationary and homogeneous if $E\{y(t,\underline{\rho})\}=0$ and

$$R(t,t',\underline{\rho},\underline{\rho}') = E\{y(t,\underline{\rho})y(t',\underline{\rho}')\}$$

$$= R(\tau,\underline{r}) \qquad (6)$$

where

$$\tau = t - t', \quad \underline{r} = \underline{\rho} - \underline{\rho}'$$

Any homogeneous random field has a spectral representation

$$y(t,\underline{\rho}) = \int_{-\infty}^{\infty}\iiint e^{j(\omega t + \underline{\kappa}\cdot\underline{\rho})} Z(d\omega, d\underline{\kappa}), \qquad (7)$$

where $\underline{\kappa} = (\kappa_1, \kappa_2, \kappa_3)$, $\underline{\rho} = (x,y,z)$ and $Z(\omega, \underline{\kappa})$ is a random function with certain properties. The correlation function can be represented by

$$R(\tau,\underline{r}) = \int_{-\infty}^{\infty}\iiint e^{j(\omega\tau + \underline{\kappa}\cdot\underline{r})} P(\omega,\underline{\kappa})\, d\omega\, d\underline{\kappa} \qquad (8)$$

where P is the spatial-temporal spectral density. As in the temporal case, an inversion formula holds:

$$P(\omega,\underline{\kappa}) = \frac{1}{(2\pi)^4}\int_{-\infty}^{\infty}\iiint e^{-j(\omega\tau + \underline{\kappa}\cdot\underline{r})} R(\tau,\underline{r})\, d\tau\, d\underline{r}. \qquad (9)$$

As an important example, consider a monochromatic (single frequency) plane wave at temporal frequency ω_0 propagating in the direction given by a unit vector $\hat{\kappa}$ at velocity c. Such a wave is represented by

$$y(t,\underline{\rho}) = \exp(j\omega_0 t + \underline{\kappa}_0\cdot\underline{\rho}) \qquad (10a)$$

where

$$\underline{\kappa}_0 = \frac{\omega_0}{c}\hat{\kappa}. \qquad (10b)$$

It is easy to verify that for this space-time function we have

$$R(\tau,\underline{r}) = e^{-j(\omega_0\tau + \underline{\kappa}_0\cdot\underline{r})} \qquad (11a)$$

and

$$P(\omega,\underline{\kappa}) = \delta(\omega-\omega_0, \underline{\kappa}-\underline{\kappa}_0) \qquad (11b)$$

which is a delta function located at temporal frequency ω_0 and spatial frequency (or wave number) $\underline{\kappa}_0$. This example indicated how $P(\omega,\underline{\kappa})$ provides information regarding the direction (and velocity) of propagation of waves.

The point of this discussion is that (spatial) spectral estimation is a way of estimating target bearing, since if we plot $P(\omega,\underline{\kappa})$ in the $\underline{\kappa}$ plane (for a fixed ω), it will tend to be concentrated around the point $\underline{\kappa}$ which corresponds to the direction of the wave propagation and hence the bearing of the target (the source of these waves). Thus, target bearing estimation reduces to the problem of estimating $P(\omega,\underline{\kappa})$ from the measurements $y(t_i,\underline{r}_j)$, where $\underline{r}_j$ represents sensor locations and t_i are the sampling times of the output of that sensor.

Many spectral estimation techniques have been used in this context; several are described in the references.

2.4 Beamforming and Array Processing

Perhaps the most common operation in processing signals in a sensor array is that of beamforming. Antenna arrays (for radar, communication, etc.) and acoustic sensor arrays (sonar) are typical examples of beamforming. Beamforming consists of a summation of time-delayed (or phase-shifted) versions of the sensor outputs, i.e.

$$Z(t) = \sum_{i=1}^{N} y(t-t_i, \underline{\rho}_i), \qquad (12)$$

where t_i represents the time-delay for sensor i and $\underline{\rho}_i$ its location. Proper choice of the delays t_i enhances the signals received from a particular direction and attenuates signals from other directions. This operation is the spatial equivalent of a temporal narrow bandpass filter.

The output Z(t) of the beamformer is a (scalar) time function, which is processed so as to obtain a measure of the signal energy in an optimal way. The most common processing schemes include

Matched filtering
Wiener filtering or least-squares estimation
Maximum likelihood estimation

This linear filtering is often followed by a nonlinear operation, e.g. squaring and integration. The order of linear filtering and transforming (also a linear operation) is often reversed, the exact structure depending on the application. The implementation of these processes is usually done in the fourier domain (with phase shifts replacing time delays) but time domain implementations are also used. A sample of the vast literature on beamforming and the associated signal processing (referred to as "array processing") is given in the bibliography.

A class of array processor of particular interest are the different types of adaptive arrays. The need for adaptive arrays arises for many reasons. Some examples:

- Null steering, to minimize interference from sources other than the target of interest.
- Adaptive filtering, to handle unknown noise and signal statistics.
- Adaptive beamforming. Beamforming requires precise knowledge of the sensor locations (within fractions of a wavelength) in order

that the steering delays be computed. Relatively small errors can lead to serious performance degradation. Thus, when sensor locations are imprecisely known or are constantly changing, a fixed processing scheme is infeasible.

3. The Need for Distributed Computation

There are many advantages in distributing computations for a large sensor network; some of the main arguments are the following:

(1) Reduction of Computational Complexity

Distributed processing is often used as a means for solving problems related to large-scale systems. This approch leads to the decomposition of a high-dimensional problem into a sequence of smaller-dimensional ones. This often results in considerable computational savings; many fast algorithms, such as the Fast Fourier Transform, are of this type. Also, certain large-scale problems simply cannot be solved in a direct manner (e.g., inversion of very large matrices) and ways have to be found to decompose the problem into smaller parts that can be handled. This can, of course, be done in a centralized manner, but the distributed approach often leads to natural decompositions and valuable insights.

(2) Reliability

Distributed systems have good properties from a reliability standpoint due to their inherent parallelism. Failure of a computational module does not necessarily result in system failure since the computational load can be redistributed among the remaining modules. Thus, a distributed system may have the ability to reconfigure and continue operation. Depending on the type of the system and its structure, its operation after reconfiguration may be at a reduced performance level. (This would be the case if the remaining computational resources were insufficient to complete the solution of the problem, or if the loss of a computational module was associated with the loss of a sensor site.) The system displays graceful degradation of performance, which contrasts with the "catastrophic" failure mode of centralized systems.

(3) Flexibility

The distributed nature of computations is often associated with distributed system structures; such a system structure might be a collection of sensor/computer/communication modules interconnected in a network. This organization leads to a very flexible structure, possessing desirable properties which are not always present in a centralized system:

- Easy system growth
- The capcity to handle topological changes in the system structure (e.g., adding or deleting nodes during maintainance without interrupting system operation).
- The possibility of incorporating many combinations of resources, with variable performance levels, as determined by the needs of each user.
- The combination, in a single network, of many types of information sensors.

4. Our Approaches

Distributed processing has by now become a term that is applied to many types of systems and is not very well defined. Since the distributed sensor net problem can be well described, see e.g. [ISL DSN Report], we shall use it as a basis for defining distributed processing. We consider three computational organizations which could be used in this context: centralized, independently distributed, and cooperatively distributed.

1) Centralized -- all sensor data is passed to a central site, where computation is performed, and the pertinent results are then returned to the appropriate remote sites.

2) Independent -- all sensor information is communicated to every other site, and each site then makes its best estimate of the environment.

3) Cooperative -- sites exchange processed information, and at most partial sensor data.

It is the second and third organizations that are normally referred to as distributed organizations, and the third, in particular, that we consider to be of greatest interest.

The Search for Distributed Algorithms

Centralized algorithms are now quite well understood, as a perusal of the extensive literature indicates. However, very few attempts have been made to unify and integrate all these different approaches and results. A typical book on radar or sonar signal processing is a rather ad hoc collection of data, methods and theory (reminiscent of a cook-book). To an outsider of this field it is extremely difficult to get a coherent picture and to make intelligent choices in applying these methods in the design of systems. A systematic representation of this knowlege, by itself a tremendous task, is required in order to make effective use of the available alternatives. It is very tempting to suggest the development of an "expert support system" combining and extending recent approaches in AI, data-base management and related fields.

Our approach to the development of (cooperatively) distributed algorithms can be summarized under the following headings:

1. Partitioning of optimal centralized algorithms, such as Maximum-Likelihood, Extended Kalman Filtering, and Beam Forming. This approach is useful when the system under consideration can be

divided into subsystems with sparse interaction.

2. Application and extension of methods developed in the context of decentralized control and estimation, e.g. team decision theory and differential games, hierarchical and multilevel systems, aggregation methods, singular perturbation and other perturbation techniques, periodic coordination and spatial dynamic programming (sdp).

3. Development of new optimal distributed algoritms for specific subsets of sensor data. For example,

   - Time/Frequency Difference Of Arrival (TDOA/FDOA) data, using ARMA modeling.

   - Range only or Angle only data, using image reconstruction techniques or distributed versions of the backprojection technique described in Section 2.2.

   - Range and Angle data, using Nonlinear Estimation techniques.

   - Mixed, possibly inconsistent/incomplete data, using a hierarchical approach.

4. Advanced Concepts

   - The physical problem of locating and identifying sources has much mathematical structure; for example, it is heavily dependent upon the choice of coordinate systems. Non Euclidean Geometry and Non Classical Statistics (Non-Gaussian) will very probably be of great benefit.

   - Signal processing of one-dimensional signals is a very well developed field; spatial and other multi- dimensional problems, however, require more advanced mathematical tools.

   - Our preliminary investigations indicate that most candidate algoritms for distributed processing require high communication bandwidths. As an alternative approach, we are investigating Probabilistic Algorithms; these algoritms potentially require lower bandwidths, are naturally suited to parallel and distributed organizations, and they can be very robust.

   - From a systems-design perspective one should consider interactions between software and hardware architecture early in the development of algoritms. For this reason, we are considering the potential impact of VLSI/VHSI designs.

Using these approaches, examples of fully distributed processing and communication algoritms can be proposed. One such example is the combination of the TDOA approach [Schmidt], the distributed estimation algorithm in [ISL-DSN Report] and a distributed protocol a la [Merlin and Segall]. These algoritms have the desired robustness and low communication bandwidths that characterize desirable distributed algoritms.

References

Time of Arrival Estimation

Bangs, W.J. and P.M. Schultheiss, "Space-Time Processing for Optimal Parameter Estimation," Proc. of NATO Advanced Study Institute on Signal Processing, Univ. of Technology, Loughborough, U.K., August-Sept. 1972 also published in J.W.R. Griffiths, P.L. Stocklin, and C. van Schooneveld (Eds), NATO Advanced Study Institute on Signal Processing, Academic Press, 1973.

Carter, G.C. and C.H. Knapp, "Coherence and its Estimation via the Partitioned Modified Chirp-Z Transform", IEEE Trans. on Acoustic Speech and Signal Processing, ASSP-23, no. 3, pp 257-263, June 1975.

Carter, G.C., C.H. Knapp, and A.H. Nuttall, "Estimation of the Magnitude-Squared Coherence Function via Overlapped Fast Fourier Transform Processing", IEEE Trans. on Audio and Electroacoustics, Vol. AU-21, no. 4, pp 337-389, August 1973.

Carter, G.C., A.H. Nuttall, and P.G. Cable, "The Smoothed Coherence Transform", Proc. IEEE, pp 1497-1498, October 1973.

mann, W.R. and S.A. Tretter, "Optimum Processing for Delay-Vector Estimation in Passive Signal Arrays," IEEE Trans. on Info. Theory, Vol.IT-19, No.35, September 1973.

Knapp, C.H. and G.C. Carter, "The Generalized Correlation Method for Estimation of Time Delay", IEEE Trans. on Acoustic Speech and Signal Processing, ASSP-26, no. 6, pp 320-327, August 1976.

Poirot, J.L. and G.V. McWilliams, "Application of Linear Statistical Models to Radar Location Techniques", IEEE Trans. Aerospace and Electronic systems, AES-10, no. 6, pp 830-834, November 1974.

Schmidt, R.O., "A New Approach to Geometry of Range Difference Location", IEEE Trans. Aerospace and Electronic Systems, AES-8, no. 6, pp 821-835, November 1972.

Tracking Moving Targets

Lindgren, A.G. and K.F. Gong, "Properties of a Bearing-only Motion Analysis Estimation: an Interesting Case Study in System Observability", Proceedings of the Twelfth Annual Asilomar Conference on Circuits Systems and Computers, November 1978.

Lindgren, A.G. and K.F. Gong, "Position and Velocity Estimation via Bearing Observations", IEEE Trans. on Aerospace and Electronic Systems", AES-14, no. 4, pp. 564-577, July 1978.

Morgan, D.R., "A Target Trajectory Noise Model for Kalman Trackers", IEEE Trans. Aerospace and Electronic Systems, AES-12, pp 405-408, May 1976.

Poirot, J.L. and M.S. Smith, "Moving Emitter Classification", IEEE Trans. Aerospace and Electronic Systems, AES-12, no. 2, pp 255-269, March 1976.

Singer, R.A., "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets", IEEE Trans. Aerospace and Electronic Systems, AES-6, no. 4, pp 473-483, July 1970.

Singer, R.A. and K.W. Behnke, "Real-time Tracking Filter Evaluation and Selection for Tactical Applications", IEEE Trans. Aerospace and Electronic Systems, AES-7, no. 1, pp 100-110, January 1971.

Target Association

Alspach, D.L., "A Gaussian Sum Approach to the Multitarget Identification-Tracking Problem," Automatica, Vol.11, pp.285-296, May 1975, (earlier version in Proc. fourth symposium on Nonlinear Estimation, 1973).

Bar-Shalom, Y., "Extension of the Probabilistic Data Association Filter to Multitarget Environments," Proc. Fifth symposium on Nonlinear Estimation, U.C. San Diego, Sept. 1974.

Bar-Shalom, Y., "Tracking Methods in a Multitarget Environment", Survey Paper, IEEE Trans. on Automatic Control, vol.AC-23, pp.618-626, Aug. 1978.

Bar-Shalom,Y. and A. Jaffer, "Adaptive Nonlinear Filtering for Tracking with Measurements of Uncertain Origin," Proc. of the 1972 IEEE Conference on Decision and Control, pp.243-247, New Orleans, La., December 1972.

Bar-Shalom,Y. and E. Tse, "Tracking in a Cluttered Environment with Probabilistic Data Association," Proc. of the Fourth Symposium on Nonlinear Estimation, U.C. San Diego, Sept. 1973 and Automatica, vol.11, pp.451-460, Sept. 1975.

Jaffer,A.G. and Y. Bar-Shalom, "On Optimal Tracking in Multiple-Target Environments," Proceedings of the third Symposium on Nonlinear Estimation Theory and its Applications, pp.112-117, U.C. San Diego, Sept.1972.

Morefield, C.L., "Application of 0-Integer Programming to Multitarget Tracking Problems," Proc. IEEE Conf. on Decision and Control, Dec. 1975, and IEEE Trans. Auto. Control, vol.AC-22, pp.302-312, June 1977.

Reid, D.B. "A Multiple Hypothesis Filter for Tracking Multiple Targets in a Cluttered Environment," Lockheed Palo Alto Research Lab. Tech. Report, LMSC-D560254, Sept. 1977.

Sea, R.G., "An Efficient Suboptimal Decision Procedure for Associating Sensor Data with Stored Tracks in Real-Time Surveillance Systems," Proc. of the 1971 IEEE Conference on Decision and Control, pp.33-37, Miami Beach, Fla., Dec.1971.

Singer, R.A., and Sea, R., "New Results on Optimizing Surveillance System Tracking and Data Correlation Performance in Dense Multitarget Environments," IEEE Trans. Auto. Control, vol.AC-18, pp.571-581, December 1973.

Singer,R.A., and J.J. Stein, "An Optimal Tracking Filter for Processing Sensor Data with Stored Tracks in Real-Time Surveillance Systems," Proc. of the 1971 IEEE Conference on Decision and Control, pp.171- 175, Miami Beach, Fla., Dec.1971.

Singer,R.A., R. Sea and K. Housewright, "Derivation and Evaluation of Improved Tracking Filters for Use in Dense Multitarget Environments," IEEE Trans. Info. Theory, Vol.IT-20, pp.423-432, July 1974.

Sittler, R.W., "An Optimal Data Association Problem in Surveillance Theory," IEEE Trans. Mil. Electronics, vol.MIL-8, pp.125-139, April 1964.

Smith,P. and G. Buechler, "A Branching Algorithm for Discriminating and Tracking Multiple Objects," IEEE Trans. Auto. Control, vol.AC-20, pp.101-104, Feb. 1975.

Image Reconstruction

Brooks, R.A. and G. DiChiro, "Principles of Computer Assisted Tomography (CAT) in Radiographic and Radioisotopic Imaging", Phys. Med. Biol., vol. 21, no. 5, pp 689-732, September 1976.

Denton, R.V., A.J. Rockmore, and B. Friedlander, "An Image Reconstruction Approach to Target Association", Proceedings of the Twelfth Annual Asilomar Conference on Circuits, Systems, and Computers, Pacific Grove, California, November 1978.

Friedlander, B., R.V. Denton, and A.J. Rockmore, "The Inverse Problem in Radar and Optical Imaging", Proceedings of the 17th IEEE Conference on Decision and Control, January 1979.

Gordon, R. and G.T. Herman, "Three-Dimensional Reconstruction from Projections: A Review of Algorithms", Int. Rev. Cytol., vol. 38, pp 111-151, 1974.

Herman, G.T., A. Lent, and S.W. Rowland, "ART: Mathematics and applications", J. Theor. Biol., vol. 92, pp 1-32, 1973.

Horn, B.K.P., "Density Reconstruction Using Arbitrary Ray-Sampling Schemes", Proc. IEEE, vol. 66, no. 5, pp 521-562, May 1978.

Mersereau, R.M. and A.V. Oppenheim, "Digital Reconstruction of Multi-Dimensional Signals from their Projections", Proc IEEE, vol. 62, no. 10, pp 1319-1338, October 1976.

Scudder, H.J., "Introduction to Computer Aided Tomography", Proc. IEEE, vol. 66, no. 6, pp. 628-637, June 1978.

Special Issue on Physics and Computational Aspects of 3-dimensional Image Reconstruction, IEEE Trans. on Nucl. Sci., vol. NS-21, no. 3, June 1974.

Wood, S.L., "Stochastic Analysis of Iterative Image Reconstruction Algorithms," Proc. 11th Asilomar Conf. on Circuits, Systems and Computers, Pacific Grove, Calif., Nov. 7-9, 1977.

Wood, S.L., "A System Theoretic Approach to Image Reconstruction," Ph.D. Dissertation, Stanford University, May 1978.

Wood, S.L., M. Morf, A. Macovski "Stochastic Methods Applied to Medical Image Reconstruction," Proc. IEEE Conf. on Dec. and Control, New Orleans, Dec. 1977.

Spectral Estimation

spatial

Capon, J., "High Resolution Frequency-Wavenumber Spectrum Analysis," Proc. IEEE, vol.57, no.8, Aug.1969.

Capon, J., "Applications of Detection and Estimation Theory to Large Array Seismology," Proc. IEEE, vol.58, no.5, May 1970.

Capon, J., R.J. Greenfield, and R.J. Kolker, "Multidimensional Maximum-Likelihood Processing of a Large Aperture Seismic Array," Proc. IEEE, vol. 55, no.2, Feb. 1967.

Cron, B.F. and C.H. Sherman, "Spatial-Correlation Functions for Various Noise Models," JASA, Vol.34, no.11, November 1962.

Cron,B.F., B.C. Hassill and F.J. Keltonic, "Comparson of Theoretical and Experimental Values of Spatial-Correlation," JASA, Vol.37, No.3, March 1965.

Feintuch, P.L., C.L. Weber, "Specification and Performance of Passive Sonar Spectral Estimators," IEEE Trans. on Aerospace and Electronic Systems, vol.AES-9, no.6, pp.889-900, Nov. 1973.

Hodgkiss, W.S. and L.W. Nolte, "Covariance Between Factor Coefficients Representing the Time waveforms Observed from an Array of Sensors," JASA, Vol.59, no.3, March 1976.

Kung, W.A., "Maximum Entropy Spectral Analysis in the Spatial Domain," Report to the Rome Air Development Center, RADC-TR-78-160, July 1978.

Lacoss, R.T., "Data Adaptive Spectral Analysis Methods," Geophysics, vol.36, no.4, pp.661-675, Aug. 1971.

Nuttall, A.H., G.C. Carter, and E.M. Montaron, "Estimation of the Two-Dimensional Spectrum of the Space-time Noise Field for a Sparse Line Array", J. Acoust. Soc. Am., vol. 55, no. 5, pp 1034-1041, May 1974.

Temporal

Baggeroer, A.B., "Confidence Intervals for Regression (MEM) Spectral Analysis," IEEE Trans. on Inf. Theory, vol.IT-22, no.5, Sept. 1976.

Burg, J.P., "Maximum-Entropy Spectral Analysis," Soc. of Exploration Geophysicists, 37th meeting, Oklahoma City, Ok., Nov.1967.

Burg, J.P., "The Relationship between Maximum Entropy Spectra and Maximum Likelihood Spectra," Geophysics, vol.37, April 1972.

Burg, J.P., "Maximum Entropy Spectral Analysis," Ph.D. Dissertation, Stanford University, 1975.

Goodman, N.R., "The Distribution of the Determinant of a Complex Wishart Distributed Matrix," Annals of Math. Stat., vol.34, 1963.

Goodman, N.R., "Statistical Analysis Based on a Certain Multivariate Complex Gaussian Distribution (An Introduction)," Annals of Math. Stat., vol.34, 1963.

Jenkins, G.M. and D.G. Watts, Spectral Analysis and Its Applications, Holden-Day, 1968.

Khatri, C.G., "Classical Statistical Analysis Based on a Certain Multivariate Complex Gaussian Distribution," Annals of Math. Stat., vol.36, pp.98-119, 1965.

Lacoss, R.T., "Adaptive Combining of Sideband Array Data for Optimal Reception," IEEE Trans. on Geoscience Elec., Vol.GE-6, No.2, May 1968.

Marple, L., EE. Dissertation, Stanford University, 1977.

Beamforming

Anton, J.J. and A.J. Rockmore, "A Unified Approach to Array Factor Synthesis for Line Arrays with Nonuniformly Positioned Elements," IEEE Journal of

Oceanic Engineering vol.OE-1, No. 1, September 1976.

Chester, T.C. and J. Frank, "Array Antennas," Chapter 11 in M.I. Skolnik (ed.), Radar Handbook, McGraw-Hill, 1970.

Dolph, C.L., "A Current Distribution for Broadside Array Which Optimizes the Relationship Between Beam Width and Side-Lobe Level," Proc. IRE, vol.34, June 1946.

Elliot, R.S., "The Theory of Antenna Arrays", Chapter 1 in Microwave Scanning Antennas, vol. 2, R.C. Hansen (ed.), Academic, New York and London, 1966.

Jasik, H., Antenna Engineering Handbook, McGraw-Hill, New York, 1961.

Rodinelli, L.A. "Effects of Random Errors on the Performance of Antenna Array of Many Elements," IEEE Convention Record, vol.4, 1959.

Steinberg, B.D., Principles of Aperture and Array System Design, John Wiley & Sons, 1976.

Taylor, T.T., "Design of Line-Source Antennas for Narrow Beamwidth and Low Sidelobes," IRE Trans. on Antennas and Propagation, vol.AP-3, January 1955.

Williams, J.R., "Fast Beam Forming Algorithm," JASA, vol.44, no.5, pp. 1454-1455, 1968.

Array Processing

Bangs, W.J. "Array Processing with Generalized Beamformers," Ph.D. Thesis, Yale University, September 1971.

Bryn, F., "Optimum Signal Processing of Three-Dimensional Arrays Operating on Gaussian Signals and Noise," JASA, Vol.34, no.4, March 1962.

Cox, H., "Interrelated Problems in Estimation and Detection, Parts I & II," Proceedings of the NATO Advanced Study Institute on Signal Processing with Emphasis on Underwater Acoustics, Enscede, the Netherlands, August 1968.

Cox, H., "Optimum Arrays and the Schwartz Inequality," JASA, Vol.45, no.1, January 1969.

Cox, H., "Resolving Power and Sensitivity to Mismatch of Optimum Array Processors," JASA, Vol.54, no.3, September 1973.

Edelblute, D.J., Fisk, J.M. and G.L. Kinnison, "Criteria for Optimum Signal-Detection Theory for Arrays," JASA, vol.41, no.1, January 1967.

Feintuch, P.L. and C.L. Weber, "Separability of Space-Time Filtering for Optimal Passive Sonar Array Detection," unpuplished notes.

Griffiths, L.J.H., "Signal Extraction Using Real-Time Adaptation of Linear Multichannel Filter," Tech. Report 6788-1, System Theory Lab., Stanford University, February 1968. Hinich, M.J., "Detection of an Incoherent Finite Signal using a Number of Sensors," JASA, vol.42, no.5, pp.1089-1094, 1967.

Kobayashi, H., "Iterative Synthesis Methods for a Seismic Array Processor," IEEE Trans. on Geoscience Electronics, vol.GE-8, no.3, pp.164-178, July 1970.

Lewis, J.B. and P.M. Schultheiss, "Optimum and Conventional Detection Using a Linear Array," JASA, Vol.49, no.4 (Part 1), April 1971.

McDonald, V.H. and P.M. Schultheiss, "Optimum Passive Bearing Estimation in a Spatially Incoherent Noise Environment," JASA, Vol.46, July 1969.

Middleton, D. and H.L. Groginsky, "Detection of Random Acoustic Signals Receivers with Distributed Elements: Optimum Receiver Structures for Normal Signal and Noise Fields," JASA, Vol.36, no.5, November 1965.

Pasupathy, S., A.N. Venetsanopoulos, "Optimum Active Array Processing Structure and Space-Time Factorability," IEEE Trans. on Aerospace and Electronic Systems, vol. AES-10, no.6, pp.770-778, Nov. 1974.

Rockmore, A.J. and N.J. Bershad, "Nearfield Discrimination Capabilities of a Line Array," JASA, Vol. 56, no.3, September 1975.

Rockmore, A.J., Lecture notes, Stanford University, Summer 1977.

Scharf, L.L., "On Stochastic Approximation and the Hierarchy of Adaptive Array Algorithms," Proc. IEEE Conf. on Dec. & Control, Paper no.TA2-6, 1972. Schweppe, F.C., "Sensor-Array Data Processing for Multiple Signal Sources," IEEE Trans. on Inf. Theory, vol.IT-14, no.2, pp.294-305, March 1968.

Stocklin, P.L., "Space-Time Sampling and Likelihood Ratio Processing in Acoustic Pressure Fields," J.Brit. I.R.E., July 1963.

Van Trees, H.L., "A Unified Theory for Optimum Array Processing," A.D. Little, Inc., Report4160866, August 1966.

Vanderkulk, W., "Optimum Processing for Acoustic Arrays," J.Brit. I.R.E., October 1963.

Adaptive Arrays

Chang, J.H., F.B. Tuteur, "A New Class of Adaptive Array Processors," JASA, vol.49, pp.639-649, March 1971.

Frost, O.L., "An Algorithm For Linearly Constrained Adaptive Array Processing," Proc. IEEE, vol.60, no.8, pp.926-935, Aug. 1972.

Gabriel, W.F., "Adaptive Arrays - An Introduction," Proc. IEEE, vol.64, no.2, pp.239-272, Feb. 1976.

Liggett, W.S., "Passive Sonar Processing for Noise with Unknown Covariance Structure," JASA, Vol.51, no.1, January 1972.

Owsley, N.L., "Source Location with an Adaptive Antenna Array," Report of the Naval Underwater Systems Center, New London Lab., Jan. 1971.

Widrow, B., P.E.Mantey, L.J. Griffiths, and B.B. Goode, "Adaptive Antenna Systems," Proc. IEEE, Vol.55, no.12, Dec. 1967.

Widrow, B., et al, "Adaptive Noise Cancelling Principles and Applications," Proc. IEEE, vol.63, pp.1692-1716, Dec. 1975.

MISCELLANEOUS

Anderson, V.C., "Digital Array Phasing," JASA, Vol.32, no.7, July 1960.

Arase, T. and E.M. Arase, "Deep-Sea Ambient-Noise Statistics," Journal of the Acoustical Society of America (JASA), vol.44, no.6, December 1968.

Brown, J.L., "Annarmonic Approximation and Bandlimited Signals," Info. and Control, Vol.10, 1967.

Carutners, J.W., "Lectures on Marine Acoustics - Volume I: Fundamentals of Marine Acoustics," Texas A&M Report No. TAMU-SG-73-402, June 1973.

Corson, D.R. and P.Lorrain, Introduction to Electromagnetic Fields and Waves, W.H. Freeman & Co., 1962.

Deutsch, R., Estimation Theory, Prentice Hall, 1965.

Faran, J.J. and R. Hills, "The Application of Correlation Techniques to Acoustic Receiving Systems," Harvard University Acoustic Research Laboratory Technical Memo 28, 1 November 1952, reprinted in JASA, Vol.57, no.6, June 1975.

Feintuch, P.L., "Passive Sonar Spectral Estimation," Ph.D. Thesis, University of Southern California, Los Angeles, California, June 1972; also published as Technical Paper No.TP-11-14, Hughes Aircraft Company, Fullerton, California, June 1972.

Flinn, E.A., "A Theoretical Discussion of Optimum Multi-channel Filter Design," Seismic Data Lab. report 227, Teledyne Industries, Inc., Alexandria Va., December 1966.

Frame, J.J., "Matrix Functions and Applications," IEEE Spectrum, March, April, May, June, and July, 1964.

Graupe, D., Identification of Systems, Van Nostrand Reinnold, 1972.

Griffith, L.J.R., "A Simple Adaptive Algorithm for Real-Time Processing in Antanna Arrays," IEEE Proc., vol.57, no.10, pp.1696-1697, Oct. 1969.

Hann, W.R., "Optimum Signal Processing for Passive Sonar Range and Bearing Estimation," JASA, Vol.58, no.1, July 1975.

Haubrich, R.A., "Earth Noise, 5 to 500 Millicycles per Second," Journal of Geophysical Research, vol.70, no.6, March 1965.

Helstrom, C.W., Statistical Theory of Signal Detection, Pergamon Press, 1960.

Hirsch, P., "Spatial Processors for Passive Surveillance (U)," Proc. of the Optimal/Adaptive Space Processing Seminar, White Oak, M.D., August

Jacobson, M.J., "Space-Time Correlation in Spherical and Circular Noise Fields," JASA, vol.34, no.7, July 1962.

Kanefsky, M., "Detection of Weak Signals with Polarity Coincidence Array," IEEE Trans. Info. Theory, Vol.IT-12, No.2, April 1966.

Levinson, N., "An Heuristic exposition of Wiener's Mathematical Theory of Prediction and Filtering," J.Math. Phys., Vol.26, 1947.

Marsh, H.W., "Correlation in Wave Fields," U.S. Navy Underwater Sound Laboratory Quarterly Report (Confidential - declassified), 31 March 1950.

Merlin, P.M., A. Segall, "A Failsafe Distributed Routing Protocol," submitted for publication.

Mermoz, H., "Filtrage Adaptive et Utilization Optimale d'une Antenne," Proc. NATO Advanced Study Institute on Signal Processing with Emphasis on Underwater Acoustics, Grenoble, France, 1964.

Morf, M., B. Friedlander, J. Newkirk, G. Verghese, "Investigation of New Fast Algorithms For Locating and Identifying Spatially Distributed Sources and Receivers," Annual Report for DARPA Contract MDA 903-76-C-0179, September 1976.

Murphy, D.A., A.J. Hockmore, and N.J. Bershad, "Optimal Estimation of Source Location and Spectrum in the Near Field," Hughes Aircraft Report FR74-11-755, AD A005521, November 1974.

Nahi, N.E., Estimation Theory and Applications, Wiley, 1969.

Nuttall, A.H. and D.W. Hyde, "A Unified Approach to Optimum and Suboptimum Processing for Array," U.S. Underwater Sound Laboratory, USL Report 992, April 1969.

Pryor, C.N., "A Simplified Automatic Tracking Technique for Signal Correlation Systems," Report NOLTR67-152A, U.S. Naval Ordnance Lab, White Oak, MD., September 1967.

Ramo,S., J.R. Whinnery and T. VanDuzer, Fields and Waves in Communication Electronics, John Wiley & Sons, 1965.

Roberts, C.A. and A.G. Favret, "Application of Monopulse Tracking Techniques to Passive Linear Arrays," JASA, vol.51, no.1, January 1972.

Rudnick, P., "Small Signal Detection in the Presence of Interference," JASA, Vol.32, no.7, July 1960.

Scharf, L.L., P.H. Moose, "Information Measures and Performance Bounds for Array Processors," IEEE Trans. on Inf. Theory, vol.IT-22, no.1, pp.11-21, Jan. 1976.

Senor, S.W., "Adaptive Techniques to Discriminate Against Coherent Noise in a Narrow-Band System," JASA, Vol.39, no.1, January 1966.

Schultheiss, P.M., and F.B. Tuteur, "Optimum and Suboptimum Detection of Directional Gaussian Signals in an Isotropic Gaussian Noise Field," IEEE Trans, Mil. Elec., July-October 1965.

Schultheiss, P.M., "Passive Sonar Detection in the DIMUS Array," JASA, Vol.43, No.3, March 1968.

Seidman, L.P., "Bearing Estimation Error with a Linear Array," IEEE Trans. on Audio and Electroacoustics, Vol. AU-19, No.2, June 1971.

Skilling, H.H., Fundamentals of Electric Waves, Second Edition, John Wiley & Sons 1948.

Urick, R.J., Principles of Underwater Sound for Engineers, McGraw-Hill, 1967.

Van den Nos, A., "Alternative Interpretation of Maximum-Entropy Spectral Analysis," IEEE Trans, on Info. Theory, Vol.IT-17, No.4, July 1971.

Van Trees, H.L., Detection, Estimation and Modulation Theory, Part 1, John Wiley & Sons, 1968.

Wenz, G., "Acoustic Ambient Noise in the Ocean: Spectra and Sources," JASA, Vol.34, no.12, December 1962.

Woodward, P.M., Probability and Information Theory with Applications to Radar, Pergamon Press, 1953.

SINGLE-SITE DETECTION AND TARGET PARAMETER ESTIMATION

by Paul Demko, Jr.
Lincoln Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02142

I. Detection and Estimation Problems

Detection theory problems are characterized by the fact that one must decide which of several alternatives are true. For example, if there are two alternatives, if a target is there or not, the problem is a binary detection problem. In estimation theory problems, which very closely parallel those of detection theory, one must estimate parameters of a signal or situation. For example, one type of signal estimator or receiver estimates values of a continuous parameter. It will rarely compute the exact signal, but attempts to be close most of the time.

The problems in which detection and estimation theory are utilized can be categorized into three levels or a hierarchy as shown in Figure 1. Level 1 problems include detecting and estimating known signals in noise; level 2, signals with unknown parameters in noise; and level 3, random signals in noise. The particular problems that will be discussed in more detail in this note are marked with an asterisk.

The general detection/estimation problem is more clearly explained with the aid of Figure 2. The real situation of interest includes the presence or absence of a target; its characteristics, e.g., its reflectivity or noise emissions; and its parameters, e.g., range, azimuth, and velocity. Noise such as ground clutter, weather formations, or background acoustic noise corrupts the target information as it propagates from the area of interest to the sensor. The sensor transforms the information presented to it into measurements or points in observation space, for example, power levels in azimuth or range bins. When the observation space has a finite number of dimensions, the problem is a classical detection theory problem. The detection statistic calculator processes the measured data to produce appropriate statistics so that target detection and parameter estimation can best be carried out. The results are the decision, and if positive, estimates of target metrics.

For illustrative purposes, consider the simple, idealized example in which the presence of a target is indicated by a d.c. level of 'd' volts; the larger the target, the greater the voltage level. Absence of the target is indicated by 0 volts. Zero-mean Gaussian noise is added to the voltage level and the sensor takes samples of the sum of the signal level and the noise. The observation space consists of the values of the sampled voltage. A possible detection rule is to compare values of the sampled voltage to a threshold, η. If the sample value is greater than η, a target is declared to be present; otherwise the area of interest is declared to be empty. The two fundamental errors that result in this example and in detection theory in general are false alarms, declaring that a target is present when it is not, and false dismissals, declaring no target when one is present. One usually wants to maximize the probability of detection, P_D, without significantly increasing the probability of a false alarm, P_F. For this simple example, P_D is plotted against P_F with d as a parameter in Figure 3. In the figure, the actual threshold used in the decision rule is proportional to η. The detection probability is plotted against the voltage level, d, with P_F as a parameter in Figure 4. Both figures illustrate that there is a trade-off between the two major errors. If one lowers the threshold to detect smaller targets, one must contend with a higher false alarm rate. Alternately, if one increases the threshold to keep the false alarm rate low, one must accept a lower probability of detection.

II. DSN Strawman Small Acoustic Arrays

1. Introduction.

The DSN strawman acoustic sensor, a small acoustic array, has been introduced in an earlier paper, Lacoss, "Tutorial on Sensor Technology," (these proceedings). The signal processing that must be performed to calculate appropriate statistics to detect and estimate target parameters--high resolution frequency-wavenumber analysis--is briefly described here. In the case of a small acoustic array, the target metric that is estimated is the acoustic azimuth. The acoustic azimuth is the true azimuth of the target at the time the acoustic signals being processed were emitted from the target. Because the propagation velocity of sound is of the same order of magnitude as the velocity of the target, the target has usually moved a considerable distance by the time the acoustic signals propagate to the sensor. The statistic that is calculated from the array data is power as a function of frequency and wavenumber. An explanation of how this result can be interpreted for target detection and azimuth estimation is first presented. Next the high resolution algorithm is described and its signal processing steps are enumerated. Finally some detection and estimation problems are discussed.

2. Frequency-Wavenumber Analysis.

High resolution frequency wavenumber analysis performs frequency-domain beamforming on the array data with the additional feature that output noise is minimized. Conventional beamforming and the resulting target detection and estimation of acoustic azimuth is illustrated by considering the acoustic sensor, a planar array of microphones. A plane acoustic wave impinges upon the array along an elevation angle ϕ from the vertical, and from the horizontal azimuth θ. Although the wave propagates at the velocity of the medium, v, it traverses the array at the apparent velocity, $C = v/\sin\phi$. The apparent velocity ranges from v for a wave propagating across the array parallel to the ground to infinity for a wave propagating onto the array from directly overhead.

The analysis, in effect, hypothesizes that a wave at a particular frequency, f, is crossing the array from a horizontal angle θ at an apparent velocity, C. The position of each of the array microphones is used to calculate the time delay or phase that the hypothetical wave would have at each microphone. The data measured at each microphone are appropriately delayed, summed with all the other such data and integrated. Coherent energy with the hypothesized azimuth and phase velocity adds constructively, whereas energy from other directions and at other phase velocities do not.

The resulting (θ,C) space is parameterized by θ and f/C, where $K = f/C$ is called the wavenumber of the wave. Another way of interpreting K is $2\pi/\lambda_x$, where λ_x is the apparent wavelength of the wave. The possible range of K and θ is the area inside a circle with radius f/C as shown in Figure 5. In the analysis, this area is discretized and the beam power at each point is calculated. If a wave indeed is passing through the array with apparent velocity C' from direction θ', the power will peak at $(\theta',f/C')$ or (θ',K'). The equal-power contours of a peak that indicates a wave propagating parallel to the plane of the array are shown in the figure.

3. The High Resolution Algorithm.

There are two basic steps in the high resolution frequency-wavenumber analysis. First, the acoustic array data are processed to compute estimates of the power spectral density covariance matrices for all the frequencies of interest. (In the DSN strawman, approximately 50 frequencies between 5 and 250 Hz will be considered.) Second, at each frequency, a spatial wavenumber analysis is performed. Wavenumber power estimation corresponds to a discrete two dimensional spatial Fourier transform with spatial sampling points determined by microphone positions. At each frequency, the power at approximately 800 to 1000 wavenumber points is calculated. Both azimuth and phase velocity resolution depend on frequency, and the number of wavenumber points that are calculated vary with the selected frequency.

The input data are M = 10 sampled waveforms collected from the M microphones in the sensor. During each two-second analysis interval, data is recorded continuously for 1.024 seconds at a 2 kHz per channel rate. The 2048 samples from each channel are broken up into K = 5 overlapping blocks, each of N = 512 points. Each block is multiplied by an appropriate window and a Fourier transform is performed on each block. Thus, M*K N-point Fourier transforms are performed, resulting in K estimates of the matrix periodograms, $X_i(f_k)$. The computations described from this point are all performed for each frequency. An estimate of the power spectral density covariance matrix for frequency, f_k, is calculated:

$$C_{ij} = \sum_{l=1}^{K} X_i^l(f_k) X_j^{l*}(f_k),$$

where * denotes complex conjugate. The resulting matrix is then normalized to equalize power in each channel:

$$R_{ij}(f_k) = \frac{C_{ij}(f_k)}{(C_{ii}(f_k)C_{ii}^*(f_k)C_{jj}(f_k)C_{jj}^*(f_k))^{1/2}}.$$

All the elements along the diagonal of $R(f_k)$ are one.

Several matrix modifications are done before the actual estimates of power. First a small diagonal matrix is added to the $R(f_k)$ to force the matrices to be non-singular:

$$R(f_k) = R(f_k) + \epsilon I.$$

Then, the complex $R(f_k)$ are replaced by their inverses.

The final step is to estimate power for each wavenumber of interest. The estimation of power, given frequency and wavenumber, reduces to calculating a Hermitian quadratic form using $R^{-1}(f_k)$ as the matrix and a complex steering vector, E_{ik}. The elements of the steering vector are complex representations of the phase delay for each of the microphone locations for a given wavenumber. Specifically, the frequency-wavenumber power estimate is

$$P_{ik} = \frac{1}{E_{ik}^H R^{-1}(f_k) E_{ik}},$$

where i is the wavenumber index and k is the frequency index. The superscript H implies con-

jugate transpose.

Power levels are thus computed at all appropriate wavenumbers for each frequency of interest, a total of approximately 42,000 frequency-wavenumber bins. A detailed computational sizing of real-time high resolution analysis indicates that approximately 11 million real arithmetic operations (adds or multiplies) per second are required.

4. Detection and Estimation.

Detection and estimation consist of examining the power vs. frequency and wavenumber results. A peak in power that rises several dB above background noise levels for a particular wavenumber at several frequencies most likely indicates a target. A time average of the results is kept such that the current power-frequency-wavenumber results may be compared to those of the recent past. If a peak occurs in several sequential results and appears to be moving at a reasonable speed, its detection is further confirmed and one can initiate an azimuth track. However, if the target does not appear to be moving, it is either traveling on a radial path towards the sensor or it is a stationary noise source. Data from other sensors might clarify such a situation. Effective, automatic target detection and azimuth estimation is a difficult problem and is currently being addressed.

III. Coherent Pulsed Doppler Radar

1. Introduction.

The DSN strawman radar has been described in a previous paper, Lacoss, "Tutorial on Sensor Technology", (these proceedings). The radar sensor is considerably more sophisticated than a small acoustic array and more target metrics can be estimated from the return signals in the observation space: target range, azimuth, and velocity. The signal processing required to calculate appropriate statistics is substantially different from that used with the acoustic array; however, the results and detection/estimation rules used to detect and determine target metrics are very similar. Since the current study has not investigated the radar problem in depth, less detail will be presented than was for the acoustic sensor.

Pulse doppler radar observation space is divided into range-azimuth-velocity (doppler) bins. The return times of the reflected pulses and location in the scan of the antenna determine into which range and azimuth bins the return data are placed. The pulses returning during a coherent processing interval or that time during which the antenna "dwells" on a target are processed and power levels for the different doppler shift frequency or velocity bins result. Similar to that for the acoustic sensor, detection and estimation for radar consists of determining the existence and location (the proper bins) of peaks in power, which are presented as a function of range, azimuth, and velocity.

2. Radar Processing.

Briefly, the radar signal processing consists of the following:

-data checking and calibration corrections
-high-pass filter (3-pulse canceler)
-windowing
-fast Fourier transform and magnitude computation
-threshold computation, target detection.

3. Detection/Estimation.

The major outputs resulting from the signal processing are (1) map data and (2) target reports. The map data represents a picture of zero-velocity ground clutter. It is extracted from those range-azimuth bins with zero-doppler shift. Target reports are declared when the power in non-zero velocity bins exceed various thresholds.

Several interesting detection problems occur. When a target has zero radial velocity with reference to the radar, its power returns appear in the zero doppler bins. However, the target's power will move with reference to the relatively stable ground clutter. Such targets can be detected in the clutter by noting any rapid time variations in the clutter map.

Often, clutter such as clouds and other weather formations can have non-zero velocities and mask targets. However, the clouds usually cover a much larger area (many more range-azimuth bins) than an aircraft. An appropriate time-varying threshold can be calculated by averaging over an area consisting of several range-azimuth bins (at constant doppler shift). Thus the larger area with relatively high power is not declared a target, but the smaller and sharper peak is detected.

The azimuth resolution of the strawman radar is approximately 10 degrees--36 azimuths per 4 second scan or 9 coherent processing intervals per second. There are 100 range bins covering 0.125 to 12.5 kilometers (range resolution of approximately 120 meters). The coherent processing interval contains returns from 512 pulses. Thus, information from 900 range-azimuth bins is processed every second: 900 512-point complex FFTs are computed per second. The net computational requirements for FFT processing is 12.5 million real arithmetic operations (adds or multiplies) per second. It is estimated that the other computations listed above require an equivalent amount of computation. Thus, the total computational requirements for the DSN strawman coherent pulsed doppler radar are approximately 25 million real arithmetic operations per second.

Level	Detection Theory	Estimation Theory
1. Known signals in noise.	1. Synchronous digital communication. 2. Pattern recognition problems.	1. PAM,PFM communications systems with phase synchronization. 2. Inaccuracies in inertial systems.
2. Signals with unknown parameters in noise.	*1. Conventional pulsed radar or sonar, target detection. 2. Target classification. 3. Digital communication systems without phase reference. 4. Digital communication over slowly fading channels.	*1. Range, velocity, or angle measurement in radar/sonar. 2. Discrete time, continuous amplitude communication systems.
3. Random signals in noise.	1. Digital communication over scatter link, orbiting dipole channel, or chaff link. *2. Passive acoustics/sonar. 3. Seismic detection system. 4. Radio astronomy (detection of noise sources).	*1. Power spectrum parameter estimation. *2. Range or doppler spread target parameters in radar/sonar problem. 3. Velocity measurements in radio astronomy. *4. Target parameter estimation: passive acoustics and sonar. *5. Ground mapping radars.

Fig. 1. Detection/estimation theory hierarchy (after Van Trees, 1968).

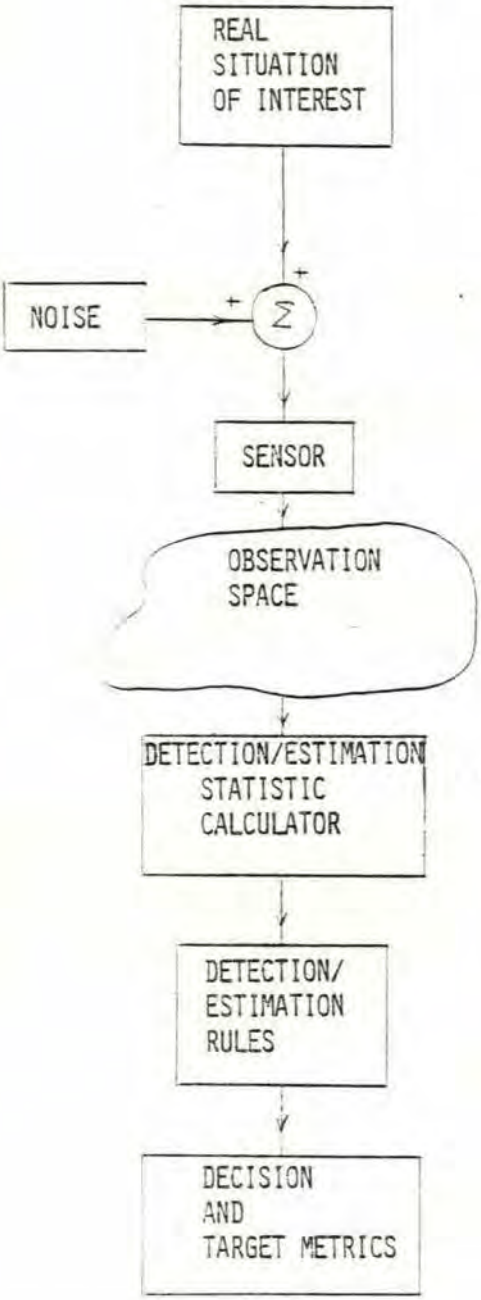


Fig. 2. Detection/estimation block diagram.

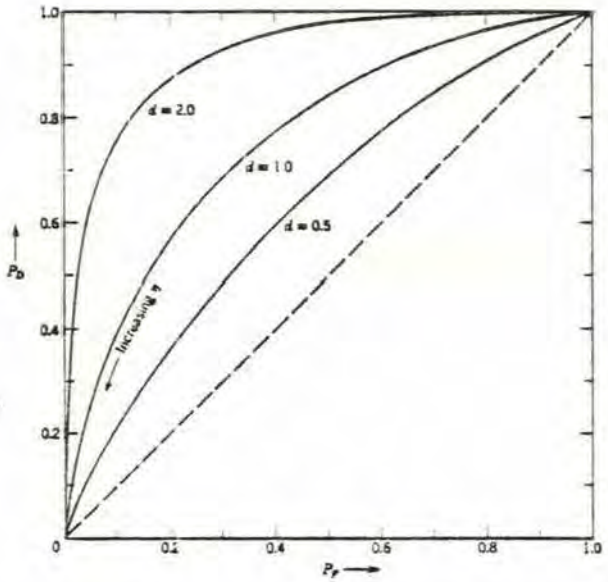


Fig. 3. Probability of detection vs. probability of false alarm with threshold and presence level as parameters.

after Van Trees (1968)

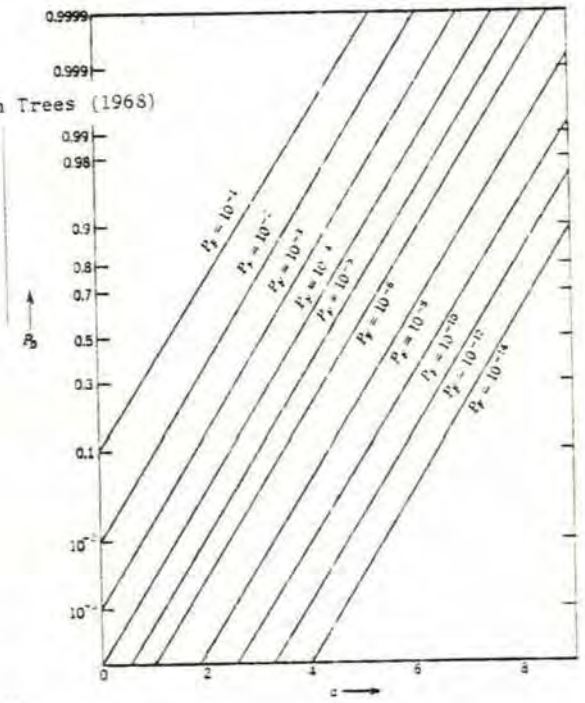
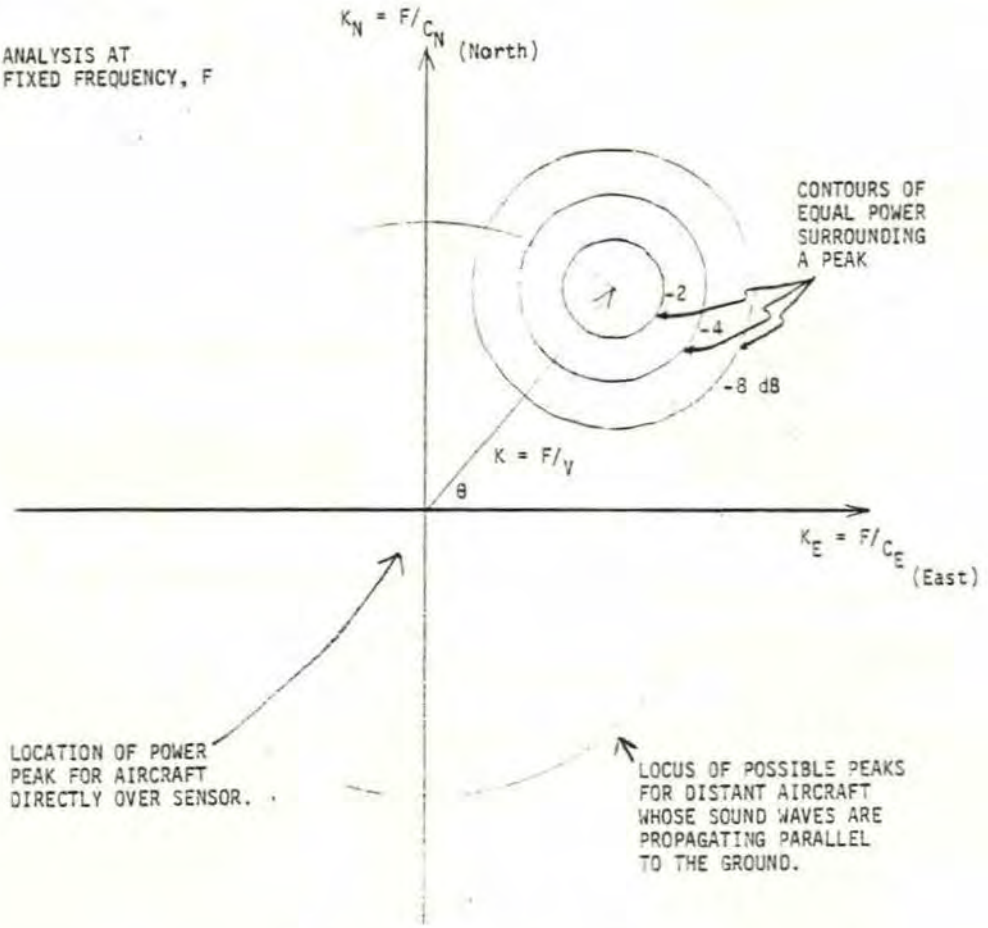


Fig. 4. Probability of detection vs. presence level with probability of false alarm as a parameter.



θ: AZIMUTH OF SOUND SOURCE
V: VELOCITY OF SOUND IN AIR
K = F/C

Figure 5. Wavenumber power plot interpretation.

DYNAMICALLY MODIFIABLE DISTRIBUTED SYSTEMS

A. N. Habermann
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

ABSTRACT

A distributed system can be constructed so that modifications can be made while it is running. Dynamic modifications include simple changes in the code, modifications in the representation of data objects and restructuring of individual data objects or entire modules. Applied to a heterogeneous computer network, coexistence of local implementations allows optimal performance in each node. Special care must be taken that modification of one part of the running system does not cause an avalanche of changes in programs that use it.

1. INTRODUCTION

It is generally recognized that the cost of software maintenance exceeds by far the cost of initial system design. Software maintenance is no longer considered to be a temporary debugging phase, but a permanent effort to improve system performance, to adapt a system to new and improved hardware and to provide better user facilities. Therefore, much is gained by facilitating system maintainability.

Maintenance of a distributed system that runs on a computer network causes some specific problems. It may be impossible or very undesirable to bring the running system to a halt and shut it down for some time in order to replace some of its parts. Typical examples of such systems are airline ticket reservation systems, banking systems, etc. This raises the question of constructing a distributed system which is modifiable while it is running. This is the topic of this working paper.

The question is answered by exploring in particular

a. system facilities needed for runtime modifiability

b. which kind of modifications can be put through while the system is running (changes in programs, changes in data representation)

c. the impact of modifications on the current state of the running system

d. the integration of modified parts with the running system.

2. ADDRESS SPACES

An important development in programming is the shift in emphasis from control flow to data abstraction. It used to be customary to partition a task along the time dimension of its subsequent activities. Research in programming methodology [1], software engineering [2] and programming language design [3] demonstrated the usefulness of organizing programs and software around data object definitions and their manipulations.

The basic concept for designing a dynamically modifiable system is that of "Address Space". An address space is a design module which provides certain facilities that can be used in other address spaces [4]. Typical facilities provided by an address space are:

a. creation of data objects which are monitored by that address space

b. operations on data objects monitored by that address space.

An address space is determined by three components, its specification, its own data and its code. The specification part is discussed further on in this paper. The own data part consists of the data structures which are needed for monitoring the objects created on behalf of other address spaces. Typical examples are a ready list used as own data in a process scheduling address space and a Symbol Table used by a lexical scanner (implemented as address space). The code part of an address space consists of subroutine and function programs.

The parts of an address space are placed in virtual memory segments. A segment is accessible through a segment descriptor which describes location, size and current state of a segment. In addition, each segment descriptor has a link field which may point to another segment descriptor and a reference count which reflects the number of descriptors pointing to it. The descriptors of an address space are linked in the order code segment, own segment, spec segment. A schematic representation of an address space is given in Figure 1.

Data objects are also placed in virtual memory segments. A data segment is attached to the address space that monitors the objects in that segment by linking its descriptor to the code segment descriptor of that address space. The reference count of a code segment is equal to the number of data segments that are attached to an address space.

3. DYNAMIC MODIFICATIONS

Modifications of an address space affect the various types of segments in various ways. We classify modifications into four categories [3]:

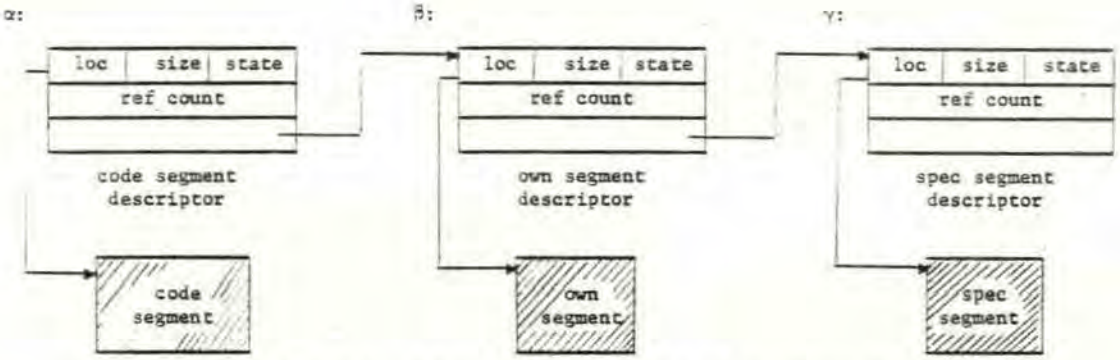


Figure 1. Schematic Representation of an Address Space

1. modifications which change the code, but leave all data representations as they are (comod)
2. modifications which change both code and data representation (repmod)
3. remodeling of existing data objects into a new representation (datamod)
4. modifications which change the representation of own data (owmmod).

3.1 Code Modifications

Let address space AS be determined by a code -, an own - and a spec segment whose respective descriptors are c, o and s. Let c' be the descriptor pointing to a new version of the code for AS. We now wish to replace old version c by new version c' (assuming that the representation of data remains unaltered). This is accomplished in a very simple way. All we need to do is to exchange the three fields (loc, size, state) of the descriptors c and c'. The effect of this exchange operation is that the old descriptor c now points to the new version of the code and the new descriptor c' to the old code version. The advantage of changing to the new code version in this way is that the reference count is preserved and that the linking of data segments, code segment, own segment and spec segment is not affected at all!

3.2 Data Representation Modifications

We introduce the notion of the "current version" of an address space. The current version is the most recently installed (code, own) segment pair. We use the link field in the spec segment descriptor to point to the code segment of the current version (see Figure 2).

In case of "repmod", a new version is introduced without deleting older versions. The new code segment descriptor c' is linked to the existing own segment descriptor o, o's ref count is incremented by one and the version pointer in spec segment descriptor s is set to point to c' (see Figure 2).

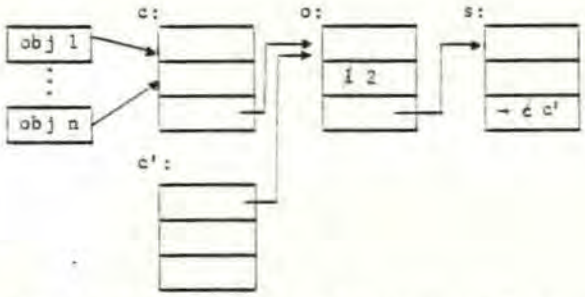


Figure 2. Data Representation Modification "repmod"

The reason for keeping the old version around is that "old" data objects (objects which were created before the representation was modified) must be handled by the old code version, because they still have the old representation. New data objects which are created after the representation modification are automatically shaped after the latest fashion and attached to the current version c' which is found through the version pointer in s.

If the representation of an address space is modified several times, there may be a number of different versions (their number is counted in ref count (o)), each having a number of data objects attached. All these objects are still usable and correctly operated upon through the pointers of data segment descriptors to code versions.

3.3 Remodeling of Data Objects

Data representation modification as discussed in the preceding section does not automatically include reshaping of old data objects. We saw that this is not necessary, because old objects can still be operated upon by the old version of the code. It is also difficult to reshape all data objects automatically, because there is no facility provided to find all objects attached to an address space. Their number is known, but no backwards referencing is provided.

However, when an old data object is accessed, its user should have the option of reshaping it according to the most recent data representation modification. This can be done if it is known how the current state of an old object is mapped into an equivalent state of a new object. This mapping must be provided by the designer of a new data representation. The mapping can be defined as a pair of programs "decode","encode". (This solution differs significantly from the one presented in [6].) Assuming that a simple canonical representation of data objects is given, procedure "decode" maps the current state of an old object into the canonical representation. Procedure "encode" maps this canonical version into an initialized object in the current representation so that the initialized new object is equivalent to the state of the old object.

When the old object is deleted, the reference count of the code descriptor it points to is decremented. If this reference count goes to zero, no object is attached to this version any longer, so it can be deleted (while the reference count in the own descriptor is decremented). The new object created by encode is attached to the current code segment descriptor (whose reference count is incremented by one).

After several representation modifications there may be several versions in use which each need their specific decoding procedure. Thus, a natural place to put a decoding procedure is in the code segment of its version. There are two reasons which suggest that the natural place for the encoding procedure is in a unique spec segment pointed to by the spec segment descriptor. First, there is only one current version, so a single encoding procedure suffices. Second, in the same way that a decoding procedure defines a path from an (old) implementation to the specs, the encoding procedure defines a path from the specs to the current implementation.

On the other hand, each time the data representation is modified, the designer of the modification must provide a decode procedure (in order to enable remodeling in the future) and an encode procedure for reshaping existing data objects. Although these procedures are not used at the same time, the fact that they are designed and written together with the new code version is an argument in favor of including both procedures in the new code segment. This has the additional advantage that there is no need for "repmod" to set the spec segment descriptor to the new encoding procedure. But it has the disadvantage that old encode procedures which will never be used again remain in existence and occupy memory space.

3.4 Reshaping Own Data

The most radical change of an address space is caused by modifying the representation of own data. This affects code and own segment and possibly the representation of data objects as well. Moreover, since own data is unique, the old own segment must be mapped into the new own segment immediately when the new representation is introduced and the old code and own segments must be invalidated. This implies that all old data segments are also useless, because they cannot use the old address space version.

The designer of a new own data representation must provide a procedure for mapping the current own segment with all its information into an equivalent new own segment. This procedure is executed exactly once when "ownmod" is performed. In addition, he must provide a new code version, including encode and decode procedures. The effect of "ownmod" on the linking information is that a new cycle ($c' \rightarrow o' \rightarrow s \rightarrow c'$) is formed where c' and o' are the new code and own segment descriptors. The old links are not deleted except for the current version pointer. Although all old data objects are useless, "ownmod" is not able to find these objects and remodel them. This does not matter, because we can wait for each data object until it is accessed. By that time it is found that the code segment descriptor it is attached to has been invalidated. The data object is then reshaped by applying the decode procedure of the old version and the current encode procedure. The newly created object is attached to the current code descriptor and the old object is deleted.

It turns out that reshaping an old object is accomplished by "datamod" in all cases. However, after a "repmod" applying "datamod" to an old data object is optional, whereas after "ownmod" application of "datamod" is mandatory.

4. REPRESENTATIONS IN COMPUTER NETWORKS

In case the nodes of a computer network are not identical, a variation on the ideas described in preceding sections is useful for "node specific" implementations of address spaces. Let us assume that most operations on data objects are performed locally in the network node in which the object resides. If the nodes are different machines, it is likely that one can find different optimal implementations for each node. The following organization makes it possible to use such optimal implementations.

An address space has a unique own segment which resides in some node. Operations on own data take place across the network (and should therefore be kept to a minimum). We abandon the idea of a unique current version of the code. Instead, each node of the network will have its own version (possibly more than one) and its own version of representing data objects. All code versions point to the unique own segment descriptor and all objects in a node are attached to its local code version. Efficiency is achieved if most operations on data objects are performed locally.

In order to allow migration of data objects from one node to another, each version of a code segment should contain a decode and encode procedure. When a data object is transferred from node A to node B, we apply the decode procedure of

A to it and the encode procedure of B. This guarantees full flexibility among all nodes that host an implementation of a particular address space.

It is obvious that this scheme pays off only if the frequency of migrating data objects is much smaller than that of local operations. The beauty of this organization is that the local implementations are entirely independent of one another. Implementation details of one node are entirely hidden from other nodes. Communication takes place through a simple general specification.

5. INTEGRATION OF NEW VERSIONS INTO THE RUNNING SYSTEM

If a code segment is replaced by a new version, we must assume that the starting addresses of corresponding subroutines and functions in the two versions are not the same. The question is how programs of other address spaces refer to these procedures.

The thing we wish to avoid is that existing code must be changed because code it has been linked to has been replaced. This problem is solved by using "entry vectors" in code segments. An entry vector is an array of procedure starting addresses. Each procedure is associated with a particular element of the entry vector. Programs in other code segments call a procedure by its entry vector element which is the same for different versions of the code.

The cost of this solution is an extra level of indirection in calling a procedure. Indirection is inexpensive on most machines and negligible compared to the normal overhead of a procedure call. The great advantage is in the fact that modifications affect the new version only, while all existing programs using it do not need to be changed.

It is not necessary for a user to include entry vectors in his program, or for a compiler to generate them. It is a simple matter for a link editor to derive a new entry vector from the new code version and the old entry vector. If it is allowed that procedures are given in a different lexical order in a new version, it is necessary to remember the mapping of procedure names onto entry vector elements. This information is useful anyway, because it facilitates resolving external procedure references in new programs.

CONCLUSION

A system can be constructed so that it is modifiable while it is running. The kind of modifications which can be made in a running system vary from simple changes in programs to modifications of entire address spaces. It is possible to work with data objects of different representations in a single address space. Old representations lose their validity only if an entire address space is modified.

The use of different data representations and code implementations may be very useful for a heterogeneous computer network. Local operations can be very efficient and migration of objects is made possible by local decode and encode procedures.

Modifications in one address space should not necessitate changes in other address spaces which use its facilities. The use of entry vectors assures that references to external procedures remain correct when the defining code segment is modified.

REFERENCES

[1] Dahl, O. J., E. W. Dijkstra, C. A. R. Hoare, Structured Programming, Academic Press, London and New York, 1972.

[2] Parnas, D. L., "On the Criteria to be Used in Decomposing Systems into Modules," Comm. ACM 15, 12, December 1972.

[3] Wulf, W. A., R. L. London, M. Shaw, "An Introduction to the Construction and Verification of Alphard Programs," IEEE Trans. on Software Engineering, SE-2, 4, December 1976.

[4] Habermann, A. N., L. Flon, L. Cooprider, "Modularization and Hierarchy in a Family of Operating Systems," Comm. ACM 19, 5, May 1976.

[5] Goullon, H., R. Isle, K. P. Loehr, "Dynamic Restructuring in an Experimental Operating System," Proc. of the Third Conference on Software Engineering, Atlanta, Georgia, May 1978.

[6] Fabry, R., "How to Design a System in which Modules can be Changed on the Fly," Proc. of the Second Conference on Software Engineering, San Francisco, California, October 1976.

APPLICATION OF KNOWLEDGE BASED PROGRAMMING TO SIGNAL UNDERSTANDING SYSTEMS

Cordell Green and Brian P. McCune
Systems Control, Inc.
Palo Alto, California

1. Introduction

The PSI knowledge based programming system is a computer program that acquires high level descriptions of programs and produces efficient implementations of these programs [Green-76]. Simple symbolic computation programs are specified through dialogues that include natural language, input-output pairs, and partial traces. The programs produced are in LISP, but experiments have shown that the system can be extended to produce code in a block structured language such as PASCAL.

This paper provides a brief description of the PSI system and discusses its applicability to the task of synthesizing signal understanding systems in time-critical environments.

2. The PSI Program Synthesis System

The PSI system works as follows. The user wants a performance program of some type, for example, a news story indexing program. The user must be versed in the application area, though need not be an expert at programming or understand how the PSI system writes programs. The user conducts a dialogue with PSI, using natural language as well as traces, examples, and very high level languages to describe the desired program. PSI synthesizes the program. Since program specification and use is an evolutionary process, each successive version of the candidate target program is coded or interpreted by PSI and tested by the user. Modifications in specification result in successive versions of the target program as new requirements develop or previous requirements are clarified.

PSI is organized as a collection of interacting modules or programmed experts as displayed in Figure 1. The major data paths and modules of the PSI system are shown in Figure 2. There is one data path for each specification method. Currently these are English, input-output examples, and partial traces. A more conventional method, that of a very high level language, is a planned addition to PSI as shown in Figure 2. These specifications are integrated in the program net and model.

PSI's operation may be conveniently factored into two parts (see Figure 1): the acquisition phase (those modules shown left of the program model), which acquires the model, and the synthesis phase, which produces a program from the model.

In the acquisition phase, sentences are first parsed, then interpreted and stored in the program net (also referred to as the "program specification" in [Ginsparg-78]). The parser is a general parser which limits search by incorporating considerable knowledge of English usage. The interpreter is more specific to program synthesis, using program description knowledge as well as knowledge about the question asked and the current topic to facilitate interpretation into the program net.

The dialogue moderator guides the dialogue by selecting or suppressing questions for the user. It attempts to keep PSI and the user in agreement on the current topic, provides a review and preview of topics when the topic changes, helps the user who gets lost, and allows initiative to shift between PSI and the user.

A new module is the explainer, which generates in English reasonably clear questions about and descriptions of program models as they are acquired, in order to help verify that the inferred program description is the one desired. It will also be able to explain the how and why of the acquisition and synthesis process to the interested user.

Another input specification method is a partial trace [Phillips-77]. A trace includes as a special case an example input-output pair. Examples are useful for inferring data structures and simple spatial transformations. Partial traces of states of internal and I/O variables allow the inductive inference of control structures. The trace and example inference expert infers a loose description of a program in the form of a program net, rather than a program model or other true algorithm. This technique allows domain support to disambiguate possible inferences and also separates the issue of efficient implementation from the inferenece of the user's intention.

Various types of programming knowledge are distributed throughout the modules of the acquisition phase. In contrast, knowledge specific to one particular application domain (e.g., knowledge about learning programs) is concentrated in the domain expert, which supplies domain support by communicating with other acquisition modules through the program net.

The program net and the program model (see Figure 2) are two of the major interfaces within PSI. Both are high level program and data structure description languages. The program model includes complete, consistent, and interpretable very high level algorithm and information structures. The program net, on the other hand, forms a looser program description. Fragments of the program net can be visited in the order of occurrence in the dialogue, rather than in execution order, and allow less detailed, local, and only partial specification of the program. Since these fragments correspond rather closely to what the user says, they ease the burden of the parser/interpreter as well as the trace and example inference module.

The program model builder [McCune-77] applies knowledge of correct program models to convert the fragments into a model. The model builder processes fragments, checking for completeness

and correctness, fills in detail, corrects minor inconsistencies, and adds cross-references. It also generalizes the program description, converting it into a form that allows the coder to look for good implementations. The completed program model may be interpreted by the model interpreter to check that it performs as desired by the user and also to gather information needed by the efficiency expert, such as statistics on set sizes and probabilities of the outcome of tests.

After the acquisition phase is complete, the synthesis phase begins. This phase may be viewed as a series of refinements of the program model into an efficient program, or as a heuristic search in a refinement tree for an efficient program that satisfies the program model. The coder [Barstow-77] has a body of program synthesis rules [Green & Barstow-75, Green & Barstow-77] which are applied to gradually transform the program model from abstract into more detailed constructs until it is in the target language. The algorithm and data structures are refined interdependently. The coder deals primarily with the notions of set and correspondence operations and can synthesize programs involving sequences, loops, simple input and output, linked lists, arrays, and hash tables.

The refinement tree effectively forms a planning space that proposes only legal, but possibly inefficient, programs. This tree structure is shared by the coder and the efficiency expert [Kant-77]. When the coder proposes more than one refinement or implementation, the efficiency expert reduces the search by estimating the time-space cost product of each proposed refinement. The better path is followed, and there is no backup unless the estimate later proves to be very bad. An additional method to reduce the size of the search space is the factorization of the program into relatively independent parts so that all combinations of implmentations are not considered. An analysis for bottlenecks allows the synthesis effort to concentrate on the more critical parts of the program.

The entire PSI system can now be used by a knowledgeable user. Programs have been generated from English dialogues for a variety of domains. Among these are:

- CLASS: A simple pattern classification program which requires much of the programming knowledge necessary for more complex programs.
- IF: Theory Formation, whose goal is to "learn" (form) an internal model of a concept by repeatedly examining examples of the concept.
- NEWS: An information retrieval program.
- Sorting algorithms: Efficient sorting algorithms for specific sorting requirements.

3. Application to Signal Understanding Systems

One new application of knowledge based programming is in the area of processing acoustic data from a distributed sensor network. We are investigating the automatic generation of the harmonic-set formation portion of the SIAP system [Drazovich & Brooks-78].

In addition, we are working on basic issues so that similarly developed systems can be applied in other important military mission areas such as electronic warfare surveillance. In the undersea surveillance application area the signal understanding program is designed to produce a description of the ocean scenario, changing with time, that indicates the platforms in the ocean that are generating the signals being perceived by the sensors of the undersea surveillance system. The ships and submarines being detected and tracked are in a noisy ocean environment that may also contain other ships of no interest to the surveillance system.

Suppose an analyst is attempting to analyze signals being received by hydrophones directed toward a group of submarines and other platforms in a noisy environment. The analyst must find the location and type of each ship and associate each frequency found with a likely source. This task is known to be quite difficult. It far exceeds the capabilities of any straightforward parametric classification or pattern recognition system. It is at the limit of what is achievable by knowledge based signal understanding systems consisting of large rule bases and programs that model the sources (blades, shafts, pumps, etc.), harmonic and ratio relations of sources, types of sources on platforms, operational patterns, the ocean environment, the noise sources, maximum speed of the platforms, whether the locations are shipping lanes, and so forth. If the platforms decide to change their sound, they could disguise themselves effectively by changing their source characteristics (e.g., by using tone altering synthesizers, running close together, using alternate pumps and acoustic masking devices, running near sound-reflecting structures, and altering their operating patterns). The situation can be further complicated by the introduction of new types of microphones or signal processing systems. With these kinds of changes happening the problem is challenging indeed. The problem is that the signal understanding program would have to be preprogrammed to anticipate each possible change in data rates, harmonic structures, amplitude and frequency modulation, etc. Similarly, any approach using learning would also have to anticipate all of the types of changes that could be expected and to be able to search the very large space of possible changes to find new patterns. It is quite unlikely that it will be possible to anticipate all such changes.

A more reasonable approach might be to allow the signal understanding system to be reprogrammed to respond to new patterns. The difficulty now is the time to reprogram and debug a complex system in the short time allowed in a tactical situation. The necessary reprogramming and debugging is of course a slow process. A solution to the reprogramming problem is to use an automatic program synthesis system to reprogram or modify existing

programs and data structures to meet the new requirements.

A scenario for the response to new signal characteristics might begin with the signal understanding system failing to respond, providing information inconsistent with other observations, or reducing certainty factors for identified sources. We assume that the appropriate portion of the existing signal understanding system was itself automatically synthesized, and that the associated explainer module could describe in English the performance of the system and help pinpoint the type of signal changes that are causing the problem. The user requests, in English, probes into the data to look for any patterns that characterize sources that seem to cause trouble. A user may know from another observation the identity and location of a particular source. If so, the user could request a learning program to find patterns, expressed as rules, that characterize that source. The signal understanding system would then be automatically reprogrammed to use the new rules and reanalyze the signals. This interactive process is repeated until a satisfactory understanding of the signals is achieved.

A more difficult situation occurs when the truth of a situation cannot be established by means of any controlled experiments, which is frequently the case. For example, in a noisy ocean environment one can never positively identify all the platforms to determine what is really producing the signals. Since one cannot ascertain truth, one can only judge that a given signal analysis is satisfactory according to some set of criteria. Then the program must still find new rules that produce some satisfactory model of the situation, according to criteria that an adequate model of the situation would satisfy.

The difficulty is compounded, in that it will not in general be possible to anticipate what criteria or meta-rules a satisfactory model must satisfy. For example, the analyst might suddenly notice that the number of submarines has drastically increased. One might add a constraint not anticipated by the system designer that it isn't possible for submarines to replicate themselves. The cause for the increase in sound sources might be that some sound generator was altered and the harmonics produced were taken as separate sources. It would then be appropriate to relax the constraints on grouping of harmonics so that previously disallowed harmonic structures would be acceptable if they arise from the same location. Another situation might be that the sounds weren't recognized because the submarines began moving at speeds that were not anticipated. One might have the system generate its best hypothesis that assumes that anything moving very quickly or erratically is not really a submarine but instead a decoy. One could also add constraints for a hypothesis that best explains all possible sound sources or is least likely to miss especially interesting ones.

The first major task for our new system (called CHI) in undersea surveillance is reprogramming the signal classification module illustrated in Figure 3. The system we develop will input the old signal classification module, plus new signal classification rules in a language natural for expressing them. CHI will produce as output a modification of the original signal classification program which appropriately makes use of these new rules. The classification program is, in this case, primarily a harmonic-set formation program that partitions the set of frequency signals into a harmonically related group produced by one source of one platform. The classification program will use as primitive operations (1) existing primitives of the target programming language, (2) signal retrieval commands to a data management system, and (3) subroutines in a simple statistics library.

The second major task being considered for CHI is to write a module that learns or hypothesizes new pattern classification rules on its own. The input for this task is a list of constraints that all rules must satisfy. The output is a program that modifies old rule sets based upon new signal information about known situations.

The feasibility of many other applications may not be far off. The promise lies in our approach; namely, that of building a large knowledge based system that emphasizes the codification of underlying programming principles combined with application specific expertise. Some generality has already been demonstrated by extending PSI to deal with ostensibly different kinds of programs, using essentially the same knowledge base.

References

1. [Barstow-77] Barstow, David R., Automatic Construction of Algorithms and Data Structures Using a Knowledge Base of Programming Rules, Ph.D. thesis, Memo AIM-308, Report STAN-CS-77-641, Artificial Intelligence Laboratory, Computer Science Department, Stanford University, Stanford, California, November 1977.

2. [Drazovich & Brooks-78] Drazovich, Robert J., and S. Brooks, "Surveillance Integration Automation Project (SIAP)", these proceedings.

3. [Ginsparg-78] Ginsparg, Jerrold M., Natural Language Processing in an Automatic Programming Domain, Ph. D. thesis, Memo AIM-316, Report STAN-CS-78-671, Artificial Intelligence Laboratory, Computer Science Department, Stanford University, Stanford, California, June 1978.

4. [Green-76] Green, Cordell, "The Design of the PSI Program Synthesis System", Proceedings Second International Conference on Software Engineering, Computer Society, Institute of Electrical and Electronics Engineers, Inc., Long Beach, California, October 1976, pages 4-18.

5. [Green & Barstow-75] Green, Cordell, and David Barstow, "Some Rules for the Automatic Synthesis of Programs", Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Volume 1, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 1975, pages 232-239.

6. [Green & Barstow-77] Green, C.C, and D.R. Barstow, "A Hypothetical Dialogue Exhibiting a Knowledge Base for a Program Understanding System", in Elcock, E.W., and D. Michie, editors, Machine Intelligence 8: Machine Representations of Knowledge, Ellis Horwood, Ltd., and John Wiley and Sons, Inc., New York, New York, 1977, pages 335-359.

7. [Kant-77] Kant, Elaine, "The Selection of Efficient Implementations for a High Level Language", PROCEEDINGS OF THE SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND PROGRAMMING LANGUAGES, SIGPLAN Notices, Volume 12, Number 8, SIGART Newsletter, Number 64, August 1977, pages 140-146.

8. [McCune-77] McCune, Brian P., "The PSI Program Model Builder: Synthesis of Very High Level Programs", PROCEEDINGS OF THE SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND PROGRAMMING LANGUAGES, SIGPLAN Notices, Volume 12, Number 8, SIGART Newsletter, Number 64, August 1977, pages 130-139.

9. [Phillips-77] Phillips, Jorge V., "Program Inference from Traces Using Multiple Knowledge Sources", Proceedings of the Fifth Joint Conference on Artificial Intelligence-1977, Volume 2, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, August 1977, page 812.

Figure 2: Major Paths of Information Flow in PSI

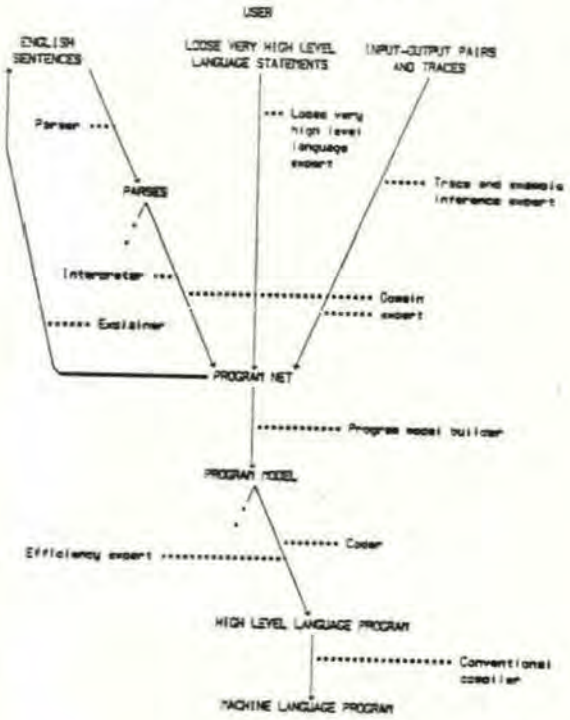


Figure 1: Block Diagram of the PSI Program Synthesis System

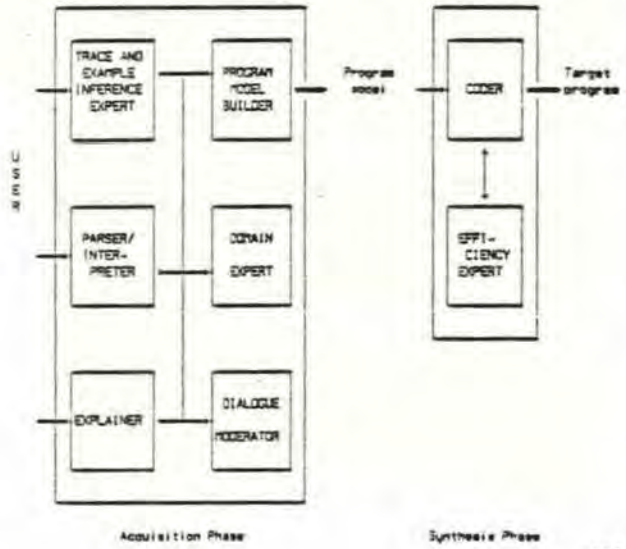
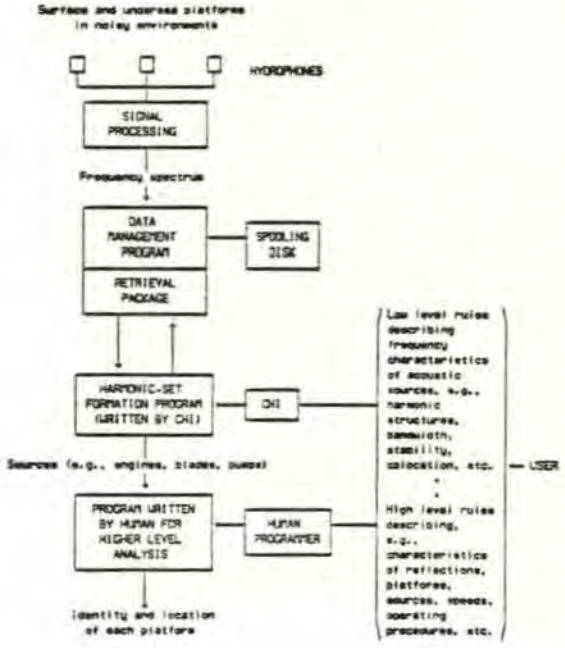


Figure 3: Signal Understanding Application



SURVEILLANCE INTEGRATION AUTOMATION PROJECT (SIAP)

Robert J. Drazovich and Scottie Brooks
Systems Control, Inc.
Palo Alto, California

Abstract

The SIAP project is an ARPA-funded effort which is involved in research that explores the feasibility of applying artificial intelligence techniques to the integration of surveillance information for detecting, classifying, and tracking platforms in the ocean. A knowledge-based system has been developed which processes multiple types of input data using several different types of knowledge and analysis tools. The system attempts to build and update over time a model of the ocean scene which it is observing. This paper provides a short description of the SIAP system, a brief history of the SIAP project and its evolution of the current capabilities of the system, an overview of the program's structure, and an indication of how it operates.

1. SIAP Goal and General Description

The Surveillance Integration Automation Project (SIAP) is developing a knowledge-based system for automating tasks involved in undersea and ocean surveillance. The program develops and maintains a model of activities in a specified area of the ocean. That is, SIAP is being designed to detect, identify, classify, and track all platforms in its area of interest. The program goal is to provide improved performance over the Navy's currently implemented surveillance system. SIAP is expected to provide an intelligent assistant for a surveillance system operator so that he can more effectively analyze the large and diverse amounts of available data. In addition, SIAP will allow the Navy to reduce its manpower requirements for operation of an effective surveillance system both by using fewer people and by requiring less extensive training of many of the remaining personnel. Figure 1 indicates the environment in which SIAP is operating.

The approach used by SIAP is to integrate knowledge and information from a variety of sources to produce the best possible response to a given situation. SIAP makes use of information obtained from signal processing algorithms, statistical analysis, factual knowledge, informal knowledge (heuristics in the form of production rules) and knowledge inferred from SIAP's previous conclusions.

SIAP processes data from a variety of distributed sensor systems. It receives data from a variety of sources, in varying formats, and at varying times. Data representing raw acoustic signals is presented to SIAP. Normally, data arrives from multiple arrays and provides SIAP with information about the same scene as detected from different locations. In addition SIAP could receive preprocessed data from other sensor systems (such as HFDF and OTH radar). The form in which SIAP receives that information is as a report of a contact and any details (such as classification, location, course, etc.) about the contact that the sensor system can provide. Similarly, SIAP can process manually generated reports about a contact such as movement reports, visual sitings, and intelligence reports.

The SIAP program undertakes several specific tasks in completing its overall analysis. These include detection and verification of algorithmically detected signals; harmonic set formation; signature formation; source and platform identification; location, course, and speed estimation; multi-array and multi-beam processing; and maintenance of a scenario history including explanations of why decisions were made. The SIAP program is organized so that each activity is processed by a separate program module known as a knowledge source. Each knowledge source performs an individual task which contributes to the overall analysis by making inferences about relationships between model components. Processing flow between the knowledge sources is controlled by another knowledge source, which contains rules indicating the proper order in which the analysis tasks are to be completed.

2. SIAP Program Structure

The SIAP program is organized into program modules. Each module is responsible for one task in the overall analysis process. The process involves building a multi-level model of the scenario being considered. For example, acoustic input comes to SIAP in the form of "lines." These lines are formed in line sets (for example, harmonic sets), which are in turn combined across multiple arrays into sources. A source is the system's representation of an actual noise-making device on a platform (an engine, pump, propulsion shaft, etc.). The sources are then combined into platforms which provide a representation of the entire scene. Figure 2 indicates the tree-like structure formed by combining these various model entities, or hypothesis elements.

The description above is simplistic in many respects. Most importantly, it does not consider the non-acoustic sensor and manually generated input data processed by SIAP. These systems typically provide information about a platform (rather than a line). This allows SIAP to hypothesize a platform node and then generate "expectations" about the acoustic manifestation of the platform (e.g., which lines would appear in the acoustic data). Thus, SIAP coordinates and combines the raw acoustic data signals with higher level information. The multiple types of inputs confirm each other and much of the processing flow and many of the knowledge sources are concerned with this task. This is a powerful mechanism because the different types of input data can support each other by filling in information gaps during periods when an input type is either absent or inaccurate.

Program flow in the SIAP system occurs by the passage of events through the various knowledge sources. There are three types of events in SIAP: (1) knowlege-based events, (2) time-based events, and (3) problem-list events. Knowledge-based events are the most numerous and reflect either the arrival of new input data into the system or occurrence of a significant decision or inference by a knowledge source. Each knowledge source communicates with other sources of knowledge by reporting additions or modifications which it has made to the scenario model through knowledge-based event declarations. As knowledge-based events are generated they are queued and are sequentially processed by another knowledge source known as CONTROLRULES. CONTROLRULES examines the event and calls other knowledge sources that would be interested in the actions and inferences reported by that event. For example, if the event reflects a location change in a platform, the program modules which calculate speed and course information would be notified.

Time-based events and problem list events are more specialized methods of processing flow control. Each are used by knowledge sources to either regain processing control or to force a specific activity. Time-based events allow a knowledge source to be activated at a specified time. This provides a mechanism for SIAP to delay a decision until more complete information becomes available, or until a definite pattern of behavior emerges over time. Problem list events are similar to time-based events except they are triggered by a specific event (for example, the appearance of a specific line, an anticipated change in course, etc). Knowledge sources use problem list events to generate expectations as to possible future activities, and then to take action if their expectations are realized.

Figure 3 summarizes the processing flow control described above. Note that the system actually maintains two major data bases: the model of the undersea scenario it is developing and a history file which describes the analysis and decision processes completed by SIAP. This history allows system users to understand why SIAP did what it did. Associated with virtually every one of SIAP's decisions is a confidence factor which indicates how sure the program was of its actions. These confidence values are used to help control the model development, but also provide insight into the analysis process. A more complete discussion of the current SIAP program structure can be found in Reference 6.

3. SIAP Development History

The initial phase of the SIAP effort (known as HASP-the Heuristic Adaptive Surveillance Project) was concerned with demonstrating the basic feasibility of developing a knowledge-based system within the context of the undersea surveillance problem. This effort involved the construction of a preliminary prototype program which was the basis for SIAP's current design. The prototype program was designed to function on synthetically created data and within limited scenarios of definite duration. In September 1975, HASP demonstrated the feasibility of the knowledge-based system approach by successfully analyzing several synthetic scenarios [1].

The SIAP project itself was started in June 1976 with a goal of demonstrating that the technology developed by HASP could be applied to real world situations and further to develop a prototype operational system. In the past two years, SIAP has moved toward these goals by completing three major milestones. The first two of these demonstrated SIAP's ability to operate on more complex scenarios (eventually multiple arrays of acoustic data and non-acoustic data for several hours duration) using "manually transcribed" real world data sets. The transcription process involved SIAP receiving data which had been transcribed from an analog representation (a lofargram) of the acoustic signal. SIAP was able to detect and classify the major contacts in data sets on which it was tested. SIAP's performance was compared to that of expert human acoustic analysts and was judged good to excellent in most cases [2,3].

Since the summer of 1977, a major research emphasis of SIAP has been to interface with an automatic signal processing front end. This step would completely automate the surveillance process involving SIAP because it would replace the process of manually transcribing the logargram data into digital form. The automated signal detection process with which SIAP has interfaced is the Signal Imagery and Measurement System (SIMS) developed by Bolt, Beranek, and Newman, Inc. [4,5]. A major exercise held by the SIAP project in March 1978 was aimed at demonstrating the initial interface between the SIAP and SIMS processes. SIAP was able to detect the major contacts in the data, but because of the changes in the input data (e.g., more noise, less accurate signal parameter estimation, less detail because of time averaging) SIAP did not perform as well as it had on manually transcribed data.

4. Current and Future Research Issues

The SIAP project is an exciting research effort in many respects. Among these issues are:

a. SIAP operates in an uncertain environment. Most of SIAP's input sources are subject to inaccuracies and ambiguities. Thus, SIAP must develop hypotheses about the true scenario and rate the certainty of the various alternatives. Resolution of conflicts between different types of inputs is also an important issue.

b. SIAP is a time oriented system. It is attempting to receive data and update a scenario in near real time. This presents another dimension for a knowledge-based production rule system in terms of controlling processing flow and comparing system entities. Because SIAP is relying on multiple types of inputs, a problem also

arises in timing the arrival of data. A hypothesis may be formed before a late arriving piece of information contradicts the analysis and forces "re-thinking" of the problem. In a real time system with new data continuously arriving, a "backtracking" scheme is difficult, so development of a dynamically adapting scenario which considers most major alternatives presents an interesting challenge.

c. SIAP integrates information from multiple data collection sources and relies on multiple domain experts. Unlike many artificial intelligence systems which attempt to emulate a task currently accomplished by a single individual, SIAP considers a task which is currently not done by one human. Knowledge from domain expert in each of the data collection systems is being combined to provide an overall system. The tasks of developing complicates the effort.

5. Current and Future Research Efforts

The SIAP project is currently involved in the following areas of development.

a. Work has continued toward improving SIAP's performance on SIMS derived data. This activity has involved the continued refinement of SIAP's knowledge sources. In addition, efforts have been undertaken to capture a portion of the SIMS process, the array line formation module, and include it as part of the SIAP system. It is felt that both the signal detection and line formation activities of the SIMS process and the higher level grouping activities of SIAP would benefit from the information supplied by the other.

b. Efforts are being undertaken to allow SIAP to receive inputs from a wider variety of non-acoustic data sources in addition to more detailed acoustic information gather systems. Issues such as the timing problems discussed above are being considered. SIAP's role as a possible allocator of data collection resources is also being undertaken. SIAP is being trained to detect when additional information is required in order to make a decision and to request the necessary data from the appropriate data collection system.

c. An expanded interactive SIAP/user system is being developed to allow the human user of SIAP to view the details of the various data bases if desired. The actual decision and analysis process is also being made visible to help users understand SIAP's actions.

References

1. Feigenbaum, E.A., H.P. Nii, J.L. Miller, S. Brooks, M. Model, N.J. Miller, HASP Final Report, Systems Control, Inc., sponsored by DARPA, October 1975 (secret).

2. Williams, Arnold, SIAP Experiment Report, MITRE Technical Report MTR-7470, February 1977 (secret).

3. Williams, Arnold, SIAP Experiment Report, MITRE Technical Report MTR-7720, February 1978 (secret).

4. Estrada, Dick, "Array Line Formation via Two-Dimensional Processing," memorandum to SONAR Group, Bolt, Beranek, and Newman, Inc., August 1975 (confidential).

5. Estrada, Dick, "SIMS1 User's Manual," <SIAM> SIMS-MANUAL. DOC; 1, September 1977.

6. Drazovich, R., R. Payne, S. Brooks, SIAP Final Report, Systems Control, Inc., November 1977 (secret).

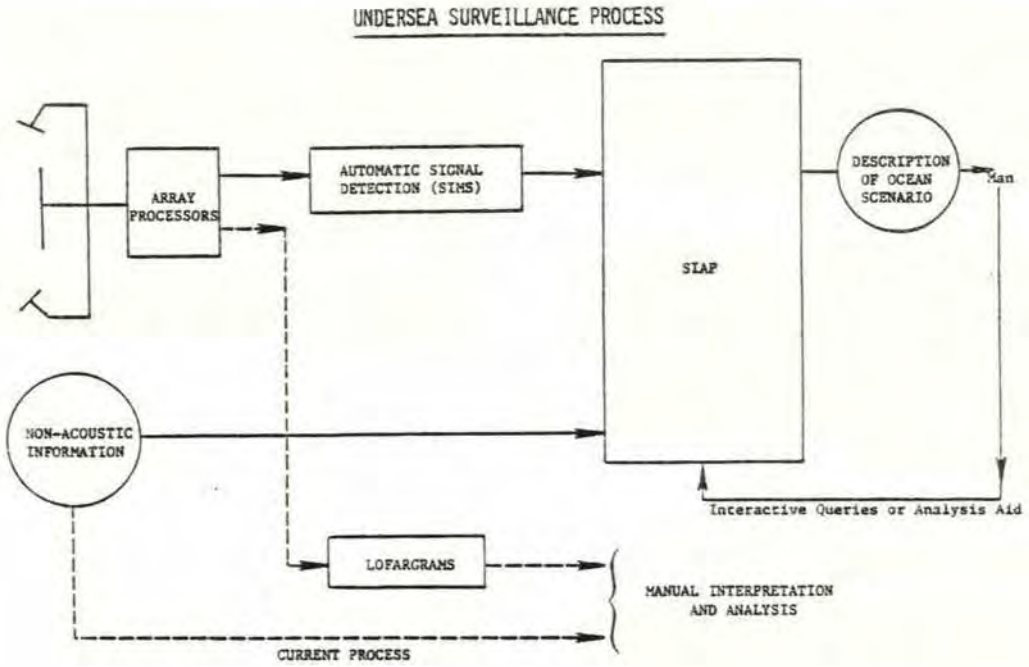


FIGURE 1

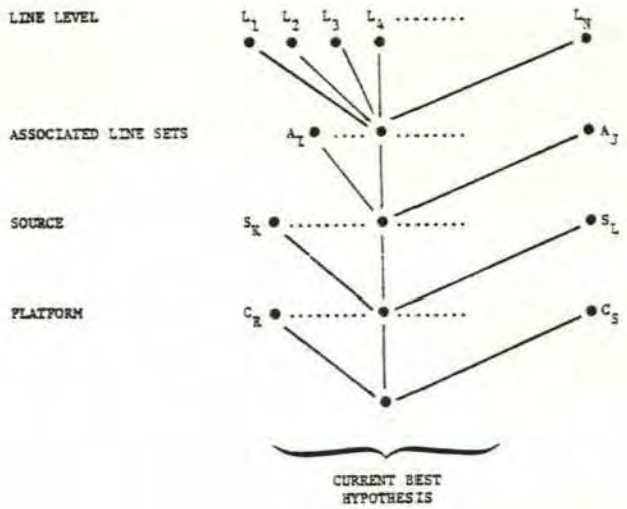


FIGURE 2

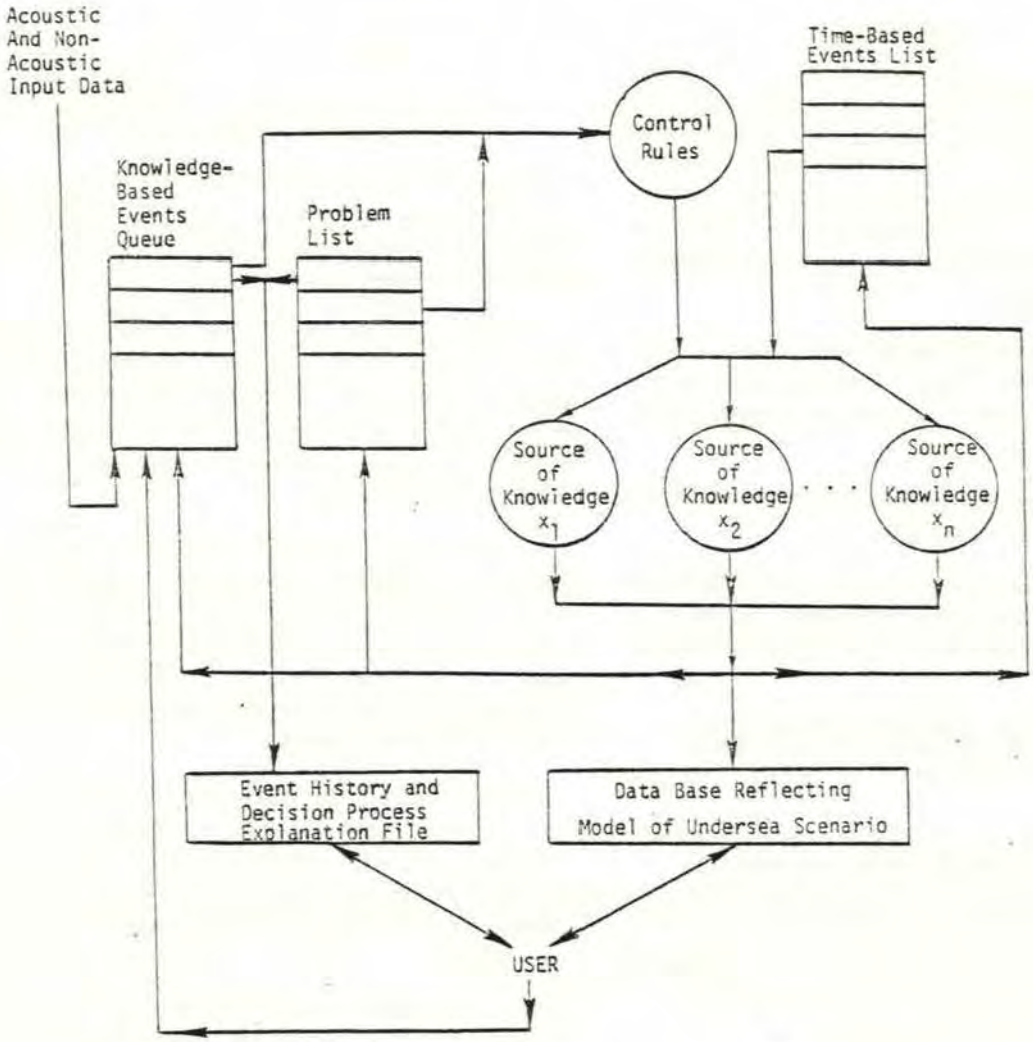


FIGURE 3

Protocols for DSN

(Abstract)

Danny Cohen
USC/Information Sciences Institute

Communication is the backbone of DSN. Without it - no cooperation between its components can take place. One of the most important lessons which we, as a community, have learned recently is that communication is not only the problem of transferring **data**, but mainly the the **problem** of transferring **information**.

Low level protocols are the ones whose focus of attention is on the bit traffic issue, whereas the higher level of protocols are concerned with the transport of higher level of information **abstraction**.

In a paper in these proceedings [1] the issues of high level protocols (HLL) are discussed.

Application of the philosophy of that paper to DSN yields that the basic problems is **WHAT** should the distributed components say to each other, rather than **HOW** do they say it. In particular:

- The major issue in designing a protocol for DSN systems is not the mechanics of communication, but the parallel processing organization of the system.
- Even though the HLL protocol hides all the communication details and its idiosyncrasies, the performance issues may still penetrate through it, all the way back into the DSN system.

By analyzing the DSN scenario several classes of information are found in need of being communicated.

1. Information about the DSN system,
2. Information about the **STATE** of the world,
3. Hypotheses, conjectures, and the like, and
4. Special requests for specific actions.

The first class, captures most of the **STATIC** (and slowly changing) information like the momentarily configuration of the system, its components, their capabilities and limitations, their position and the like.

The second class, captures the **DYNAMICS** of the situation which the DSN system is supposed to detect. This includes mainly detected targets and associated attributes.

In situations where some/all of the sensors are mounted on mobile platforms (for examples on cruise missiles), the **INTERNAL** information class, (1), may be as dynamic as the **EXTERNAL** information class, (2).

One may guess that the first class information generally has to be communicated in relatively high reliable fashion, and is not very urgent. On the other hand, the information of the second class is much more timely, as it may be very urgent, but probably is redundant enough such that less reliability may be required.

The third class of information, conjecture and hypotheses, relates to the intelligence and smartness of the system, and allows components to receive advance warnings about expected upcoming events (such as the appearance of a target in a particular area), which could be utilized for focusing attention and for gathering supporting evidence for some hard to detect situations.

The fourth class, the special requests, are for interactions such as flow control to protect critical resources, like overloaded processors and communication media. It also includes requests for specific information, about default values used throughout the systems, modes of operations, and the like.

We strongly believe that it is too early now to focus the attention on the mechanics of the communication, and on the interaction of the DSN protocols with the supporting lower level protocols.

REFERENCE

[1] R. F. Sproull and D. Cohen, "High Level Protocols", Proceedings of the IEEE, November 19768, Special Issue on Packet Communication Networks, pp. 1371-1386.

MACHINE RECOGNITION AND UNDERSTANDING OF MANUAL MORSE

Albert Vezza, P. David Lebling,
Edward H. Black*, Timothy A. Anderson*, John F. Haverty*,
David Sherry*, and Gail E. Kaiser

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

A. INTRODUCTION

"Morse code (as sent by hand) is one of the simplest of aural languages, yet it has features common to all spoken languages." That was written nearly 20 years ago by M. Freimer et al [1]. Until recently machine recognition of manual Morse code (its translation to printed text) eluded acceptable solutions, albeit not from lack of attention. A project in the mid-fifties at Massachusetts Institute of Technology's Lincoln Laboratories resulted in MAUDE (for Morse Automatic Decoder), one of the earliest machine translators of manual Morse code [2,3,4,5]. MAUDE and other early machine translators [6,7,8,9] were based on a small set of simple rules, both statistical and linguistic. The rules did not in any sense comprise a model of the language -- for instance English -- to be translated. These early attempts were restricted solely to translation. No attempt was made to incorporate into the systems an understanding of the information being sent, nor were aspects of the radio domain taken into account. A main premise of MAUDE and other early Morse code systems -- explicitly stated or implicitly assumed -- was that Morse code could be decomposed into a sequence of symbols comprised of the alphabet, the numerals 0 through 9 and punctuation marks. Since such a set is complete any sequence of Morse code could in principle be expressed by a sequence of these symbols.

Recently, a project at Massachusetts Institute of Technology's Laboratory for Computer Science has developed a new perspective on the manual Morse code problem. We have taken quite seriously the character of manual Morse code as expressed by the first sentence of this paper. Indeed our approach has been to include in a manual Morse code system extensive models or knowledge of the Morse, radio and natural language domains. Further, understanding of parts of a Morse code conversation is attempted, and it appears that it is necessary to understand some parts of what is transmitted in order to translate it correctly.

B. THE MANUAL MORSE CODE PROBLEM

The problem domain is that of hand-sent Morse code in an amateur radio network environment. To provide a global perspective of the Morse problem, a scenario of a typical amateur Morse-code network may prove helpful. A Morse-code network can be thought of as being composed of a hierarchical collection of sub-networks. A message in one sub-network may be destined ultimately for another sub-network in the hierarchy. A sub-network is typically composed of a group of operators and a controller (who is also an operator). The network controller's function is that of a traffic manager and general overseer to insure proper message flow within his sub-network. The network (we will henceforth drop the prefix "sub") becomes operational when the controller calls it to order. Each operator, on joining the network, communicates to the controller the number of messages the operator has to transmit, and to whom. The controller matches up operators and assigns them to a nearby (usually within 10 or 15 kilohertz) frequency to conduct their business. After the two members complete their communication, each reports back to the controller, indicating what traffic was transmitted as a check of completeness. At any time after being dispatched, a pair of operators may select a new frequency in a clearer area of the radio spectrum.

All negotiation between operators and controller is accomplished with the aid of a shorthand network protocol. Often the protocol is not followed precisely [10]. The protocol language is called "chatter," and it is composed of words whose generic type is "Q-sign," "Pro-

*Mr. Edward Black is currently employed by N.E. Berg Company in Bedford, New Hampshire; Mr. Timothy Anderson by Computer Corporation of America in Cambridge, Massachusetts; Mr. John Haverty by Bolt Beranek and Newman in Cambridge, Massachusetts; and Mr. David Sherry by the Xerox Palo Alto Research Center at Xerox Parc, Palo Alto, California

sign" or "Call-sign." The vocabulary of chatter includes about 100 essential words and as many as 900 others. The chatter language allows an operator to make statements, ask questions, and give orders. It provides a means for statement and query of operator identification, signal characteristics, rendezvous information, message traffic information, and so forth. Chatter contains an error recovery procedure for obtaining retransmission of either a word, the first character of each word in the message, everything after or before a particular word or phrase in the message, etc. The traffic that is communicated between Morse-code operators consists of messages, each comprised of a header, body, and signature. Header information typically contains from whom, to whom, handling instructions, precedence, number of words in the message, and other items.

1. The Radio Domain

There are a number of attributes of Morse code signals which we call radio domain attributes. Some of these attributes help but many hinder an operator's ability to translate Morse code. Unfortunately, even those attributes which helped operators perform translation heretofore were a bane to machine translators. Transmitter characteristics such as chirp are used to good advantage by an operator to separate a wanted signal from interference. Not only did machine translators not take advantage of such transmitter characteristics but such characteristics hindered the demodulation process to the extent that errors were introduced into the translation because of the characteristics. In addition to errors introduced by the demodulator's inability to handle transmitter idiosyncrasies the atmosphere adds its own characteristics which interfere with the Morse code signal; for example shot noise, fading, multipath, etc. make translation difficult for operators and machines alike.

(The current Morse code project did consider the problem of transmitter characteristics; but unfortunately, because of some contractual restrictions, was prevented from dealing with atmospheric characteristics.)

2. Sender Induced Irregularities

Morse code is composed of a sequence of alternating marks and spaces. In on-off keyed Morse code, a mark is characterized by the presence of an audible tone for some interval of time and a space by its absence for some interval of time. There are two types of marks: dots and dashes. Ideally, the duration of a dash is three times that of a dot. There are three types of spaces. They are given various names. Here we shall call them mark-space, letter-space and word-space. Mark-spaces are used to separate marks of a single character, and their ideal duration is that of a dot. Letter-spaces are used to separate adjacent letters, and their ideal duration is three times that of a dot. Word-spaces are used to separate adjacent words, and their ideal duration is seven times that of a dot.

The precision and correctness of manual Morse code is far from ideal, even if problems associated with the radio domain are omitted. Senders themselves induce three types of irregularities: (i) spacing-errors, (ii) mark-errors and (iii) spelling-errors. More than ninety percent of all sender induced irregularities in hand-sent Morse code are spacing-errors. Such irregularities occur when a sender does not keep the proper ratios between mark-spaces, letter-spaces, and word-spaces. The result is analogous to spoken language that is slurred or broken by arbitrary pauses. The segmentation problem in manual Morse code is analogous to the segmentation problem in continuous speech. Irregular spacing makes it a difficult problem. Hand-sent Morse code of plain English text often has a spacing error in each word of the message and some words (typically the long ones) may contain several spacing errors. The next most frequent sender induced irregularity is a mark-error. A mark-error occurs when a sender omits, adds or changes the sense of one or more of the marks (dots and dashes) making up a word. Our experience indicates that as many as twelve percent of the words in a Morse code message may contain mark-errors. Spelling-errors which can be so classified occur most infrequently. Certainly less than one percent of the errors are classified as belonging to this type, mainly because senders know the chatter language very well and because many common English spelling errors map directly onto a spacing-error or a simple mark-error, making spelling-errors indistinguishable from these other errors.

Needless to say, sender induced irregularities in manual Morse code proliferate to the extent that early machine algorithms such as MAUDE often produced translations that could be read only with great difficulty, especially if the reader had no knowledge of Morse code. Yet operators have little or no difficulty coping with the irregularities resulting from operator lapses, radio noise and interference and can translate a manual Morse code signal on which moving-threshold translators such as MAUDE would produce hopelessly garbled results.

C. DOMAIN MODELS

It is clear that good Morse operators have conceptual models of the Morse code environment that they use to help them perform their task. They have models of Morse "sounds" -- sequences of dots and dashes with rhythm and timing information -- and map these sounds into the letters and words. They have models of the language constructs that are used, be they English, another natural language, or the chatter language. Operators form models of other operators' idiosyncratic

mannerisms and use these models in the translation and understanding processes and in identifying other operators. Operators also have models of the Morse code and radio domains. It is common knowledge that "TH" is often sent with a short space between the letters, so that a machine often interprets it as "6" (thus "6E" is really "THE"). Similarly, "AN" is often interpreted as "P" (thus "PD" is really "AND"). In the radio domain, knowledge about where relevant operators are in the frequency spectrum, what a particular transmitter sounds like, how a signal fades and returns -- all these form the models that help operators identify, track, transcribe and understand one another. It is the human beings ability to interpret Morse sounds in the context of such mental models that allows her or him to perform so well.

At this point a slight digression is in order. Listening to a Morse code conversation among a group of operators, one notices three distinct aspects of the conversation. These correspond to (i) network chatter, (ii) message headers, and (iii) message bodies. The chatter section is often very poorly sent. Characteristically, many letters and words are slurred or separated and corrupted by other operator lapses. Yet receiving operators have little difficulty understanding chatter, because they have a model of the global situation: what question was asked by whom, who is currently waiting on the network, who has message traffic for whom, and so forth. This model and the ability to understand the conversation is vitally important to translation.

Headers of messages are structured but, unfortunately, not rigidly. Again, to translate them correctly one must have some understanding of what headers are about. For instance, dates may be sent as "3 Dec 78" or "3 12 78" or "31278" and times may be sent as "1000Z" or "1000". We have written the dates and times in an ideal manner, but, in fact, they might be -- as a result of operator lapses -- segmented quite differently, so that parts of numbers are run together or a number is split apart, and one or more numbers might contain a mark-error. One other aspect of numbers is very important. All numbers in Morse code are five marks long -- see [11] under Morse code -- yet they are often abbreviated and sent as what are called cut-numbers. Again, context is often required to perform translation correctly.

The body of amateur radio message traffic is typically English with some abbreviations. To attempt to understand all of the English language would be far beyond the scope of this research. We have built into the system just enough knowledge to let it perform in a creditable fashion. Our experience indicates that a vocabulary, some rules about where numerals can occur in text, rules about how to handle error signs, and a measure of closeness in a Hamming-like space (for correcting operator induced irregularities) are absolutely essential to the correct translation of plain text. Given those sine qua nons, our experience suggests that knowledge about idiosyncratic and irregular behavior of individual operators facilitates translation.

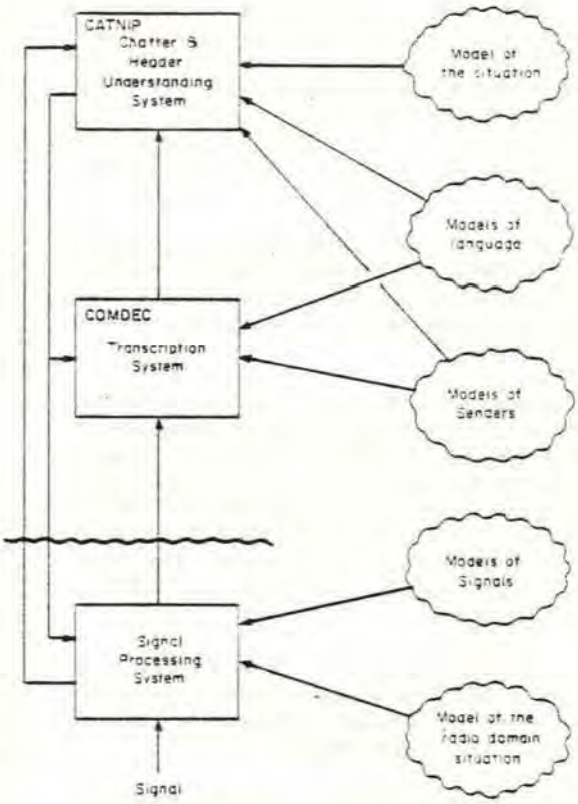


Figure 1. The Three Major Modules of the Morse Code System and the Domain Models They Use.

Military message traffic differs slightly because the body may consist either of plain text or of cipher groups. A message sent in cipher has no language context, but knowledge about the number of groups in a message, the number of characters in a group, and whether groups are alphabetic, numeric, or mixed is necessary to translate such a message.

Figure 1 shows a block diagram of the three major modules of the Morse code system under development by the authors. Also shown are the necessary domain models required by each module in order for it to perform its task properly. The wavy line in the diagram indicates that the signal processing system, which is composed of special hardware and a PDP-11 computer, is not integrated with the other major modules which are COMDEC, the transcription (or translation) module, and CATNIP, the chatter and header understanding module. The last two are software modules written in MDL (a LISP-like language) [12] and running under TOPS-20 [13] and ITS [14]. Experiments are conducted independently

for the signal processing system, and human intervention is required to transfer the results to the other two modules. COMDEC and CATNIP are well integrated, with appropriate feedback, and externally they appear to behave as one system.

A few phrases about each domain model may prove helpful:

1. Model of the radio domain situation -- how the individual transmitters of interest sound, i.e., whether a transmitter has any characteristic envelope or carrier distortions, and if so what kind and a measure of the amounts.

2. Model of the Network situation -- which operators are logged into the network, which are off control frequency, which are on control frequency and where each operator's transmitter is tuned relative to those of the other operators on his frequency. This last bit of information turns out to be quite important, as we will show later, even though all the operators are working in a thirty to fifty Hertz band.

3. Models of senders -- the irregularities a particular sender may introduce, such as his or her idiosyncrasies of language, a proclivity to introduce extraneous dots or omit dots, etc.

4. Models of language -- the full gamut of possibilities are required for parsing and understanding chatter, but we have rather simple models for handling message bodies such as a vocabulary and some simple rules for handling some special constructs and numbers.

5. Models of the situation -- the system must know when a question is asked and the possible range of expected answers; it must know that a frequency change has been ordered or negotiated and how to respond appropriately; and so forth.

D. THE MORSE CODE SYSTEM

As mentioned, the Morse code system is composed of three major modules and some domain models. We will attempt in this section to present a short explanation of each of the major modules.

1. Signal Processing System

We believe that an interesting and important development in the signal processing area of the Morse code project was the implementation of a novel tandem phase-lock-loop filter that utilizes time reversal of the input signal. Despite the use of time reversal, the output can be obtained in real time, albeit with a constant delay. The nature of Morse-code signals -- the fact that they are on-off keying or frequency-shift keying -- and the fact that initial experiments indicated that the transient response of the phase-lock-loop filter interfered with the measurement of important signal parameters, led to the development of the novel filter.

There is a great deal of information contained in the audio sound of a Morse-code signal -- the signal characteristics per se -- besides the timing information of the marks and spaces. It became clear while running some experiments in understanding Morse-code network conversations that the signal characteristics contained information that was an important part of the context of the situation; it is necessary to extract this information in order to understand the network conversations (q.v.).

A small digression is required to explain what a human operator hears in the sound quality of the signal in order to understand what must be extracted from Morse code signals. Briefly, a human operator is capable of detecting and tracking certain signals in a crowded spectrum of similar competing signals. Note how one can follow a particular conversation at a crowded cocktail party. One can do fairly well even with one ear. There is no binaural effect in the Morse-code domain. This discriminating ability of human beings is evidently knowledge-based. To discriminate signals, an operator uses information about how the signal sounds: (a) its frequency; (b) its anticipated frequency drift; (c) its amplitude and rate of chirp, if any; and (d) the amount of envelope distortion such as hum, clicks, yoop and whatever other characteristics of the waveform can be characterized. A good signal-processing front end should be capable of measuring some, if not all, of the above signal characteristics and of using the measured characteristics for signal discrimination.

a. Tandem Phase-Lock-Loop

The general requirements can be translated into specific requirements of a receiving filter process for the Morse-code application. (The specific filter design is for an on-off keyed signal, and experiments were conducted only with such a signal. Therefore, the discussion that follows is in the context of on-off keyed signals. However, it should be pointed out that similar arguments can be made, and similar results can surely be obtained, for the case of frequency-shift keyed signals.) Extracting the on-off timing information for marks and spaces as well as signal quality information requires determination of the transitions of the signal as well as continuous estimation of the amplitude and frequency of the signal. The latter information serves a dual purpose. First, it is used to characterize transmitter signals for use in transmitter recognition. In addition, the frequency on which a station is transmitting is part of the situation model, and an uncharacteristic frequency shift of ten or several tens of hertz often indicates a change of

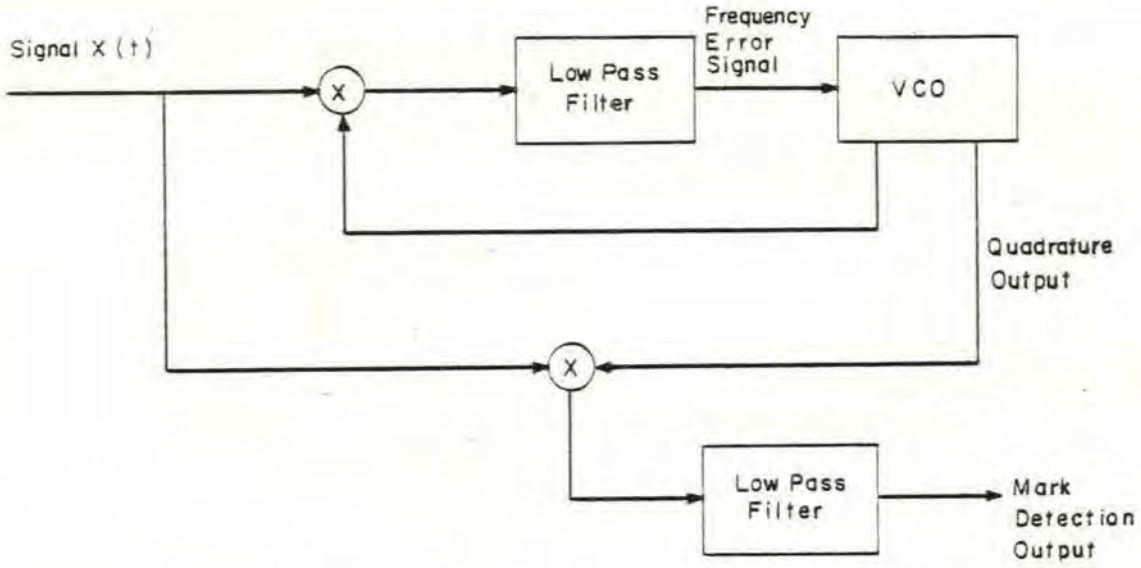


Figure 2. Phase-Lock-Loop Detector.

"sender" during network conversation. This type of cue is extremely useful, because, after contact has been well established, operators often do not re-identify themselves. Furthermore, without the change of sender context information, some chatter constructs are ambiguous.

The tracking-filter model of the phase-locked loop (PLL), Figure 2 [15, 16], is well suited to extracting the information indicated above from a signal in that it gives continuous frequency and amplitude estimates, and presents a relatively narrow-band filter to a frequency-modulated carrier.

However, before a PLL can give accurate demodulation, it must achieve lock. (Lock is the state of the PLL when the voltage controlled oscillator (VCO) tracks the incoming signal with a constant phase lag.) The time to achieve lock is inversely proportional to the natural frequency of the loop, and is affected by such factors as the initial frequency error (difference in frequency between the VCO and the input) and noise in the loop.

Chirp is frequency modulation which frequently occurs in low quality transmitters. It is often caused by inadequate filtering of the power supply, which causes the oscillator to change frequency when the power stage is turned on. Thus, the frequency modulation, or chirp, exists where the signal makes a transition from off to on, or vice versa. Most often, it exists only at the beginning of the "on" period or "mark" of the Morse-code signal. Unfortunately, in a traditional PLL arrangement, or for that matter any type of traditional filtering, the transient response of the filter is superimposed on the signal and is largest at the signal transition points. The problem is exacerbated when interference is considered; as one narrows the bandwidth of the filter to eliminate the interfering signals, the period for which the filter transient response is a significant factor in the output is lengthened. In the case of the PLL, the transient between acquisition and lock at the beginning of the signal is the major one, because a PLL will track the signal during the on-to-off transition until it reaches a signal-to-noise ratio at which the signal is lost. Thus, because the frequency and amplitude estimate of the signal prior to lock contains important information, i.e., the chirp information and the time at which the mark began, it is desirable to reconstruct that portion of the signal.

A number of ways of recovering the pre-lock information can be conceived. The method settled upon is simple. It involves sampling and storing the input to the PLL, and then, after the PLL has completed processing the mark in the forward direction, sending the stored samples in reverse order through the loop. The loop then demodulates a time-reversed replica of the original signal, and the original leading-edge information is reliably obtained from the trailing edge of the reversed signal.

Because it was desirable to run the process in real time, only the beginning portion of the signal is reversed and a second PLL is used to demodulate it so

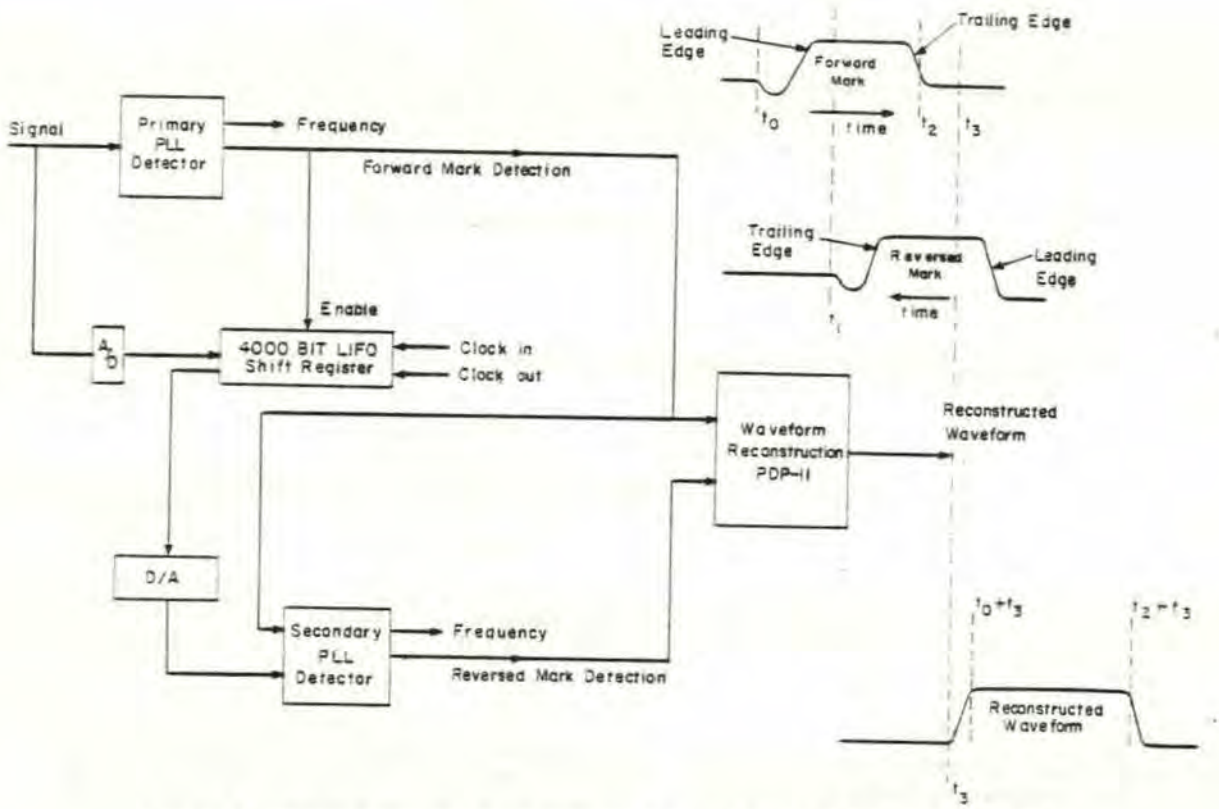


Figure 3. Tandom Phase-Lock-Loop Detection System with Wave Form Reconstruction.

that the first PLL can continue to demodulate the forward signal. In addition, the reversed signal can be compressed in time by sending the reverse-order samples through the secondary PLL at a rate faster than they were collected. Of course, the secondary PLL needs to run at a higher frequency. Thus, it is possible to have reconstructed the mark before the next mark begins.

The tandem phase-lock-loop system is shown in Figure 3.

The input is sampled and stored in a last-in-first-out (LIFO) memory. Meanwhile the input to the secondary PLL is taken from the quadrature phase of the primary PLL. When lock is indicated (by the quadrature, or correlation output of the primary PLL) the input of the secondary PLL is switched to the digital-to-analog converter (DAC), and the stored samples are read out in reverse order. The outputs of the secondary PLL are taken as the demodulated signal for the time prior to lock. The outputs of both PLL's are sampled and a PDP-11 program reconstructs the waveform (mark).

2. COMDEC: The Morse code Translation System

The COMDEC (computerized Morse Decoder) control structure was designed to consist of any number of modules, each of which would be an "expert" on one aspect of translation. Each module can add suggested translations to a lattice of possible translations. Successive modules decide whether further error correction is necessary by examining the quality of existing suggested translations in the lattice, then pass that section of code to the next module in the chain, and so on, until each translator module has examined the entire message. A major part of the design is an N-dimensional metric for measuring trial translations and a heuristic algorithm for comparing the measures.

Translator modules are ordered approximately in terms of increasing severity of the sending errors they are able to correct. Thus, spacing-errors are corrected first. Later, mark-errors are corrected.

The first module to process a code sample is a moving-average translator, a hybrid of MAUDE [2] and FRAUD [7]. It classifies marks and spaces into the

appropriate types by comparison with continuously updated averages for each type and thresholds between types. More importantly, it associates with each mark and space a number which represents its confidence that the classification was correct. These assignments and confidences are used by later modules to select areas of the sample where errors appear to have occurred.

a. Handling the Segmentation Problem

The performance of moving threshold Morse code translation systems suffered quite severely from the effects of improper segmentation of Morse code sequences. Morse code translators suffer from two types of segmentation difficulties: first, character-spaces and mark-spaces are confused such that two characters are sometimes decoded as one, e.g., 'AN' as 'P', and vice versa; second, word-spaces and character-spaces are confused such that 'THEY ARE' is translated as 'THEYARE' and 'HOLD' is translated as 'H OL D'. The DEGARBLER [17], a program based on the use of digraphs and trigraphs found in English words and a dictionary of words, was capable of correcting many of the word-segmentation errors that occurred in MAUDE output, but it could not obtain the proper transcription of Morse sequences that had character-segmentation errors present.

Because words corrupted only by segmentation errors contain a correct sequence of marks, we realized that words can be represented by the "run-length sequence" (RLS) of the marks. The run-length sequence is a string of numbers, composed of the number of consecutive dots which begin the word, followed by the number of consecutive dashes which follow those dots, and so on. (By convention, a run-length sequence always begins with the number of dots, even if the number is 0.) The principle is simple. All mark and character spaces are removed from the representation. Because spacing errors occur frequently -- one or more per word -- a major source of difficulty is removed. Basically, the RLS is a mapping of words in the Morse-code domain onto some space. Each word along with its variants, obtained by taking all possible combinations of character segmentation errors, are mapped to the same point in that space.

The reverse mapping of run-length sequence onto legal words is not necessarily unique. However, better than 85% of the RLS for a dictionary of over 10,000 words are unique and most of those that are ambiguous have but a few members in their ambiguity sets. The ambiguity resulting from representing words by their RLS is not a significant problem, because often the simple heuristic of comparing all the words with a specific RLS to the MAUDE transcription of that segment of Morse code is sufficient to determine the one the sender intended.

To understand how the RLS is used, consider the following. The sequence of dots and dashes for the word "the" was sent with spaces such that a context-insensitive decoder could not transcribe it into a proper word, i.e., the sequence sounds like '6E'. A general heuristic that one might think of adopting is this: find all the possible words which that sequence of marks could represent and pick the one that seems to fit best according to some criterion. If one tried to implement such a heuristic in a brute-force manner, 2**5 or 32 possibilities would need to be considered, as the word "the" (- ___ .) has three mark spaces and two character spaces for a total of five spaces. Thus 32 permutations and look-ups would be required to find "the" and any other words with that mark sequence. On the other hand, using the RLS representation with no permutations and one look-up, "the" and all other legal words with that RLS are returned. Consider the word "complex", not an unusual or overly long English word. If it were sent with a spacing error, and a brute-force method were used to obtain possible candidate words, the possible spaces associated with the sequence of dots and dashes would require more than two million (2**21) permutations and look-ups, only to discover that in the dictionary currently in use only the word "complex" could be represented by such a sequence of marks. Again, the RLS representation would return the word "complex" in one look-up.

The use of the RLS solves only the character segmentation problem. Word segmentation is handled by a heuristic that splits or joins character sequences. The heuristic uses the confidence levels returned by the MAUDE decoder to determine how to proceed. (For more details on RLS translators, see [18, 19, 20]).

b. Handling Mark Errors

The mark-error correction modules are able to correct eight different classes of mark-error in words which contain at most one mark error. Statistically, the vast majority of mark errors are of the types COMDEC can correct, and occur only one to a word. The types of mark errors corrected by COMDEC are (i) sending an extra dot, (ii) sending an extra dash, (iii) sending two extra dots, (iv) running two dots together as a dash, (v) splitting a dash into two dots, (vi) dropping a dot, (vii) dropping a dash, and (vii) dropping two dots.

These mark-errors are easily correctable because words with these classes of mark-errors map to nearby points in the RLS space and the space is very sparsely populated for containing eleven marks or more.

c. Handling Error Signs and Number Constructs

When a sender recognizes that he or she has made an error, he or she will resend the erroneous word, phrase, or sentence, signalling with an error sign that

retransmission is about to occur. This behavior is somewhat analogous to a typist who spaces back over an error and overstrikes it with X's. An error sign is usually a sequence of dots sent rapidly, rarely fewer than six, and rarely more than twenty. The number of dots sent varies even within a single transmission, as does the separation of the dot-sequence from the erroneous code preceding it and the "correct" code following it. More importantly, the semantics of an error sign vary even more widely. Thus it is necessary to locate and to ascertain the meaning of an error sign. An error sign may mean to ignore the previous word or characters, or it may mean that the previous word or phrase will be resent, and so on. Some examples from actual code (with the symbol "@" used to represent an error sign) follow:

ANY B OY OR GIRL 13 TO 10 @ 19 WHO

The correct translation:

ANY BOY OR GIRL 13 TO 19 WHO

This is the most typical use of an error sign. It signals that the previous word or object was in error, and the sender resends the word correctly. The error sign in this example contained thirteen dots.

PAGE B 23 OF TODAYS PE TPERS @ TODAYS PAPER

This is similar to the previous example, but two words are erased and resent. The error sign contained eleven dots.

THE CORNER OF WASHINGTON
BLVD @ AND SCHOOL STREETS

In this example, the word "BLVD" is erased -- it should not have been sent at all. The error sign contained seventeen dots.

d. Handling Numbers in Plain Text

COMDEC recognizes arbitrarily long sequences of digits as "numbers".

The problem of transcribing numbers is analogous in some ways to that of transcribing error signs, and it arises from the fact that most of COMDEC's transcribing is vocabulary-based. Since it is theoretically possible to send a number containing an arbitrary number of digits, it is impractical to use a "dictionary" of numbers. Instead, COMDEC takes advantage of the properties of the Morse code used to represent the digits: (i) All digits consist of five marks (except cut numbers which are also handled). (ii) Every digits contains a sequence of dots followed by a sequence of dashes, or vice versa. (iii) A number very often appears in context, for example, as a part of a date, time, address, page number, or age specification. This context is used to reinforce the probability that the item is a number. A number appearing out of context must be allowed, as all possible contexts have not been or cannot practically be implemented. If a number appears out of the contexts in which a number is expected, it is looked upon by COMDEC with suspicion, and it will be allowed to remain a number only if it is relatively well sent.

COMDEC searches for mark sequences that fit these criteria and then attempts to "expand" them on either side (to produce complete numbers). The only limitation on this algorithm is that at least one digit of an n-digit number must be sent correctly.

e. The COMDEC Dictionaries

COMDEC's dictionary for plain text messages is comprised of 4200 root words. A morphology program embedded in the translator handles a large but not complete set of word endings. Thus, a word usually appears in the dictionary only in its root form. The current morphology program understands the endings "-s," "-ed," "-ing," "-er," "-est," "-ly," "-tion," "-ment," and combinations of the preceding, such as "-ers." The dictionary specifies the endings taken by each word. When endings are considered, the effective size (about 18,000 words) of the dictionary is several times the number of its roots and it approaches the size of the conversational vocabulary of the average English speaker.

3. CATNIP Chatter and Header Understanding System

CATNIP is a semantic-syntactic augmented-transition-network (ATN) parser that chooses a path through the lattice of possible translations created by COMDEC.

CATNIP uses ATN diagrams to choose the correct word from a lattice of possible translations. It starts in a certain state of the transition network, and progresses from one state to another, depending on the next word or words in the lattice. With each state is associated a list of words, and with each word a new state. CATNIP matches the list of words from the state with the list of words possible at that point in the translation lattice, matches yield valid new states.

If that were all, the network would simply be an unaugmented transition network. However, CATNIP retains a context, which it changes (usually with every word) and which can be tested when it is trying to match the words. The context includes such things as who is the sender of the current transmission, who is the receiver, who is the net controller, and so on. ATN's, as opposed to unaugmented transition networks, are good for parsing grammars that are dependent on the context and on past occurrences [21].

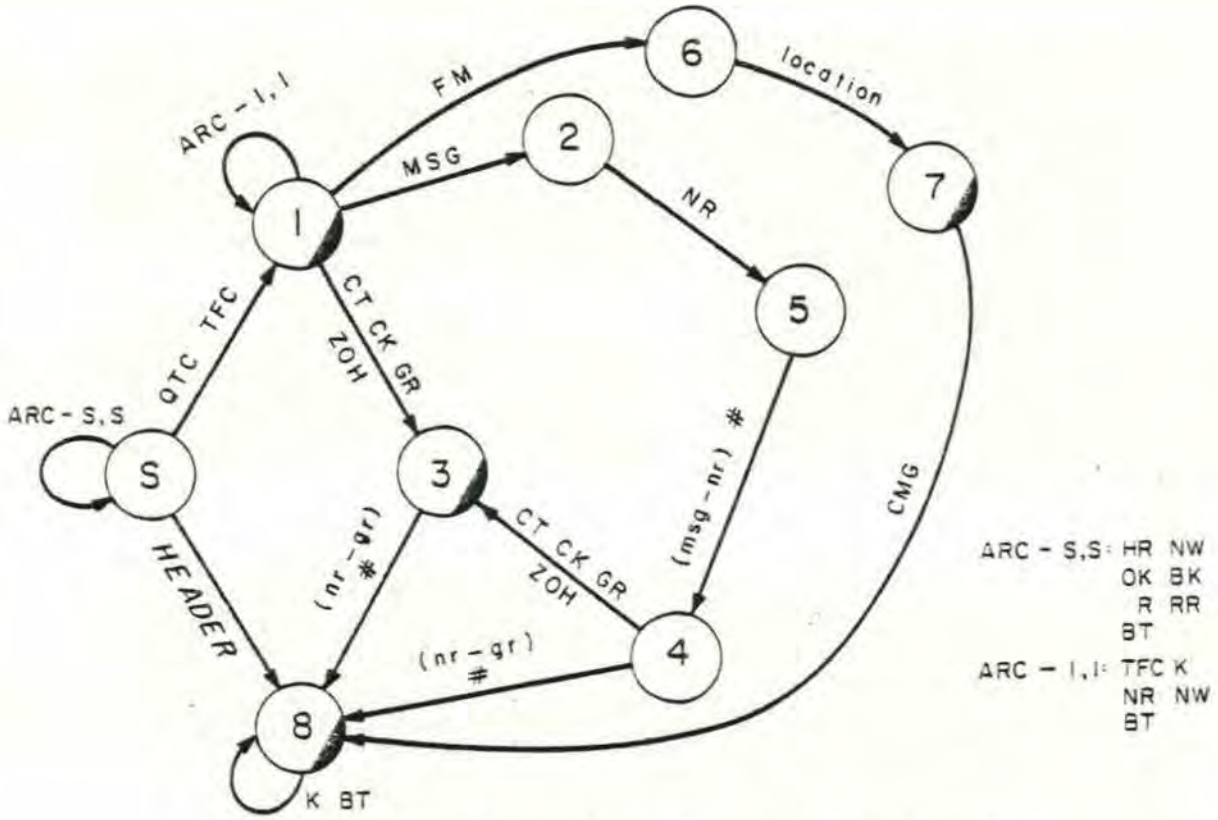


Figure 4. Augmented Transition Network Diagram Called Traffic Header.

Naturally, ambiguities creep in. Sometimes more than one match is possible; CATNIP allows for this by processing one of the new valid states and saving all the others. The context at that point is saved with the states that were saved. CATNIP has the ability to return to the saved states and try those alternate paths.

Finally, CATNIP also has a limited understanding of the events on the net. Understanding these events is important in understanding the state of the net at any point (how many operators are working, who they are, who is talking, etc.) and it is important in choosing the correct word at a particular point in the translation.

The context is used as the "understanding" part of CATNIP. Take the following transmission as an example: "ROCK ROCK ROCK DE SALT SALT QSA ? QRK ? K". Upon completion of the parse, the parser would retain a context that contained the information that the receiver was ROCK, the sender was SALT, and SALT had asked ROCK two questions: "What is my signal strength?" and "What is my intelligibility?"

Retaining this kind of context helps find the right translation and decide later ambiguities (such as who is the receiver at a certain point, if he or she was not explicitly named). The successive contexts also furnish a synopsis of the entire session after the parser is finished.

CATNIP is a recursive procedure that allows one to name ATN diagrams of simple structures (such as Q-signs that are often used), and to use those as parts of other diagrams without actually duplicating the simple diagrams. Thus a more structured "grammar" can be created without over complicating the data base.

Figure 4 shows a typical ATN diagram. Each italicized ARC label such as "Header" indicates a call to another diagram; in lower case are labels such as "location", which indicate that the labeled input that will parse is a location (such as "BOSTON" or "BOS"); arcs labeled with a number sign "#" mean that a number is acceptable as input (with the parenthesized statement indicating the meaning of the number, e.g., (nr - gr) means number of groups); and arcs labeled in upper case mean that literal input of one of the specified labels is acceptable. The system currently contains about 25 diagrams with an average complexity of the one shown in Figure 4.

E. RESULTS

Two sources of test data were used. We obtained approximately 30 thousand characters worth of data from amateur operators and others who could send and receive Morse code. This data was hand-keyed in our laboratory. Some of this data was obtained using actual transmitters connected to our cable network [22]. This data was composed primarily of plain text with a small admixture of network chatter. Chief Warrant Officer Robert Bolling at the Army's Morse code school at Fort Devens arranged to have several instructors at that school simulate an actual network environment. From that simulated environment we received audio tapes containing approximately 20 thousand characters of hand-sent Morse code. This data was composed of chatter, headers, plain text messages and cipher group messages. It was the most useful data in that it forced us to face many problems and constructs we had not encountered previously: error-signs, numbers in plain text and many different types of chatter constructs.

We measure the performance of our Morse code system in two ways. First we compare the final results with the results produced by the first stage of our translator, which is a MAUDE-like moving threshold translator. Second we compare the final results with an experienced operator's translation. The experienced operator had the use of a variable speed tape recorder. He at times found it necessary to replay parts of the tape, occasionally at slower speeds.

Figure 5a shows a translation by a MAUDE-like translator of one of the plain text messages contained in the Fort Devens data. While it can be read, especially by a person who knows Morse code, a considerable effort is required. Figure 6a shows the same message as translated by COMDEC, our Morse code system's

```
IN ANATT E MPT TOALLEVE EE(.....)ALLE4IATE YOUTV UN E MPLOYMENT I N THE CI
TY MIM T (.....) CI TY, TSE SUNDAYGLOBE ON MAY30 WILL PUBLIS5 FRE E
ADVERTIS E MENTS FOR GOSTON T E E NAGERS S E E KING SUMMER JOBS. ANY B OYORGIRL
13TO 10 (.....)19 WSOLIVES IN BOSTONCANP LACE AJO6 WANTED AD WI THOUT
CAAEEEEEEEEEEEEEEE CH#GE 6YFILLING OU T THE C(---..-)PON ON PAGE B 23 OF TODAYS
PETPER(.....)(.....) TODAYSPAPER ANDMAILING I T TO SUMMER JOBS (--..-)
TSE BOS(----)N GLOBE . BOSTON . MAS S AC S EEEEEEEEEEE MAS SACSUSE T TS 02107.
TE E N(.---.)ERS MAY ALSO TAKE TS E COUPONS T O TSE GLOBES OOWNTOWN OFFICE . A
T TSE C ORNER OF WASHINGTON GLVO EEEEEEEEEEEEEEEE AND SCH OOL STRE E
TS.ORATITS MAI N OF FICE . 13 (.....) MORRISS E Y BLVD. OORC4 E STER A(.-.-)
COUPTANS (--..-)S T B E RE CEIVED (-.....)Y EEEI P M ,WEDNESDAY M(..--)
MAY(..--) T EEEEEEEEEEE . MAY2T5.JOBSE E KI EEEEEEEEEEE JOBSEEKERS MAY BE
&SPECIFICAS TH E Y L IKE I N ME NTIONING TSE SO(..-.-.)S OR OAYS TSEY AR E
AVAILABTIIE(.....) AVAILABLE FOR E MPLOYMENT. THE TYPE OF WTMRK 6 E YOES IRE
(---.-.)CANOO(---.-.)WV AT W(.---.) E S TH E Y E XPECT. THE
(.--..)SWILLAPPE#INTSE MAY30CLASS IFII EEEEEEEEEEEEEEEET CL&SISE I E O SECTION
UNDER EH E EHE (.--..)ING 5I EEEEEEEEE HEADING S IRE A8OSTON T E E NAGE RFOR TH
E SUMMER .
```

Figure 5a. MAUDE Translation of Plain Text.

```
IN AN ATTEMPT TO [xxxxx e] <ALLEVIATE> <YOUTH> UNEMPLOYMENT IN THE [xxxxx e]
CITY . <THE> SUNDAY GLOBE ON MAY 30 WILL <PUBLISH> FREE ADVERTISEMENTS FOR
<BOSTON> TEENAGERS SEEKING SUMMER JOBS . ANY BOY OR GIRL 13 TO [xxxxx e] 19
<WHO> LIVES IN BOSTON CAN PLACE A <JOB> WANTED AD WITHOUT [xxxxx e] CHARGE <BY>
FILLING OUT THE COUPON ON PAGE B 23 OF [xxxxx e] TODAYS PAPER AND MAILING IT TO
SUMMER JOBS <.> <THE> BOSTON GLOBE . BOSTON . [xxxxx e] <MASSACHUSETTS> 02107 .
TEENAGERS MAY ALSO TAKE <THE> COUPONS TO <THE> GLOBES DOWNTOWN OFFICE . AT
<THE> CORNER OF WASHINGTON <BLVD> [e] AND SCHOOL STREETS . OR AT ITS MAIN
OFFICE . 135 MORRISSEY BLVD . <DORCHESTER> . <COUPONS> MUST BE RECEIVED THEY S
PM . WEDNESDAY [xxxxx e] . MAY <26> . [xxxxx e] JOB SEEKERS MAY BE AS SPECIFIC
AS THEY LIKE IN MENTIONING <THE> <HOURS> OR DAYS <THEY> ARE [xxxxx e] AVAILABLE
FOR EMPLOYMENT . THE TYPE OF WORK <BE> (Y) DESIRE OR CAN DO OR <WHAT> WAGES
THEY EXPECT . THE ADS WILL APPEAR IN <THE> MAY 30 [xxxxx e] <CLASSIFIED>
SECTION UNDER <HE> [xxxxx e] HEADING <HIRE> A BOSTON TEENAGER FOR THE SUMMER .
```

Figure 6a. COMDEC Translation of Plain Text.

translator. An "@" in square brackets indicates an error-sign; "XXXXX" preceding the "@" indicates that a portion of the message, believed to be the part in error, was suppressed; and "< >" indicates a word obtained from the dictionary by assuming the sender made a mark-error. The COMDEC translator made four errors: transcribing "BY" as "THEY", "THEY as "<BE>", "THE" as "HE"; and not suppressing "BLVD" after the word "WASHINGTON". To handle the last error correctly the translator would need to know that in Boston, Washington is a street and not a boulevard. Less than three percent of the words in the message were in error. The operator made 33 errors in translating this message: a 20% error rate. Both COMDEC's result and the operator's result are quite readable.

Figure 5b shows a translation by a MAUDE-like translator of some chatter, a header and a portion of a ciphered message. Figure 6b shows the identical portion of Morse code as translated by the cooperative efforts of COMDEC and CATNIP. Both the operator and the COMDEC-CATNIP translations of the chatter and header were perfect. (The "{ }" at the beginning indicates that the enclosed sequence cannot be translated. Consequently the translation obtained from the moving threshold first stage translator was used. (This construct resulted from a splice in the audio tape.)

The COMDEC-CATNIP tandem translated 19 groups incorrectly and the operator 32. It is interesting that both the operator's translation and and the machine's translation had 13 incorrectly translated groups in common. In addition to translating groups incorrectly both the operator and the machine added 9 extra groups because the semantics of the error-signs were ambiguous.

One very encouraging aspect of the system is that on a DEC 20 it is 2 to 10 times faster than real-time, depending upon the number of irregularities and errors in the Morse code.

One should not draw hasty conclusions from these results. We have worked very hard to obtain the results we have but the job is not yet complete. For instance, the chatter and header constructs preceding the plain text message of Figure 6a are not yet understood by CATNIP. Furthermore, a different set of operators using

```
"MAUDE" decoding:

SALT:   (..-VVVV) (VVV) QSA? K

ROCK:   RRRRQSASQSAS K

SALT:   RAE (..-.-.) METSA 5 EAEN METS AS UEN METENK 5 GTTC MATC TN A? K

ROCK:   RRRQRVK

SALT:   RRR (-..--) NWHR TFC HR TFC = = = NR 1 GR 200 044e = VDLVR OZQNT
        PQTOR TCY(..-....) VLHAP XGJIN NVRLC TJOMM (V.--.-..)AG OTZTZ
        PQDIH QRCNAGZOC E LYOE HSS GQOTR# H L(-.---..)OE B TXTB 01e 32
```

Figure 5b. MAUDE Translation of Chatter, Header, and Parts of a Ciphered Message.

```
COMDEC decoding:

SALT:   {(..-VVVV)} VVV QSA ? K

ROCK:   RR RR QSA 5 QSA 5 K

SALT:   RR UR QSA 5 UR QSA 5 UR QRK 5 QTC QTC GA ? K

ROCK:   R RR QRV K

SALT:   RR R NW NW HR TFC HR TFC [BREAK]
        [BREAK]
        [BREAK]
        NR 1 GR 200 <0445> [BREAK]

        VDLVR OZQNT PQTOR TCYUH VLHAP
        XGJIN NVRLC TJOMM VWLAG OTZTZ
        PQDIH QRCNA GZOCE [xxxxx e] GQOTR [xxxxx e] LYOOE BTXTB 01532
```

Figure 6b. COMDEC Translation of Chatter, Header and Parts of a Ciphered Message.

a different set of chatter conventions might require a significant amount of effort to augment the domain models to understand and translate the new chatter. Yet a human operator could adapt to the new operator's style and acquire knowledge about the new chatter conventions by listening to the conversations for a few hours or at most a few days.

F. ACKNOWLEDGMENT

This research was supported by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract number N00014-75-C-0661. We are grateful to J.C.R. Licklider for drawing our attention to the Morse code problem. We also wish to thank Stuart W. Galley and Janet L. Schoof for their editorial assistance.

G. REFERENCES

1. Freimer, M., B. Gold and A.L. Tritter. The Morse Distribution. Cambridge, Ma.: M.I.T. Lincoln Laboratory, January 1959.

2. Eisenstadt; Gold; Nelson; Pitcher; and Selfridge. "MAUDE (Morse Code Decoder)." Cambridge, Ma.: M.I.T. Lincoln Laboratory Group Report 34-57.

3. Gold, Bernard. Machine Recognition of Hand-Sent Morse Code. Cambridge, Ma.: M.I.T. Lincoln Laboratory.

4. Byrnes; Gold; Nelson; Pitcher; and Selfridge. "Some Results of the MAUDE Program." Cambridge, Ma.: M.I.T. Lincoln Laboratory Group Report 34-72.

5. Gold, B. "Machine Recognition of Hand-Sent Morse Code." Transactions of IRE, PGIT, IT-5, 17, (1959).

6. Smutek, John M., and Richard S. Gawlick. A Digital System for Recognising and Translating Morse Code. May 1968.

7. Poehler, P.L. "Computer Recognition of Hand-Sent Morse Code." S.M. Thesis, M.I.T. Department of Electrical Engineering and Computer Science, July, 1968.

8. Petit, R.C. "The Morse-A-Verter -- Copying Morse Code by Machine." QST, Vol. IV, No. 1:30-35, January, 1971.

9. Guenther, J.A. "Machine Recognition of Hand-Sent Morse Code Using the PDP-12 Computer." Air Force Institute of Technology, Wright-Patterson Air Force Base, December, 1973.

10. American Radio Relay League. The Radio Amateur's Operating Manual. ARRL Publication #24, Newington, Conn., 1972.

11. Webster's New Collegiate Dictionary. Springfield, Ma.: G. & C. Merriam Company, 1974.

12. Galley, S.W. and Pfister, Greg. MDL Primer and Manual. Cambridge, Ma.: Massachusetts Institute of Technology, 1977.

13. DECSYSTEM-20 User's Guide. Maynard, Ma.: Digital Equipment Corporation, 1978.

14. Eastlake, D.; Greenblatt, J.; Holloway, J.; Knight, T.; and Nelson, S. ITS 1.5 Reference Manual. Cambridge, Ma.: Massachusetts Institute of Technology, 1969.

15. Van Trees, Harry L. Detection, Estimation, and Modulation Theory; Pare II: Nonlinear Modulation Theory. New York: John Wiley and Sons, Inc., 1971.

16. Gardner, Floyd M. Phase Lock Techniques. New York: John Wiley and Sons, Inc., 1966.

17. McElwain, C.K., and M.B. Evens, "The Degarbler -- A Program for Correcting Machine-Read Morse Code," Information and Control. Vol. 5 (1962), pp. 368-384.

18. Anderson, T.A., "Machine Recognition of Hand-Sent Morse Code," M.I.T. Laboratory for Computer Science, Programming technology Division Document, SYS. 17.00, July, 1975.

19. Lebling, P.D., and Haverty, J. "Improvements to COMDEC." M.I.T. Laboratory for Computer Science, Programming Technology Division Document, SYS. 17.01, July 30, 1975.

20. Massachusetts Institute of Technology's Laboratory for Computer Science, Progress Report XII, July 1974 - July 1975, Cambridge, Massachusetts.

21. Woods, William A. "Transition Network Grammers for Natural Language Analysis." Communications of the Association for Computing Machinery. Vol. 13 No. 10, 1970.

22. Massachusetts Institute of Technology's Laboratory for Computer Science, Progress Report XIII, July 1975-1976, Cambridge, Massachusetts, p. 185.

THE POSITIONING PROBLEM - A DRAFT OF AN INTERMEDIATE SUMMARY

Yechiam Yemini
USC/Information Sciences Institute

AN INFORMAL INTRODUCTION

The positioning problem arises when it is necessary to locate a set of geographically-distributed objects using measurements of the distances between some object pairs. In a Packet Radio Network, for instance, any two network members that can talk to each other may use a simple time-stamping mechanism to measure the distance between them; a distance measurement protocol may then be developed. The problem is whether and how the distance measurements can be used to determine the geographical location with respect to a given system of coordinates.

A knowledge of the precise location of each network node is crucial to the operation of Distributed Sensors Networks. The data collected and interpreted by different sensors may be correlated and integrated only if we know their precise location. A position-locating system may be invaluable to the operation of a fleet of vehicles, each equipped with a Packet Radio Unit. For example, monitoring the location of a fleet of security vehicles, aircraft, a tank division, or a flock of missiles could all be assisted by a position-locating system. Clearly a positioning system would be an important service to Packet Radio Network users.

A few problems must be solved before a good positioning system may be developed:

1. Efficient algorithms to determine the location of objects by using distance measurements should be developed.
2. Conditions under which a solution exists or does not exist should be identified.
3. Conditions under which there exists a unique solution should be established.
4. Conditions under which there exists a finite number of solutions should be identified. It should also be understood how to transform one solution into another.
5. Conditions under which the solution is insensitive to small measurement errors should be established.
6. Tight bounds upon the accuracy of the solution should be determined.
7. Ill-conditioned problems should be identified.

However, while the formulation of the problems is simple, the mathematical and algorithmic intricacies of deriving solutions are perplexing.

To develop some insight into the problems, let us consider a few simple examples. The simplest positioning problem of interest is to locate three points using distance measurements. By means of simple trigonometry, this may usually be done easily. However, let us consider a degenerate triangle (Figure 1). Because the system is very sensitive to errors, a small error in the measurement may produce a large error in the computed position. Some of the questions to be addressed are as follows: Why is the degenerate triangle sensitive to errors? How can we determine whether or not other systems are sensitive?

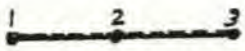


Figure 1. A Degenerate Triangular System

Positioning systems may be constructed by a simple procedure of pasting triangles together, and such systems may be positioned by solving the triangles from which they are constructed. For instance, consider the system of points depicted in Figure 2. It is possible to locate the points in the order numbered. However, the same system admits a few solutions (the number of which grows exponentially with the number of nodes). If we had some further information about the positions of the objects, how could we use it to identify the true solution? For instance, if one node is known to be a vehicle moving on a certain road, many of the feasible solutions that satisfy the distance constraints can be eliminated, because they assign the vehicle to a position not on the road. How should that elimination be effected?

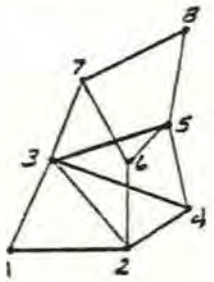


Figure 2. A Triangulated Positioning System

In particular, if a new measurement were given (Figure 3), how can we find the right solution among all possible ones? (A few solutions may match the measurements to within a given error.) We have shown that this last decision problem belongs to the class of difficult combinatorial decision problems--the so-called NP complete problems. The last statement also proves that the existence problem (i.e., Does a solution exist?) is NP complete.

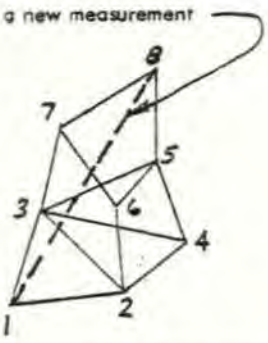


Figure 3. A Triangulated Positioning System with an additional Measurement

Not all positioning systems may be solved with the aid of triangles. In fact, for a system possessing a sufficiently large number of nodes, it is always possible that the position of the nodes can be located only by solving for the location of all points simultaneously. Such unfortunate systems require an enormous amount of computation. For a simple example, consider the hexagon of Figure 4. It is impossible to solve the location of its nodes using an incremental algorithm; all must be solved at once.

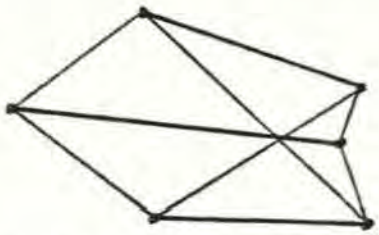


Figure 4. A Positioning System which may only be solved simultaneously

Fortunately enough, such primitive systems (whose parts cannot be positioned unless the whole system is) seem to be rare. Many positioning systems could be solved using an incremental process (which simplifies the solution algorithm and increases its speed and accuracy). However, an algorithm that would construct the location of a given point system by constructing subsystems first should be able to identify constructible parts. In particular, such an algorithm should be able to decide whether or not a given positioning system contains a subsystem that may be solved independently. Namely, any incremental construction algorithm must decide whether or not a given positioning system is primitive (i.e., constructible but having no constructible subsystems). The last problem seems to be a difficult combinatorial problem which we suspect to be NP complete (though we do not yet know how to prove this).

If the last conjecture is true, then the problem of constructing solutions to the positioning problem, using an incremental algorithm, is NP complete. This unfortunate result does not imply that "brute force" (i.e., iterative algorithms) should be preferred. It is reasonable to believe that most actual positioning problems may be better solved by means of an intelligent incremental algorithm. The precise meaning of "most" is yet to be defined.

Numerous other challenging and interesting related problems exist. While we will not make a comprehensive presentation, we will examine some of the problems formally, expose the difficulties, and present some partial solutions we have developed. This report is an extended summary of our present state of knowledge. A more detailed report is now being prepared.

1. THE PROBLEMS

The positioning problem can be described as follows:

1. $P \triangleq \{P_1, P_2, \ldots P_N\}$, a set of points in the plane.
2. A set of distance measurements between some pairs of points. Each measurement datum consists of the identity of the pair P_i and P_j, the measured distance d_{ij} and an estimate of the measurement error ϵ_{ij}.
3. Position coordinates for at least three points, say P_1, P_2, P_3, to be called the base triangle.

We shall call the set of points P_i together with the distance measurements $\{d_{ij}\}$ and the base triangle, a point system.

1.2 PROBLEMS

A feasible position of the point system is a set of coordinates that satisfies the distance constraints and assigns to the base triangle its actual coordinates. The set of all feasible positions will be called the solution set of the positioning problem. The positioning problem consists of characterizing the solution set namely,

1. Is the solution set empty? (existance)
2. Does the solution set contain a continuum of solutions, or is it a discrete set? Is it a finite set? (generalized uniqueness)

3. If the solution set is finite, what are all the feasible solutions? (position construction)

4. What is the error in the position due to propagation of measurement errors? (error analysis)

5. What is the sensitivity of the solution set to measurement errors? (sensitivity analysis)

In what follows we examine some partial answers to some of the difficult problems posed above.

Note that in the sequel we restrict ourselves to the problem of positioning points relative to each other, i.e., with respect to any coordinate system of our choice. This leaves us three degrees of freedom (two for translation and one for rotation) and the orientation for our choice of the coordinate system. With the aid of the third item on the input list (section 1.1) it is possible to position the point system absolutely once it has been positioned relative to an arbitrary coordinate system.

2. GEOMETRIC RESULTS

We associate with the point system a graph whose vertices represent points and whose edges represent distance measurements. We call this graph the measurements graph and say that a given property of a point system is combinatorial when it can be expressed in terms of the measurements graph only. A structure is a graph together with a mapping of edges into positive real numbers which we call lengths. A positioning system may be considered as a pin-jointed bar structure, i.e., a truss. Problems of uniqueness of (i.e., structure of the solution set) for the positioning problem translate into problems of rigidity of the respective truss. Problems of a solution's sensitivity to errors translate into problems of infinitesimal rigidity of the respective truss (i.e., admissibility of infinitesimal flexing of the truss). Problems of constructing a solution to the positioning problem correspond to construction of the truss. Therefore we shall use methods and terminology that pertain to both structures and trusses.

2.1 RIGIDITY

The results in this area fall into three classes:

1. A number of different characterizations of infinitesimal rigidity (error sensitivity).

2. An algorithm to determine whether or not a given solution of the positioning problem is sensitive to measurement errors.

3. A combinatorial characterization of plane rigidity in terms of a property of the underlying measurements graph.

While the problem of structural rigidity has attracted mathematicians, engineers, and architects for several centuries,[1] the solution to most fundamental questions of rigidity are far from known. Recently interest in this age-old problem has been renewed [WHITELY 77,78], and some significant contributions produced.

2.1.1 Characterization of rigidity

The first problem, i.e., that of characterizing rigidity, has a few solutions, all of which employ local infinitesimal characterizations. Loosely speaking, a structure is rigid if it does not admit relative motions of its parts, that is, if the only motions which it admits are trivial (i.e., translations and rotations). Therefore the study of rigidity is a study of possible motions. A rigid structure corresponds to a positioning problem with a discrete solution set. An infinitesimally rigid structure corresponds to an error-insensitive solution.

There are two approaches to motions of structures: It is possible to consider the velocity vectors of the nodes or the relative angular motion of edges attached to a common node. Accordingly, it is possible to develop two notions of infinitesimal rigidity. Another possible approach is to consider the stresses in the structure resulting from applying external forces. Rigidity may be defined as the ability of the structure to resolve forces. It is possible to show that both the above approaches are equivalent [GLUCK 75, WHITELY 77, 78].

Open problems:

1. Characterization of rigid structures which are not infinitesimally rigid.

   An infinitesimally rigid structure is rigid, but not vice versa. It is possible for a structure to be rigid (i.e., admit no finite relative motions of its parts) yet to admit infinitesimal perturbations (i.e., be sensitive to errors). A typical example of an error-rigid rigid structure is the degenerate triangle in figure 1.

   The difficulty in solving this problem is that we do not possess extensive tools for global analysis, while local analysis is well developed.

2. Characterization of rigidity with respect to discontinuous motions such as reflections.

   Consider the pair of pasted triangles in figure 5(a) below. The two triangles may be positioned with respect to each other in two distinct ways. The two resulting structures are rigid (do not admit non trivial motions) but admit relative reflection of parts. Other structures

[1] A partial list of researchers interested in the problem includes Pascal, Euler, Cauchy, Maxwell, Cayley, Alexandrov, and others.

admit a more complex form of discrete movement of their internal parts., e.g., Figure 5(b).

The problem of characterizing rigidity with respect to discrete motions is difficult, for we not only have to address a problem of global analysis, but also face difficult problems of combinatorial topology.

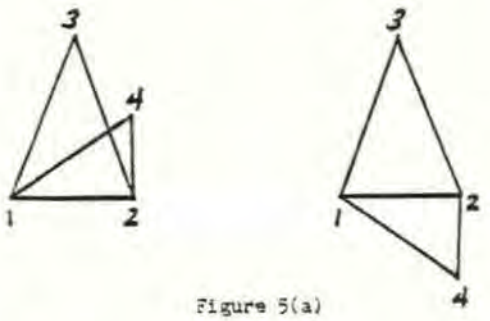


Figure 5(a)

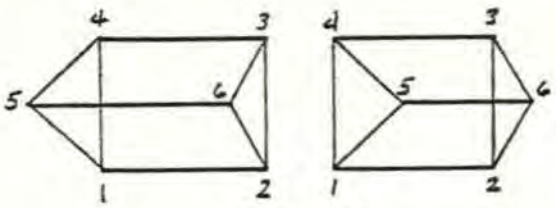


Figure 5(b)

Figures 5(a) and (b). Two cases of two rigid structures solving the same positioning problem.

2.1.2 Error sensitivity

The different definitions of infinitesimal rigidity induce different algorithms to determine whether or not a given structure is infinitesimally rigid. However, most of those algorithms have been produced by and for mathematicians unconcerned with computational efficiency. We have developed a novel algorithm to test whether a given feasible solution of the positioning problem is infinitesimally rigid, i.e., insensitive to errors.

The idea behind the rigidity testing algorithm is simple: try to solve for an admissible assignment of infinitesimal velocities that flexes the structure. It is necessary to examine only the effects of a velocity assignments over a set of basic circuits of the underlying measurement graph in order to reduce the problem to the solution of a linear system of equations.

Open problems:

1. Establish measures of error sensitivity and algorithms to compute sensitivity.

The characterization of error sensitivity in terms of infinitesimal rigidity is too crude. We would like to have an estimate of how much error sensitivity a given structure possesses.

2. Develop sensitivity measures in terms of the distance measurements data, not the particular solution they yield.

In addition to the difficulty involved in developing apriori sensitivity measures, we face the difficulty that the same measurement data may have a number of solutions, each possessing different sensitivity. Apriori it is even possible for a given system of measurement data to possess some rigid and some nonrigid solutions. Figure 6 depicts two solutions of the same positioning problem, one rigid and the other infinitesimally flexible.

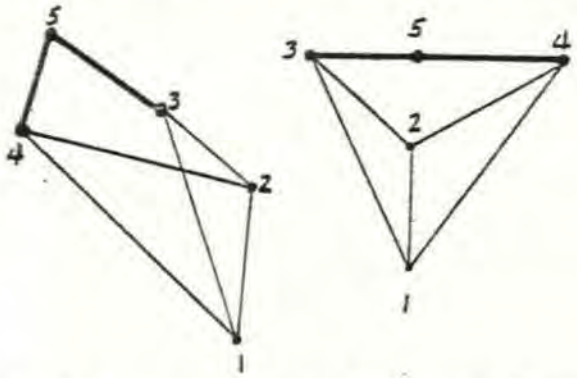


Figure 6. A positioning problem possessing a rigid and an infinitesimally-flexible solution

We shall now describe the third class of results in more detail.

2.2 FROM GEOMETRY TO COMBINATORICS

2.2.1 Stiff graphs

The most powerful results of the study of rigidity appear to be combinatorial characterizations of rigid structures. The ideas behind the passage from geometry to combinatorics are founded on some simple intuitive experiences,[2] the major idea of which is that some combinations of bars (edges) and hinges (nodes) are rigid for almost any choice of plane imbedding. For instance, the full graph on three nodes is rigid for all plane imbeddings; it is not infinitesimally rigid when the three points are colinear. Similarly, structures whose

[2] Intuition should not be pursued blindly, however, when it comes to problems of rigidity, for--as we shall see--many intuitive expectations turn out, surprisingly, to be false.

underlying graph is a full graph are rigid for almost any plane imbedding. On the other hand, some graphs will produce flexible structures for almost any plane imbedding (e.g., a circuit on four nodes is almost always flexible, except when one of the edges is assigned a zero length or when all four points are colinear). The problem is: Is it possible to characterize graphs almost all of whose embeddings form rigid structures?. This question has been the major problem of the theory of rigidity.

We shall define a critical combinatorial property of graphs which we call (plane) stiffness. First we associate with a graph $G \triangleq \langle V,E\rangle$ a number $f(G) \triangleq 2|V|-|E|-3$, measuring the overall excess of unknowns over constraints. (Each vertex contributes two unknown coordinates, and each edge constrains the two vertices incident unto it through a single equation for the distance. Note that we discount three degrees of freedom to account for possible external motion, i.e., translation and rotation.) The quantity $f(G)$ measures the overall freedom of internal movement of the graph.

The quantity $f(G)$ may be used to express the property of stiffness, i.e., of having a sufficient number of constraints to prevent relative motions of different parts of a graph. Loosely speaking, a graph is stiff if it is possible to remove some redundant edges so that the remaining graph has 0 degrees of internal freedom and non of its subgraphs has an excess of constraints (i.e., a negative internal freedom). Formally, a graph $G=\langle V,E\rangle$ is stiff if it has a spanning subgraph $G'=\langle V,E'\rangle$ (i.e., G' is generated by removing excessive constraints from G) such that

1. $f(G')=0$ (i.e., G' has 0 degrees of internal freedom)
2. if G" is any subgraph of G' then $f(G") \geq 0$. (i.e., G' does not possess internally over-constrained subgraphs).

2.2.2 The rigidity theorem

The most important result of the geometric theory of positioning is

THEOREM (Plane Rigidity) [3]:

> A rigid structure must have a stiff measurements graph. Conversely, almost all plane imbeddings of a stiff graph are rigid.

(Here "almost all" is used in the topological sense, i.e., the set of rigid plane imbeddings of a stiff graph is open and dense in the space of imbeddings, which further implies that for any Borel probability measure on the space of all imbeddings, continuous with respect to Lebsegue measure, the set of non rigid imbeddings of a stiff graph is of measure 0.)

[3] [LAMAN 71, GLUCK 75, WHITELY 78, ROTH-ASSIMOW 78]

The theorem above is a significant tool for handling positioning problems, namely, it makes it possible to infer properties of structure and derive answers to the problems of positioning by examining the measurement graph only. The results that we derive will be true for almost all assignments of distances to edges of the measurement graph, which greatly simplifies the study of the positioning problem.

To further appreciate the power of the result let us note in passing that the theorem does not generalize even to three dimensional structures. That is, it is possible to have a sufficient number of well distributed constraints and yet have a structure with a continuum of solutions no matter what lengths are assigned to the edges. Figure 7 below depicts a typical system. The combinatorial characterization of rigid structures in spaces of dimensions greater than two is an open problem.

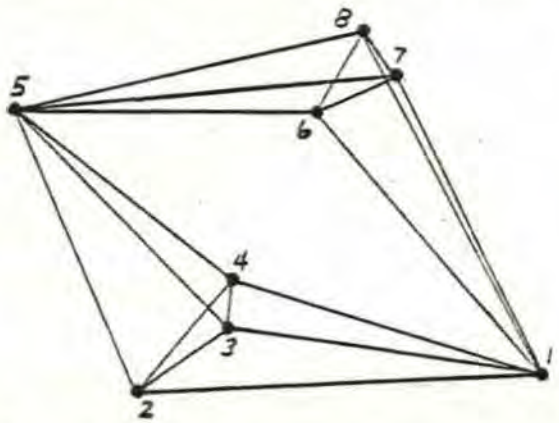


Figure 7. A counter example to a 3-dimensional rigidity theorem

2.3 LIMITATIONS OF THE RIGIDITY THEOREM

The rigidity theorem guarantees that a structure based on a stiff graph will almost always be rigid. However, it is possible to assign lengths to edges of a stiff graph such that the resulting structure admits infinitesimal flexing and is, therefore, very sensitive to errors. In fact, the structure becomes an error-increasing mechanism. One such structure is the degenerate triangle of Figure 1, another is Pascal's hexagon depicted in Figure 8. This hexagon is stiff and thus rigid for almost all plane imbeddings. However, the hexagon is infinitesimally flexible whenever (and only if) its nodes lie on a conic section. This bizarre result, due to [CROFTON 1878], has been rediscovered independently by many researchers (including ourselves). The proof follows from a simple application of a celebrated theorem of Pascal.

An even worse case is that of a stiff graph admitting a continuous motion (i.e., an imbedding of the stiff graph that is not even globally rigid). Such a structure is depicted in Figure 9.

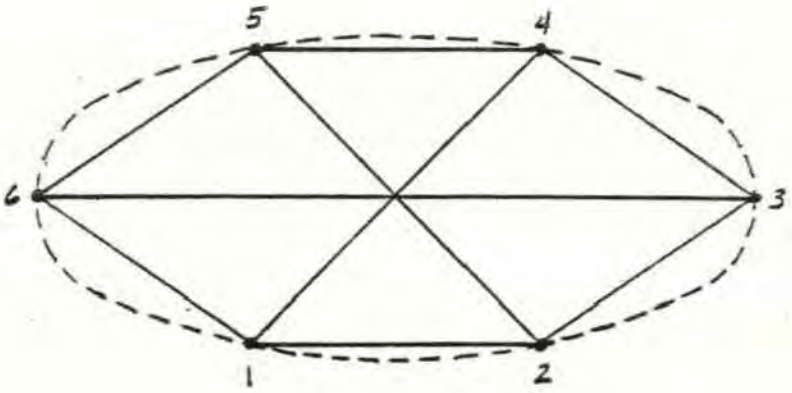


Figure 8. Pascal's Hexagon

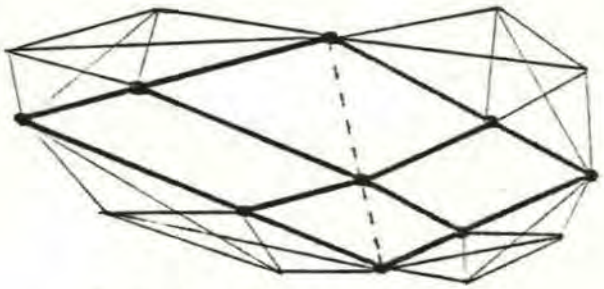


Figure 9. A flexible structure whose underlying graph is rigid.

Although the underlying graph is stiff (even overconstrained), the solution set of the respective positioning problem contains a continuum of solutions.

To produce such an example we started with a nonrigid structure (i.e., a mechanism), then meticulously added bars that did not constrain the motion of the original mechanism. Specifically, we started with a four-bar mechanism, selected a point on any diagonal and connected this point to points on the original mechanism. By careful selection of the connections to produce two parallelograms, the motion of the original mechanism is not perturbed by the additional "constraints." Additional bars are used only to hold each of the original four bars together.

The rigidity theorem guarantees that such unpleasant positioning systems are extremely rare. Yet we should bear in mind that they may exist (highly symmetric structures are very likely exceptions).

3. COMBINATORICS OF POSITIONING

Having seen the significance and limitations of stiffness, we will now study stiff graphs in order to develop methods for recognizing stiffness and for tearing structures into stiff subparts, later cementing those parts together. (Tearing is the natural instrument for a divide-and-conquer approach to the construction problem.)

3.1 CHARACTERIZATION OF STIFF GRAPHS

The question that we would like to address in this section is: What makes a graph stiff? We wish to derive necessary and sufficient conditions for stiffness in terms of well known graph properties. The most natural of classical graph properties relating to stiffness is connectivity.

3.1.1 Some simple characterizations

We have been able to derive a list of useful properties of stiff graphs:

1. A stiff graph is a block (i.e., has no cut vertex).
2. A cut-set of a stiff graph, separating it into two nontrivial components, must contain at least 3 edges.
3. A 3-cut-set of a stiff graph, separating it into two nontrivial components, must also separate it into two stiff graphs.
4. If $G=\langle V,E\rangle$ is a stiff graph and $V' \subset V$ a vertex cut-set separating G into components $G_i = \langle V_i, E_i\rangle$ and if the subgraph of G

spanned by V' is stiff, then so are the subgraphs spanned by V' ∪ Vi.

5. If v is a vertex of degree two in a graph G, then G is stiff iff the subgraph of G formed by removing v is stiff too.

The above results are a sample from a larger class of results, all of which serve to establish tools for a "divide and conquer" approach to the stiffness problem. For instance, we would like to determine whether a given graph can be torn into stiff subparts which may later be used to synthesize the original graph. Figure 10 depicts a typical configuration, corresponding to result (3) in the above list. Figure 11 depicts another possible configuration which admits tearing, this time a particular instance of (4).

3.1.2 Stiffness and connectivity

Loosely speaking, stiffness is a property of graphs which has to do with the density of edges, i.e., it is a measure of how well different nodes are attached to each other. It is only natural to expect that such a property should bear relation to classical measures of edge-density.

There are three major classical measures of edge-density: node degrees, minimal edge cut-set, minimal vertex cut-set. Here we explore the relations among these three properties and stiffness.

1. <u>There exist graphs whose nodes possess arbitrarily large degrees but which are not stiff.</u>

To prove this result we describe a simple method to construct counter examples. Start with a flexible graph, say a four-bar mechanism. Add nodes and edges to increase the degrees, preserving the flexibility. This process is demonstrated in Figure 12.

The above process serves to show that:

2. <u>There exist graphs with an arbitrarily large minimal edge-cut-set but which are not stiff.</u>

Thus, two of the classical measures of connectivity are not related to stiffness. Let us now consider the strongest measure of connectivity, vertex-connectivity.

We have seen in the previous section that a stiff graph is at least 2-connected. It is easy to construct 2 and 3-connected graphs which are not stiff. But with k-connected graphs, $k \geq 4$, the problem is no longer simple. Indeed any 4-connected graph must have a minimal degree which is at least 4. Thus the total number of edges m is at least twice the number of nodes. Therefore the overall number of internal degrees of freedom $f(G)=2n-m-3$ is not greater than -3. Not only is a 4-connected graph over-constrained, but the

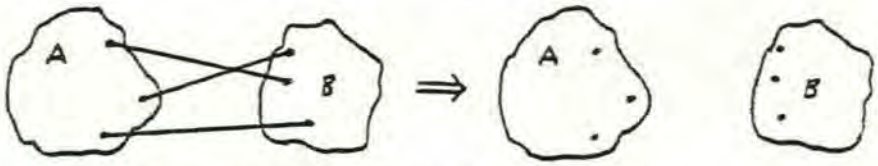


Figure 10. Tearing along a 3-cut-set

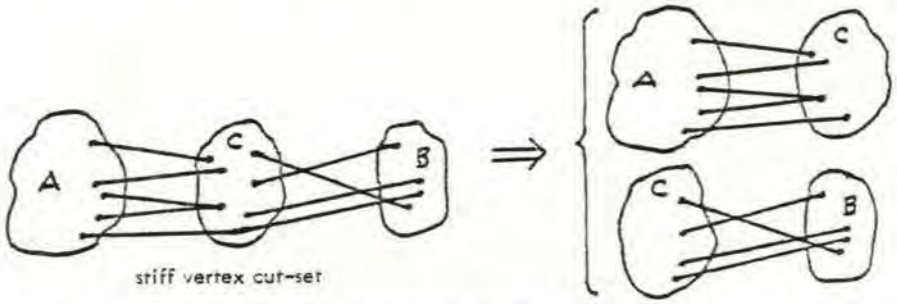


Figure 11. Tearing along a stiff vertex cut-set

connectivity implies that the edges must be well distributed. Intuitively one would expect that a 4-connected graph is stiff.

Not so. It is possible to construct 4-connected and even 5-connected graphs which are flexible. Two such examples are depicted in Figures 13 and 14.

The process we applied to derive these two surprising graphs cannot be applied to produce 6- (or more) connected graphs which are flexible. We do not know at all whether such graphs even exist. The problem is open:

3. Is there a number k such that any k-connected graph is stiff?

At this point, however, the problem is mainly of an academic interest, for a condition which requires such a high connectivity seems to be of no practical significance.

To summarize, we have seen that the relation between stiffness and measures of connectivity (if there is any) are not simple, contrary to the apriori intuition which leads us to explore these relations.

3.2 CONSTRUCTION PROBLEMS

A position-locating algorithm is essentially a process that starts with some set of points whose relative positions are known and gradually attaches new sets of points whose relative positions are computed with respect to the original nucleus. Such a process may be viewed as a (possibly parallel) solidification of parts of the measurement graph into bodies.

To be able to describe incremental construction processes we need to introduce a suitable formalism. In the following we shall describe such a formalism, then apply it to develop and implement construction algorithms.

3.2.1 Stiff hypergraphs

A Hypergraph $H \triangleq \langle V,E \rangle$ consists of a set of vertices V and a set of edges E. An edge is a subset of vertices (not necessarily just two, as in graphs). We shall use this generalized notion of an edge to describe a set of vertices whose relative positions are known. An edge is said to be incident upon a given vertex if it contains the vertex. A vertex is said to be incident upon a

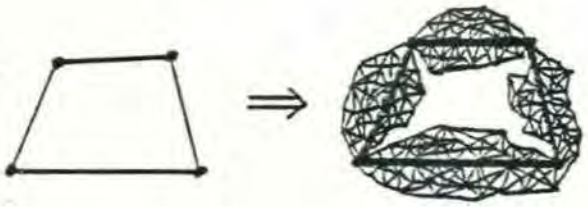


Figure 12. Constructing flexible graphs with arbitrary node degrees

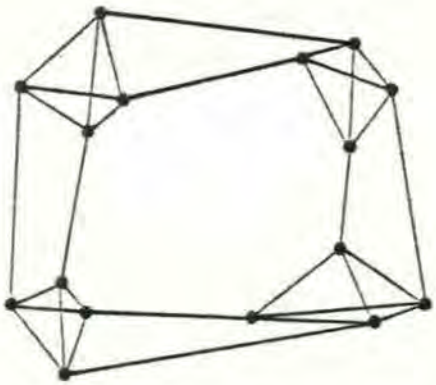


Figure 13. A 4-connected flexible graph

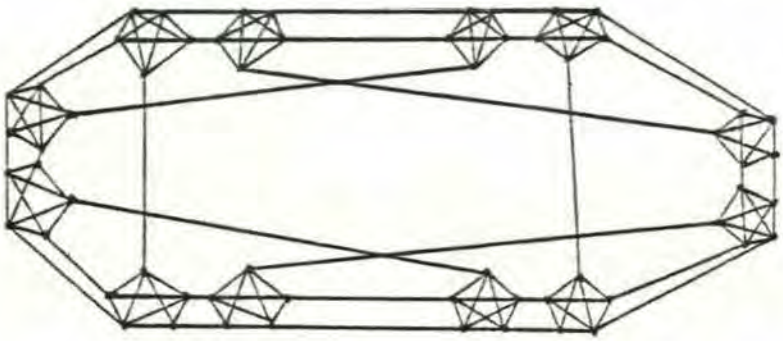


Figure 14. A 5-connected flexible graph

given edge if it is contained in this edge. We shall consider only hypergraphs for which the intersection of any two edges contains at most one point.

To develop a visual intuition of the processes to be discussed, one should consider hypergraphs as a generalization of structures. Rather than considering pin-jointed bars, we consider pin-jointed metal sheets of an arbitrary shape and number of pins (each metal sheet corresponding to an edge of the hypergraph and each vertex corresponding to a joint). To draw further on the analogy, we shall use the term link as an alternative to an edge; the term joint will be employed as an alternative to vertex.

Let d_i denote the degree of the i-th vertex (i.e., the number of edges incident upon it). Let d_i^* denote the degree of the i-th edge (i.e., the number of vertices incident upon it). The degree of the hypergraph H is defined to be the number $d(H) = \sum_{i=1}^{n} d_i = \sum_{i=1}^{m} d_i^*$, where m(n) is the total number of edges (vertices).

The number $f(H) \triangleq 2n+3m-2d(H)$ is called the internal freedom of H. It is easy to verify that for a graph H the definition of f(H) above degenerates to the number of internal degrees of freedom defined in previous sections.

The notion of stiffness can be generalized to hypergraphs as follows. A hypergraph H is said to be stiff iff it contains a spanning hypergraph H' such that

1. $f(H') = 0$
2. for any subhypergraph H" of H' $f(H") \geq 0$

Let us introduce a partial order over hypergraphs, which we call welding. A hypergraph H' is said to be a welding of a hypergraph if each edge of H' is a union of edges in H which span a stiff subhypergraph of H.

It is possible to show that stiffness is preserved under welding. Moreover, each hypergraph possesses a unique welding which is maximal (cannot be welded any more). If M(H) designates the maximal welding of the hypergraph H, then f(M(H)) defines the degrees of freedom of H. It can be shown that $f(M(H)) \geq 3$ with equality iff H is stiff, in which case M(H) has a single edge covering all the nodes.

3.2.2 Incremental construction algorithms

An incremental construction algorithm is a process that starts with a given positioning problem and develops a solution by gradually increasing the sets of points whose relative locations are known. At each stage, the state of the computation may be described as a hypergraph whose edges consist of sets of points whose relative positions are already known. Such a hypergraph is necessarily a welding of hypergraphs in the previous stages. In short, an incremental construction algorithm is a process that traces a chain in the partial order of welding.

We define a welder to be an operator that takes a hypergraph and produces a welding of it. An incremental construction algorithm is thus a process of successive applications of welders.

We have developed software to represent and manipulate structures and hypergraphs. Two types of welders have been implemented and some simple construction algorithms tried. We possess the tools which are necessary to develop position-locating algorithms of increasing sophistication.

REFERENCES

1. Asimow, L. and B. Roth. 1977. "The rigidity of graphs." I, II, Preprint, University of Wyoming, Laramie.
2. Bolker, E. and H. Crapo. 1978. "How to brace a one story building." Environment and Planning Bulletin.
3. Crofton, M. 1878. "On self-strained frameworks of six joints." Proceedings of the Mathematical Society of London, 10, 13-16.
4. Gluck, H. 1975. "Almost all simply connected closed surfaces are rigid." In Springer-Verlag Notes, 438, Geometric Topology, Heidelberg, 225-239.
5. Laman, G. 1970. "On graphs and rigidity of plane skeletal structures." Journal of Engineering Mathematics, 4 (4), pp. 331-340.
6. Rodenberg, I. 1975. "Structural rigidity in the plane." Preprint CMR-510, University de Montreal.
7. Whiteley, W. 1976. "Infinitesimally rigid polyhedra." Preprint, Champlain Regional College, St. Lambert, Quebec.
8. Whiteley, W. 1977 and 1979. "Introduction to structural geometry." I, II, III, IV, Preprint, Groupe de Researche, Topologie Structurale, University de Montreal, Montreal, Quebec.
9. Whiteley, W. 1978. Private Communication.

THE DEVELOPMENT OF A MULTI-SENSOR SYSTEM FOR ASSESSING A THREAT ORDER OF BATTLE*

Thomas D. Garvey and Martin A. Fischler
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
December 1,1978

I INTRODUCTION AND OVERVIEW

A. Problem Background

The key to survival in a tactical environment is the ability to understand, assess, and anticipate the evolving threat situation. This perception of the threat is a necessary prerequisite for formulating an appropriate response, such as avoidance, countermeasures, or suppression.

While background information to prepare the tactical commander for this task is provided from intelligence estimates and reconnaisance data, real- or near-real-time information is derived during the actual engagement from electronic support measures (ESM) systems. In the past, an ESM system (such as a radar warning receiver) could identify threat systems from measured signal parameters of weapon-related radars. Present and anticipated threat systems, however, have evolved to the point where current ESM systems are no longer adequate.

In the air-defense environment, several factors can be identified that limit the ability of current ESM systems to cope with the threat environment. These factors, which we will mention briefly, provide driving forces to the development of an effective ESM system. First, air-defense (AD) systems are densely deployed in tactical areas, and can, by sheer numbers, cause heavy operator workloads. Second, since there may be a very short delay between the time a threat system emits a signal, and a weapon impacts on the target, fast response time is required. Third, many modern AD systems are able to use a variety of sensors for target acquisition, tracking, and weapon guidance. This means that a system to detect these threats must be able to acquire a variety of distinct types of sensor data itself. Even worse, some systems will never emit a signal, and will need to be actively searched for. A final factor of concern is that most AD systems today are highly mobile, and the value of information regarding their location decays rapidly with time.

* This work is being supported by the Advanced Research Projects Agency of the Department of Defense, and monitored by the Air Force Avionics Laboratory under Contract F33615-77-C-1250.

B. Alternative Approaches

There are three main options to consider for a satisfactory solution to the problem of providing near-real-time threat deployment information.

The first possibility is a system that includes a human operator to control and monitor data from the sensor suite, and draw appropriate conclusions. The primary advantage of this approach is that the operator can interpret sensor data in the context of a wide variety of knowledge and experience, which allows him to anticipate certain threats, and ignore those areas which are unlikely to support specific AD systems. The drawback is that operator workload quickly becomes unmanageable.

A second approach is to make a semiautomatic system by tailoring special-purpose combinations of sensors into "super-sensors." These specialized, hard-wired combinations (of which there are only a few examples) provide information about a narrow range of situations, but lack both the scope and the flexibility to be of more general use.

The third approach is to build an automatic system that can operate a variety of sensors, integrate and interpret their data, and have the flexibility to adapt to changing situations. There are (at least) two avenues that could be pursued here, a classical statistical approach, and an approach combining elements of probability theory with symbolic inference mechanisms.

A purely classical approach suffers from number of drawbacks. The rapidly changing characteristics of the threat situation makes it unlikely that an adequate probility model would be available. If an acceptable model were available, the number of states to be included would be overwhelming.

C. An Autonomous Multi-Sensor ESM System

The problem of interpreting sensor information to determine the threat situation is viewed as a perception problem. Our approach to this problem is to interpret sensor data in the context of a prior model. The approach combines probabilistic techniques with inference mechanisms in an attempt to realize the advantages of the three approaches mentioned above.

The problem separates naturally into three parts: inferring likely threats from available information, planning and executing a sensor utilization strategy, and integration of the sensor data into the situation model. A key aspect of the approach is the idea of using prior information to anticipate threats likely to be encountered. By anticipating likely threats, the system achieves two important advantages. First, the domain of discourse is limited. That is, a large number of possible threat systems can be reduced to a smaller number of plausible threat systems that actually ned to be considered when interpreting sensor data. Second, by determining which threats are likely,

the system can use its sensors to best advantage in acquiring relevant data.

The primary inference mechanism is derived from the Inference Net (INET) system developed at SRI[1] The INET is a production rule system that incorporates a modified Bayesian approach for updating the likelihoods of hypotheses.

The ESM system is currently undergoing development. Several major components have been completed, while others are expected to be finished shortly. The next section will discuss the system in greater detail.

II SYSTEM DESCRIPTION

A. Overview

There are several factors which make this environment unique with respect to others which have been addressed using either expert system approaches or perception techniques. First, it involves an intelligent, hostile opponent capable of deception. Second, the large number of decisions that must be made in a relatively short length of time means, a program operating in this environment must be autonomous. That is, it must require a minimum of human interaction. Next, the system must be able to deal with data from a variety of imperfect, "incommensurate" sensors, integrating their outputs in order to derive the most plausible threat assessments. Finally, the environment can change rapidly, and the program must be flexible enough that it can be easily modified to deal with new threats and situations not explicitly anticipated when it was written.

In this system, we are attempting to compile a threat Order of Battle (OB) that contains the identity of all known threats in an area and their locations. We first determine which threats are plausible at a specific location, based on terrain factors which determine accessability and the desireability of the location for situating an AD system.

In addition, we use an existing OB which records locations of known threats. Rules describing typical deployments of AD systems can be used to infer the presence (or absence) of one threat given knowledge of another. For example, a rule describing the fact that a particular antiaircraft artillery usually provides low altitude coverage for a specific missile system allows the program to infer the presence of one, given knowledge of the other.

A list of anticipated threats derived from the inference module is passed to a sensor strategy planner (which is not yet implemented, but will be based on work described in [2] which prioritizes them and creates a plan for operating the sensors so as to optimize the utility of the overall sensor suite. The sensor data that results will typically provide ambiguous identifications, which must then be resolved.

Disambiguation involves three steps. The first uses information from sensors with overlapping coverage. The second uses geometric and parametric constraints specified by typical deployment information. The third uses information about plasible threat locations derived in the anticipation phase of the system. The resulting threat identifications are integrated back into the OB, and the process continued.

The remainder of this section will describe the main steps in greater detail.

B. Threat Anticipation

This phase uses a rule-based inference system called the Inference Net (INET) to draw conclusions from a variety of types of prior information, including graphic data derived from maps.

In the INET, knowledge is expressed as a set of inference or implication rules, similar to a logical implication of the sort,

$$E \Rightarrow H.$$

This is a logical inference rule which states that if statement E is known to be true, then H is also known true to by implication. In a rule-based system, however, it is typically the case that knowing E is true does not unequivocally establish H, but only allows the truthfulness of H to be estimated. This necessitates a probabilistic inference approach, where rules are represented as

$$E \overset{L}{\Rightarrow} H.$$

Here L represents the "strength" of the rule, that is, the degree to which knowing the evidence, E, allows the system to infer the hypothesis, H. L, then, can be used to "update" the likelihood of H given knowledge of E.

Since it is frequently the case that more than one rule may impinge on a given hypothesis, a computational procedure is required to allow the determination of the combined effects of the individual pieces of evidence. Typically, an assumption of conditional independence among the various pieces of evidence is made, and then relatively straightforward manipulation of Bayes' rule allows computation of the effect of the evidence.

H may in turn be the evidence for some other node. In this manner, it is possible to create an interconnected network of rules.

Several types of knowledge are encoded into rules for use by this system. These include: descriptions of composite threat systems (e.g., an SA-6 surface-to-air missile system consists of a STRAIGHT FLUSH radar and a number of transporter-erector-launcher vehicles (TELs)); deployment

information (e.g., a 23-mm antiaircraft gun usually provides low altitude support for the SA-6); terrain influences (e.g., SA-6 systems are usually found near roads -- where "near" is a function of terrain); and the desirability of a location (e.g., SA-6s prefer to occupy hilltops).

A version of the INET program was developed for drawing inferences at each point in a raster overlaid on a designated area[5] and operates in two passes. The first pass encodes the raster of points over the designated area with information including their distance from items of interest, such as roads, areas of clear terrain, or locations whereа threat has previously been observed. These distances are accessed during the second phase of the operation in order to determine the likelihood of evidence nodes, such as "Point x is near a road." By propagating such evidence through a series of implications, it is possible to compute the likelihood of a location being favorable to a specific threat.

The output of this program can be displayed to indicate these favorable locations. The results of interest (i.e., the likely threat locations) are then passed to the sensor strategy module.

C. Sensor Utilization Strategies

The role of this phase is to examine the list of anticipated threats, prioritize them (usually by probable lethality), compare them to models of the sensors expected performance, select the appropriate sensors for detecting the highest priority threats, and determine the sensor parameters that need to be preset. While the module for computing the sensor strategies has not yet been implemented, we can outline its operation. First we will describe the sensor modules used by the system.

It is appropriate to mention here that this effort assumes certain capabilities on the part of the sensors. In particular, we do not attempt to deal with raw data, but instead assume that each sensor has an associated preprocessor which enables it to analyze its data, detect targets of interest, and identify them (although not perfectly). The data returned from the sensors consists of these identifications with whatever location information is appropriate to the sensor.

Models of sensor performance are designed to represent the operation of the sensor with respect to a specified set of targets. The model includes probability arrays which specify the likelihood of the sensor detecting each system, and the possible identifications the sensor could assign with the likelihood of each. These tables are compiled assuming certain specified environmental conditions. For example, the probability tables for an electrooptical imaging sensor might have been compiled assuming a clear day, a certain range to the target, a certain focal length for the lens, and so on.

In order to adapt the models to environmental conditions, we provide a set of probability degradation factors. These factors are combined and used to modify the tables for existing conditions. In this manner, the program is able to estimate the efficacy of a given sensor for detecting and identifying a specific threat under nonstandard conditions.

Certain of these factors are related to controllable sensor parameters, such as lens focal length, or the bandwidth of a receiver channel. The strategy system will attempt to select appropriate sets of parameters, and balance the "cost" of using a given sensor with its utility, in order to create a cost effective sensor strategy.

Once sensors have been selected and their parameters set, data is collected and passed to the interpretation module.

D. Sensor Data Interpretation

Since sensors are not perfect, the identifications they render are likely to be ambiguous. An important job for the overall system is to disambiguate, as much as possible, the sensor returns.

The first step in disambiguation is to compare responses from sensors whose coverages overlap. By using the probability models that describe the sensors, it is possible to develop identifications based on the composite sensor information and, in general, reduce ambiguity.

The next step is to use information describing typical geometric deployments of threat systems and related systems, and their relative parameters to further narrow the field of possibilities. This phase uses geometric descriptions of threat systems (for example, an SA-6 system consists of a centrally located radar surrounded by four SA-6 TELs in a square array, 100 meters on a side) expressed as templates. Using a variation of a linear embedding algorithm[4] the pattern matching program attempts to find the best matches of known templates to the sensor data.

This has two effects. It allows the system to use typical geometric constraints to aid in the identification of threats. In addition, since it is often the case that not all elements of a template are detected, it allows the system to specify not only which elements are missing, but where they should be located.

The results of this level of interpretation are compared with the favorability data computed by the inference net, and confident identifications entered into the OB through the INET. This new information causes the system to update its likelihoods and the process is continued.

E. Status

This system is currently under development. Most elements have been implemented, with the exception of the strategy planner. Models for several typical sensors have been developed. A revised set of threat rules is being incorporated into the system at present. The pattern matcher and the sensor integration have been implemented. The next phase will see the integration of the existing pieces and the development of the planning module.

F. Extensions

There are several extensions and improvements that can be identified at this time. Two main suggestions concern the operation of the sensors and the probabilistic updating mechanism in the INET.

The current approach does not address the problem of analyzing raw sensor data at all. Rather, it assumes certain interpretive capabilities on the part of the sensor preprocessor. The Image Understanding community generally believes that the more relevant, high-level information available to a sensor-data processing system, the better its analysis is likely to be[3] The current approach consolidates most of the relevant information in the interpretation routines, where it is usd to criticize sensor performance in an a posteriori fasion, rather than allowing it to be passed to the sensor systems. In a working system, with actual sensors, it will probably be critical to make this information available at a low level.

Furthermore, some of the most relevant data may be that which is being collected simultaneously by other sensors. Instead of performing all data integration at a high level, as is currently the case, it appears more promising to create temporary, direct communication paths between individual sensors.

The second area where improvement could be made concerns the likelihood updating computations in the INET. There are two primary drawbacks to the current Bayesian approach. The first is that assumptions required in order to make the computations tractable frequently are not valid in our environment. In particular, the assumption of conditional independence among evidence nodes impinging on a hypothesis is usually not valid. In many cases of interest, situations are highly interrelated. For example, normal encoding of the rules linking situations often leads to loops. Use of the independance assumption in these cases provides incorrect results. Our approach has been to perform inferences based on tightly coupled situations outside of the INET. For example, the pattern matcher was implemented after several attempts to do the matching in the INET. Another approach would be to extend the INET formalism to allow weakening of some of the current INET assumptions.

Furthermore, the use of point probabilities (which pervades the whole system) does not allow adequate modelling of many situations. Instead, some idea of the distributions involved are required. In future versions of the system, we will implement a scheme for using intervals as a more accurate model of actual distributions.

III SUMMARY

Our approach to the problem of integrating multisensor data with prior knowledge has several key points which should be underlined.

The characterization of the task as a perceptual problem suggested many of the subproblems that had to be addressed, and pointed out the need for a common frame of reference within which sensor data could be integrated and interpreted. In this system, the reference frame is the Order of Battle. Data from sensors is interpreted in the context of the OB. In turn the OB provides information about likely threats.

Another key point is the use of anticipated threats to both narrow the domain of discourse, and to enable the system to plan for effective use of sensors. In particular, the use of "extra-signal" information such as terrain data and deployment rules to produce the list of anticipated threats seems to mimic the behavior of a trained and briefed crew member.

The incorporation of a planning element provides many capabilities. It provides a mechanism for effective use of modular sensor descriptions, which can be added, deleted, or modified as the system configuration is altered. Use of a planner also provides the flexibility needed to allow easy modification of threat data, since changing threat characteristics will result in different sensor strategies.

The modular sensor descriptions could provide the basis for extensions to the system. Sensors are, in effect, modeled as knowledge sources. It appears likely that more abstract sources of information could be modeled in the same manner, thereby allowing the system access to other types of information, such as intelligence estimates. The models will also allow the system to be run in a simulation mode, in order to determine which characteristics should be improved to make a sensor more effective. It could even be used to decide which tradeoffs would provide the greatest capabilities in a sensor, before completing its development.

A final point is that the up-to-date OB determined by the system should be easily transferrable from one platform to another. This could provide the capability for removing a tape cassette from a returning aircraft, and inserting it into one about to depart, thereby transferring the real-time OB to the outbound aircraft. This

would allow maximum us of the new information while it is still timely, without requiring the lengthy debriefing-analysis-briefing process.

In summary, the approach provides the potential for an extremely robust ESM system, capable of using a wide variety of information, in ways not possible in any existing program. In addition, the work suggests several possible extensions and spinoffs to related problem areas.

REFERENCES

1. Duda, R. O. et al, "Semantic Network Representations in Rule-Based Inference Systems," in Pattern-Directed Inference Systems, pp. 203-211, Waterman & Hayes-Roth, eds., Academic Press (1978)..

2. Garvey, T. D., "Perceptual Strategies For Purposive Vision," Artificial Intelligence Center Technical Note 117, SRI International, Menlo Park, California, (September 1976)

3. Kanade, T., "Region Segmentation: Signal vs. Semantics," The Proceedings of Fourth International Joint Conference on Pattern Recognition, November 7-10, (1978).

4. Fischler, M. A. and R. A. Elschlager, "The Representation and Matching of Pictorial Structures," IEEE Transactions on Computers, Volume C-22, Number 1, pp. 67-92 (January 1973)

5. Hart, P. E. et al, "A Computer-Based Consultant For Mineral Exploration," Second Semiannual Report, Artificial Intelligence Center, SRI International, Menlo Park, California 94025 (April 1978).

Language Design for Distributed Systems

Peter G. Hibbard

In this paper we review some aspects of language design and language implementation as they relate to networks. In particular, we examine how the incremental integration of software components must lead to a re-assessment of the role of the compiler and the linker, and consider what additional facilities might be desirable. We examine the current paradigms for expressing parallelism which have been developed for closely coupled multiprocessor systems, and explore how well these satisfy the additional constraints of networks, viz the lack of common shared memory and the relatively low communication bandwidth. We also consider briefly the run-time organizations which will be needed to support these facilities.

1. Compilation

We assume that any particular task which is being executed by a network will comprise several relatively independent processes executing on possibly different hosts. During the lifetime of the task individual sections may need to be rewritten or replaced, or additional sections may need to be added, and it may be necessary to perform these modifications while the rest of the sections continue to operate. Traditionally such modifications would be done by employing some form of separate compilation, followed by a linkage editing phase exerting only weak checking on the external specifications of the components. However, we believe that any interaction which takes place between sections of program should be subject to the usual checks provided by compilation - these checks should at least include type checking of messages passed between processes. We are currently exploring the possibility of checking processes' specifications of their patterns of use of resources against system-wide constraints.

2. Specifying parallelism

Considerable progress has been made in devising and assessing the techniques required to coordinate pseudo-parallel and parallel tasks. However, the diversity of the tasks which it is required to coordinate is so great that they cannot all be programmed with a single technique. In part this problem arises because we lack sufficient experience in programming parallel algorithms, and in part it arises because the current programming control constructs, onto which the parallel constructs are grafted, overspecify the sequentiality in some cases, and do not provide any means for expressing parallelism in other constructs. Fortunately, when dealing with networks the restricted environment causes schemes based on message passing to be attractive, and a number of proposals have recently been based upon these primitives (Hoare, Brinch Hansen, Hewitt). We examine one such technique (eventual values) in detail, since it provides a high level model for interprocess which closely relates to the manipulation of values in algebraic languages, and disguises the details of communication and remote computation in such a way as to allow specification of programs which are reasonably independent of the network configuration.

3. Runtime organizations

There will be several different patterns of interaction between the processes which comprise a task. In the pattern of interaction which comprises a server process serving several client processes, it is possible for an interaction to take place by only one round of messages passing between them. In this case the server need keep no record of internal state, and it may release any resources aquired for the interaction. However, it is possible for the interaction to be a transaction which comprises several messages, and the server process must maintain some internal state, and retain access to resources until the transaction is complete. In the case where the completion does not occur, because either the server or the client has catastrophically terminated, it will be necessary to restore the resources to a consistent state, or to restore them back to the state which pertained at the start of the transaction. Techniques for handling these problems, at least in simple cases, have been developed for distributed data bases and file systems. Essentially they rely on simulating the effects of the operations requested by the client, without destructive change to the resource. This only takes place when the client signals that all interactions in that transaction are complete. During the period when the resource has not been updated, it is essential that the server responds to the client's requests for actions on the resource as though the update had occurred. In the case of file updating and reading, this is relatively simple, however investigations still need to be made for more complex cases to ensure that the simulated operations have a bounded time cost, irrespective of the prior operations performed on the resource. We report on experiences gained in developing bounded-cost pseudo-operations in a related programming environment.

DEPLOYMENT CONSIDERATIONS FOR LOW ALTITUDE DEFENSE*

By
John Fielding
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, Mass. 02173

SUMMARY

First order considerations of possible DSN system deployments for strategic defense against a cruise missile attack are presented. The deployments are discussed in terms of their geometrical coverage and their vulnerability to offense concentration.

Perimeter defense, ring defense, and random area defense are considered in the context of a defense of the Soviet Union.

Assumptions

It is assumed that a distributed sensor network is available for deployment which works in the sense that it provides the requisite tracking data to some undefined interceptor system in a way that does not limit overall system performance. It is assumed that the cruise missile attack is massive (thousands of missiles) and responsive to the defense deployment, but that the DSN elements are no more susceptible to direct attack than are other defense system components.

Perimeter Defense

The total perimeter of the Soviet Union is over 65,000 Km, however most of the Soviet economic structure is enclosed by a perimeter defined by the line connecting Leningrad with Novosibirsk, the line connecting Novosibirsk with Rostov-Na-Danv and the western border between Rostov-Na-Danv and Leningrad. This area, shaped like an ice cream cone, has a perimeter of approximately 9200 Km.[1] A defensive barrier against cruise missile attack need not be a closed perimeter. If the defense takes advantage of knowledge of cruise missile range and possible release areas, then a shorter and perhaps more effective barrier might be deployed as follows: from Vorkuta (app.67°N, 64°E) to Kalingrad to Odessa to Baku. This barrier is 5500 Km long, and may be said to protect almost all of European Russia.

The number of sensors required to implement this 5500 Km barrier depends on the spacing and on the depth of the barrier and will not be further discussed here. The number of weapon stations (SAM batteries or manned interceptor locations required) depends on their effective width. For SAM battery effective widths between 5.5 and 55 Km, 1000 to 100 batteries will be required. The depth of sensor net required to support these batteries will depend on the details of system operation, but will be approximately equal to the battery spacing. Fighter interceptors must be stationed so that they can reach the cruise missile before it leaves the coverage of the sensor net. Their spacing may be calculated from the cruise missile velocity, the fighter interceptor average velocity, and the DSN depth. For example, if the cruise missile velocity is equal to the fighter-interceptor average velocity and the depth is 50 Km, then the fighters may be stationed 100 Km apart.

Perimeter defenses suffer a serious deficiency, in that the offense may concentrate his attack into a few corridors and by means of local exhaustion, saturation, or defense suppression penetrate the barrier at modest cost. The attackers are then free to fan out from the penetration corridors and attack where they will. If the offense requires c corridors of width ω Km per unit length, then we may define an offense leverage, L, given by

$$L = (c\,\omega)^{-1}.$$

The reciprocal of L is the percentage of the perimeter resource that can be brought to bear on the concentrated attack. If the offense requires two corridors per 1000 Km of barrier and a corridor width of 20 Km, then L is 25 and only 4% of the defense weaponry is usable. To achieve this leverage, the offense incurs a modest range penalty and the losses associated with exhausting or destroying the defense resources in the corridors.

The views and conclusions contained in this document are those of the contractor and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the United States Government.

*This work was sponsored by the Advanced Research Projects Agency of the Department of Defense.

Ring Defenses

The defense can reduce the offense leverage by arranging the perimeter defense into rings which enclose the value to be defended. If, for example, rings of 100 Km perimeter (16 Km radius) are chosen then the offense leverage is limited to 100 divided by π. Thus a 20 Km corridor offers an offense leverage of 5.

The resources to be defended will not in general be found to be conveniently clustered for the purposes of defense. In the absence of detailed information on target clusters or value distributions, Zipf's law[2] may be used to approximate the distribution. Thus if the rings are ordered by decreasing enclosed value, then the value in the i^{th} ring will be proportional to $(i)^{-1}$. By arbitrarily assigning a value of 1000 to the first ring, the following table may be constructed:

i	Value in i^{th} Ring	Value in All Rings Up To i
1	1000	1000
10	100	2929
100	10	5187
1000	1	7485
10000	0.1	9788

It is apparent that the laws of social agglomeration are working against the defense designer if he tries to defend a sizable fraction of the nation's resources with small rings.

A defense based on uniformly defended rings will be seriously unbalanced. For example if the first 10 rings are defended by 10 batteries each (effective battery width of 10 Km), then the tenth ring is defended at 10 x the level on a per unit value basis as is the first ring. To produce a balanced ring defense, the defense must equip the rings in proportion to the value enclosed. If the enclosed value does not justify enough equipment to fully man each ring, then a balanced defense of that many rings at that proportion of defense cost to defended value is not possible.

Random Area Defense

The defense may deploy defense batteries at random throughout the region containing the targets (for the case of the Western USSR - about 3.4×10^6 sq. Km). If the offense does not know the location or ignores the presence of these randomly deployed batteries, then we can model the encounter process as random:

$$P = e^{-bdx}$$

where d is the density of sites (# per Km^2), b is the effective battery width (Km), x is the distance transversed by the cruise missile, and p is the probability of not encountering a defense battery in distance x. If we set $p = e^{-1} = 37\%$, we see that the e-fold distance is $(bd)^{-1}$. If b is 10 Km, and d is 10^{-4} then x is 1000. That is about 37% of the cruise missiles will survive a 1000 Km trip through an area containing 1 defense battery per 10,000 Km^2, each battery having a battery width of 10 Km.

If the locations are known to the offense, then rerouting can degrade the defense performance. If the presence of a random area defense is known, but the specific locations are not, the offense may still improve its position by routing the cruise missiles in a systematic way in an attempt to locally exhaust portions of the defense.

---

[1] "Area Handbook for the Soviet Union," DA PAM 550-95 contains a good introduction to Soviet geography.

[2] Zipf's law estimates the size of each member of an ordered set of social agglomerations as:

$$S_i = S_1 \, (i)^{-1}$$

where i is the rank order and S_i is the size of the i^{th} member. This is discussed further in "System Engineering" by Goode & Machol, McGraw Hill Book Co., 1957.