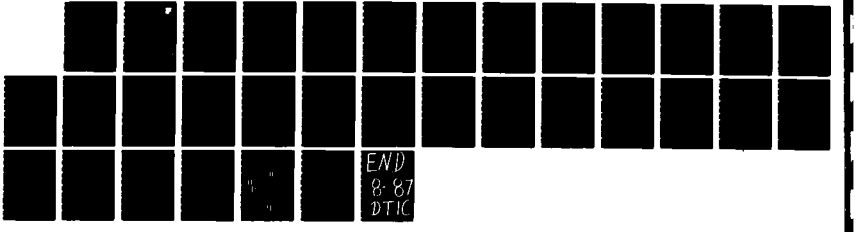
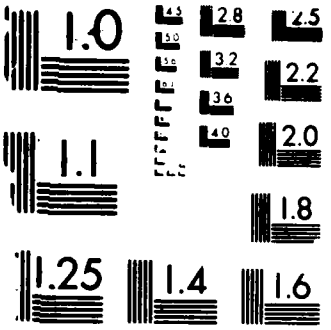
AD-A182 596 DISTRIBUTED SYSTEMS EVALUATION ENVIRONMENT: AN INTRODUCTORY REPORT(U) ROME AIR DEVELOPMENT CENTER GRIFFISS AFB NY A M NEWTON ET AL APR 87 RADC-TM-87-5 1/1

UNCLASSIFIED F/G 12/7 NL






MICROCOPY RESOLUTION TEST CHART

(12)




RADC-TM-87-5
In-House Report
April 1987



# DISTRIBUTED SYSTEMS EVALUATION ENVIRONMENT AN INTRODUCTORY REPORT

Anthony M. Newton and Richard H. Sweed

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED*



**ROME AIR DEVELOPMENT CENTER**
**Air Force Systems Command**
**Griffiss Air Force Base, NY 13441-5700**

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TM-87-5 has been reviewed and is approved for publication.

APPROVED: Ronald Raposo

RONALD S. RAPOSO
Chief, $C^2$ Systems Technology Division
Directorate of Command and Control

APPROVED: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command and Control

FOR THE COMMANDER: Richard W. Pouliot

RICHARD W. POULIOT
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTD) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE AD-A182596

Form Approved OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS<br>N/A |
| 2a. SECURITY CLASSIFICATION AUTHORITY<br>N/A | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>RADC-TM-87-5 | 5. MONITORING ORGANIZATION REPORT NUMBER(S)<br>N/A |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Rome Air Development Center | 6b. OFFICE SYMBOL (If applicable)<br>COTD |
| 6c. ADDRESS (City, State, and ZIP Code)<br>Griffiss AFB NY 13441-5700 | 7a. NAME OF MONITORING ORGANIZATION<br>N/A |
| | 7b. ADDRESS (City, State, and ZIP Code)<br>N/A |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>Rome Air Development Center | 8b. OFFICE SYMBOL (If applicable)<br>COTD |
| | 9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>N/A |
| 8c. ADDRESS (City, State, and ZIP Code)<br>Griffiss AFB NY 13441-5700 | 10. SOURCE OF FUNDING NUMBERS |

| PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO | WORK UNIT ACCESSION NO |
|---|---|---|---|
| 62702F | 5581 | 28 | 17 |

11. TITLE (Include Security Classification)
DISTRIBUTED SYSTEMS EVALUATION ENVIRONMENT AN INTRODUCTORY REPORT

12. PERSONAL AUTHOR(S)
Anthony M. Newton, Richard H. Sweed

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| In-House | FROM Jan 86 TO Dec 86 | April 1987 | 32 |

16. SUPPLEMENTARY NOTATION
N/A

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Computer Systems; Distributed Operating Systems; Distributed Systems; Networks; |
| 12 | 07 | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Distributed Systems Evaluation Environment (DISE) is an in-house research program of the Rome Air Development Center (RADC) started early in the calendar year of 1986. It seeks to provide a computing environment that allows Center researchers the capacity to design, test, and evaluate distributed computing systems. This paper discusses the progress during the initial implementation phase, as well as discussing some future directions that we intend to take.

| 20 DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21 ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS | UNCLASSIFIED |

| 22a NAME OF RESPONSIBLE INDIVIDUAL | 22b TELEPHONE (Include Area Code) | 22c OFFICE SYMBOL |
|---|---|---|
| Anthony M. Newton | (315) 330-2158 | RADC (COTD) |

DD Form 1473, JUN 86 *Previous editions are obsolete* SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

# Introduction

## 1. Background

The Command, Control, and Communication (C3) systems that the Air Force (AF) currently uses is a collection of independent and interdependent systems. While this sounds like a contradiction, the systems were usually procured for a particular function first, then integrated into the Command and Control (C2) structure later. Each system retains a measure of independence, but the information they process is critical to our C2 measures and countermeasures. Our interest and thrust within this C2 environment is to provide the capability for all of the individual systems to be integrated into a distributed system (a system of systems). This entails having a set of heterogeneous hosts that communicate and manage one another in a timely fashion with fault tolerance, reliability, survivability, et cetera. This system needs to be capable of adjustment as new demands, capabilities, and technologies are integrated through software or hardware additions. Moreover, all of this must be done at a reasonable cost.

The Distributed Systems Branch of the Rome Air Development Center (RADC) recently has established an in-house capability called the DIstributed Systems Evaluation Environment (DISE .. parse it any way you wish), wherein, we plan to continue development of distributed system technology areas and provide a demonstration capability for our funding sources.

The idea for a DISE came to fruition after realizing that the current level of research in Distributed Operating Systems (DOS) had reached a level of maturity.

It seeks to provide an environment where researchers can investigate and demonstrate issues of concern for a DOS, such as fault tolerance, reconfiguration, reconstitution, et cetera. Moreover, we also envision it as a place where true distributed applications can be written, experimented upon, and verified. By having one or more instances of distributed systems, we can allow the programmer to see the behavior of his application under conditions much more realistic than one or two host simulators.

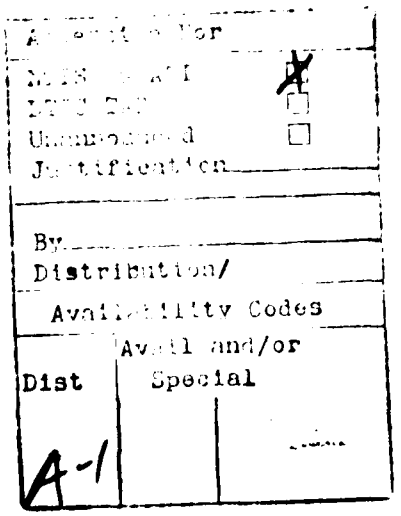
## 2. Goals

There are several goals to be pursued with the DISE. First, it will provide an environment for the design and development of distributed systems while also serving as a running demonstration of the systems integration technology within which distributed systems will play a part. Next, it will provide researchers with the tools necessary to evaluate current distributed system technology, that is integrated into a distributed application environment suitable for experimentation. Finally, it will provide AF C2 systems with the capability for "system evolution", thus, negating the need for total system replacement that currently exists.

The working definition of system evolvability requires that the services, capabilities, data, and elements of older systems should not be lost when those systems become antiquated or obsolete. Systems evolution should allow new systems to be integrated without sacrificing the old. Moreover, the system should have the ability to be "rejuvenated" by infusing new equipment and extracting the old equipment without any of the collective services ever being interrupted or disturbed.

Current distributed applications are written on one or two host nodes, with software routines or delays built into the code to simulate the effects of host distribution. The DISE will provide the ability for programmers to have realistic simulations that will give them a better understanding of the abilities and impositions of distributed systems.

The DISE must also allow us to probe and investigate the abilities and shortcomings of other distributed systems. DISE will provide a stable computing environment for the test and evaluation of distributed computing systems. In addition, it will provide an environment for the integration of new technologies, both related and unrelated to distributed computing, over the life of the research effort. In this way, we envision DISE as being a true instantiation of the evolvable system and, thus, constitute a "proof" by existence.




## Progress

### 1. Configuration Definition

Although the concept for the DISE does not limit the DISE to a single clustered set of machines, this was the best starting methodology. Thus, the initial aims where to: assemble a set of machines capable of supporting our research needs, physically connect those machines through some communication media (local-area network), select a Distributed Operating System (DOS) to bind the computer systems together, and link into a geographically dispersed (wide-area) network.

The initial configuration centered around a set of low cost workstations. These would be "generic" systems that could be introduced to and removed from the DISE with relative ease. This was done so that we could experiment and evaluate the usefulness of pluggable power for computational needs within the DOS environment. To accommodate the desire to experiment with Distributed Data Base Systems (DDBS), an IDM 500 relational database machine from Britton'-Lee is being integrated into the DISE. In order to provide a development environment and the capacity for large processing power, the VAX VMS 8650 was chosen. In order to gain insight into the resources and time needed to integrate a host with a type of architecture different than the VAX family line, the DECSYSTEM 20 mainframe is being incorporated.

Within RADC, a broadband Local Area Network (LAN) interconnects the various laboratory and office facilities. The current capability built into the broadband network was a simple terminal access channel that allowed remote

terminal access to connected machines throughout RADC. Other channels of the LAN were used as the "backbone" of our local network that would connect the DISE. ETHERNET was selected as the access protocol (an agreed set of rules for communications between computer systems) because of its longevity and industry acceptance. Details on the engineering effort to create the broadband Ethernet system, as well as the selection of a DOS, are discussed in later segments.

The DOS provides the interoperability to conduct experiments on network activity and computer resource management policies when those computers are separated by large physical distances. The ARPANET research network was chosen because of its accessibility and reliability. In addition, connection to the WIDEBAND Satellite Network provided an alternate, high bandwidth, long distance connection for future experimentation. Further discussions are detailed in a later segment.

## 2. Engineering a Broadband Ethernet

### 2.1. Understanding the Broadband Ethernet

RADC has a broadband LAN cable system that extends throughout its buildings. This LAN was selected as the basis upon which to build a broadband Ethernet implementation. Ethernet is a methodology that allows packetized data communication between computer systems utilizing a connection medium (a network) that supports serial transmission. All systems connected to the network must agree to use the Ethernet conventions when communicating with one another. Typically, Ethernet is divided into two devices, a controller and a transceiver. The controller historically contains the processing power necessary to implement the Ethernet protocol and is connected to the computer system. The transceiver is traditionally the physical connection with the network, where the transceiver

is responsible for transmitting data from the controller onto the network and receiving data from the network and sending it to the controller. Logical connections to the network (i.e. the appearance of sole network ownership by the user(s)) are usually handled through the protocol software running on a machine.

Regressing, there were two Ethernet implementations, referred to as Type I and Type II, that existed at the time we were putting together our configuration. Both these implementations were used by manufacturers in marketing their Ethernet controllers. Each standard had its own methodology for the use of collision signalling. Collisions basically occur when two or more Ethernet hosts attempt to transmit their Ethernet data packets simultaneously. The earlier standard used the packet collision signal if and only if there was an actual occurrence of a data packet colliding on the cable with another packet. This is a Type I implementation of Ethernet. The newer implementation provided the Type I collision signalling, but, in addition, used the collision signal after every packet that a controller would try to transmit. This was primarily done to provide Ethernet controllers with the ability to monitor their collision mechanisms so that malfunctions could be detected. This later version became known as collision reinforcement, and was the standard that was accepted by the Xerox, DEC, and Intel triumvirate. This is designated as being Type II.

In a baseband cable system (where one frequency is used to transmit information, as opposed to multiple frequencies that are available through broadband), the transceiver is a very simple machine whose basic process is to transmit data on the baseband cable, receive data from the baseband cable, and detect any collision that might have occurred while transmitting. Figure 1 shows the basic functionality for the transceiver. Data is sent to the transceiver with Manchester encoding. This data is handed to a transmitter and placed on the

cable. The receiver, sitting next to the transmitter on the cable, begins to receive the data that the transmitter emits. A DC level detector also sits on the cable to detect a possible collision by sensing the voltage level on the cable and seeing whether a threshold has been violated. If the voltage exceeds threshold, the transceiver will send a collision signal to the controller, and the controller will abort the packet. When the packet is aborted, the controller will continue to transmit the data packet for a specified amount of time. This time period must allow all transceivers connected into the network to: (a) realize that a collision has occurred on the network, and (b) signal the controller regarding the collision.

Broadband systems use frequency multiplexing to assign channels for each user needing cable access. Every user is expected to stay within that channel assignment of frequencies. Moreover, the modulation scheme for a broadband network is much different from a baseband system. In our case, the broadband modulation scheme uses Differential Binary Phase Shift keying with baseband Nyquist filtering to prevent signal aliasing. The code is then scrambled using the CCITT V.29 polynomial with the seed changing for each packet.

Figure 1 also shows a possible outline for a broadband transceiver. The Manchester encoded data that is received by the transceiver is scanned for the source address of the controller. When the source address is encountered, it is copied into the internal storage of the transceiver. Next, the data (still Manchester encoded) must be translated into the appropriate broadband transmission scheme. More synchronization bits are added for the receiver and the results get sent to a transmitter. The transmitter requires time to "turn on", so that impulses in the frequency domain can be avoided. The signal traverses the transmit cable and reaches the cable plant's Head End, where it is

transferred to the receive cable. The receiver detects a signal, synchronizes, and sends data to a translation device that will convert the broadband modulation scheme into the Manchester encoded signal that the controller expects. If proper network installation procedure has been followed, the transceiver can guarantee that everyone on the cable has seen and understood that there is activity on the broadband cable. A comparator device compares the incoming source address with the address sent out on the transmit side as the Manchester encoded signal travels from the translation device to the controller. If the two addresses do not match, the transceiver sends a collision signal to both the controller and the other receivers on the broadband cable. As usual, if the controller sees the collision signal, it will abort the packet it is transmitting.

## 2.2. Implementation

Digital Equipment Corporation (DEC) offered a broadband transceiver that fit our needs and specifications. Their specifications indicated that these transceivers adhered to the Institute of Electrical and Electronic Engineers (IEEE) 802.3 standard as it existed in draft form. These were procured, tested, and found to be compatible with our broadband LAN system.

The initial complement of workstations for the DISE contained both types of controllers within the different units. The controller manufactured by 3COM (Model 3C400) adhered to the Type I Ethernet. The other controller was presented as a SUN product, and it was capable of adhering to either the Type I or the Type II version of Ethernet. All were configured and capable of being operated using Type I Ethernet, but, the DEC transceivers that we bought required the use of Type II Ethernet. Fortunately, the solution to this problem was simply to exchange the 3COM controllers for SUN controllers.

With all the workstations using the same Type of Ethernet, the issue of connecting the SUN workstations to our broadband cable system was addressed. Problems existed with the SUN workstations communicating on the broadband LAN when the DEC transceivers were connected. Everytime the workstation would emit a "keep-alive" signal on the Ethernet, an error appeared. Yet, when the SUNs were connected to a baseband cable system, using baseband transceivers, the SUNs had no problem in communicating to one another.

Based upon broadband transceivers, whose only purpose is to broadcast packets into the cable network, the SUN controllers were configured to listen for their own packets while those packets are being transmitted. This is possible because there is a small "turnaround" time involved between the time a baseband transceiver transmits and the time it receives its own transmission. The controller will scan the incoming packet for the Ethernet packet preamble. It expects to see the preamble at about the same time that it is sending the destination address of the packet to the transceiver. If the controller receives no data at all from the transceiver, it will assume something is wrong and abort the transmission.

The broadband transceivers purchased from DEC were guaranteed to fit IEEE 802.3 standard specifications, but this really applies to the interface between the controller and the transceiver. The internal functions that were being performed, in addition to the cable delay, required much more time than the baseband counterparts.

SUN apparently based their controller implementation on data available for baseband systems, and so, the controller would "timeout" before the broadband

transceiver ever had a chance to transmit and perform its own internal checking. Thus, for any attempt at network communication: the SUN controller would hand off the packet to the transceiver, the transceiver would "warm-up" and start to transmit the packet, the SUN controller time period for the transceiver to send a packet back to it would expire, and the controller would abort the transmission signal being sent to the transceiver.

In an effort to isolate the cause, DEC transceiver or SUN controller, a copy of the IEEE 802.3 draft standard for CSMA (Carrier Sense Multiple Allocation) networks was reviewed. Unfortunately, the IEEE draft standard does not technically reference the procedure of a controller aborting the transmission of a packet if it does not receive data from the transceiver within a specified amount of time, nor does the standard set a time period within which the transceiver must provide part or all of the transmitted packet.

The solution to the problem was ultimately gained through software modification. The heart of the SUN controller is an Intel 82586 network coprocessor. This processing unit reads a configuration table sent to it by the SUN workstation processor during system initialization. One of the parameters in this configuration table allows the network coprocessor to enable or disable Receive Data Sensing, the capability that allowed the controller to listen to its own data packets. Within the SUN Unix kernel, there was a table that controlled this function.

The Unix kernel was patched so that the Carrier Sense feature of the network coprocessor was disabled. Without the DEC broadband transceivers this would be disastrous, for the controller would never know if a packet has sucessfully traversed the network. With Carrier Sense turned off, the controller would be

willing to wait an infinite amount of time for the data packet to return. However, since the DEC transceivers check the source address we had the flexibility to push this function into the transceiver as opposed to keeping it in the controller.

## 3. Heterogeneous DOS Capability

### 3.1 DOS Selection

While the network aspects of the DISE were under completion, attention centered upon the selection of a Distributed Operating System (DOS) that would support the required level of interoperability. For our purpose, a DOS is defined as a system of machines capable of providing not only remote communications and access to remote resources, but also the ability to effectively adapt and manage those facilities beyond the traditional master-slave relationships. Beyond this definition, there were several more conditions that the DOS must satisfy in order to fulfill both the established goals of DISE and other RADC research paths.

System heterogeneity was vital. While homogeneous systems can easily be optimized for performance, it was not viable to take an entrenched position regarding new technologies that might be introduced. By compromising the heterogeneity issue, we would have also been refusing to acknowledge the existence of the "real" world, where organizations that have heavily invested in computer hardware and software from different vendors are not going to be willing to write-off that investment.

The system needed to be based upon a simple model that would allow a consistent and uniform view for both operating system and application. This model needed to

be reasonably flexible so that we might incorporate or integrate with other distributed systems in the future.

Expanding the hardware base, required the consideration of a factor dealing with "ease of integration". Because the testbed was going to be extended to incorporate new machines, it had to be concerned with the effort that would be expended in bringing these new systems under the DOS umbrella.

Finally, a system that offered a low cost and expandable solution to a DOS was essential. This meant working with existing technology, standards, and protocols. This was not hard to accept, when considering the tradeoff penalty that was expected to occur with heterogeneity. The ultimate approach was to prove the feasibility of the concepts as a first step. The software optimizations would come at a later time.

### 3.2 Cronus Integration

Cronus, a DOS created and developed by researchers at Bolt, Beranek, and Newman, satisfied the DISE needs. The Cronus DOS resides atop the existing (native) operating system, and, looks like an application from the native operating system's perspective. This allows new machines to be integrated without losing the native operating system and all of its accumulated capabilities. Cronus is designed to be portable, and, it succeeds.page lengthl by using the C programming language and well defined interfaces to the native operating system. All Cronus activities (Services) are built on top of the Cronus Kernel. This module mediates the Cronus processes that are resident on the host, as well as providing the communications interfaces to external hosts. Data that is passed between machines is encoded and decoded using a canonical form that allows transformation

between the different internal data formats of machines. These communication and data transformation components are the major pieces that must be rewritten in porting Cronus from one host to another host of different type. Since Cronus layers its functionality on top of the Cronus kernel, no other modules are usually changed.

By using the object model throughout its concept, design, and implementation phases, Cronus provides an extremely flexible environment where the distinctions between system and application Services are distinguishable only in the mind of the user. For those unfamiliar with the object model, it is a design philosophy that was developed throughout the seventies. In brief, it proposes that activity can be viewed through an abstract datatype and an operation. Thus, an automobile could be broken into an "object" of fixed data and parameters that describe the state of the auto, and, a set of "operations" (Create, EngineOn, EngineOff, Accelerate, LeftTurn, RightTurn) that *manipulate* the state of the automobile contained in the object.

The only problem encountered with the Cronus integration occurred with the SUN workstations. SUN's operating system is based on the 4.2 Berkeley implementation of UNIX. Specifically, when dealing with the network software, it was found that Berkeley, and by default SUN, defined their network broadcast address by having the host portion of the net address filled with zeroes. Cronus adheres to the DDN Internet standards that require broadcast addresses to be formed by filling the host portion of the net address with ones. This problem was solved by a suitable patch for the UNIX kernels of the SUN systems, which was installed by reconfiguring the kernel and remapping subroutine pointers using the UNIX debugger. Once this was complete, the remainder of the installation was free of trouble as libraries and source code was being compiled and stored.

## 3.3 Post Installation Observations

With the installation complete, the DISE incorporates a DOS with most of the functions that were needed for application development. The Cronus DOS provides a transparent communication medium between objects by using the simple primitives Invoke (send) and Receive. Cronus Managers (processes) supervise the Cronus Objects (datatypes) and perform the operations invoked on its Objects. A hierarchical Catalog maintains a mapping between the user generated symbolic names and the 96 bit system name (called a UID) generated for every Object. User Authentication and Access Control functions exert control over the unauthorized manipulation of objects A global Filesystem with replication capability is also provided.

Perhaps, the one entity that is most lacking in Cronus is the software development environment. There are no low-level software development tools (editors, compilers, debuggers) that have been written for the Cronus environment, nor, are there any native system tools that have been totally integrated into the Cronus environment. Software development has been primarily done within the native host domain using any available tools. Some attempts have been made at integrating the native tools, but, inevitably, this involves modifying the native operating system source code present on a host. Moreover, this process would have to be repeated for each new version of software that was released, thus leading one to question the ultimate utility of such action.

A high-level development facility does exist to assist users in generating code for Objects and Managers (See Figure 2). The user defines an Object specification which is automatically checked for correctness and stored on-line

within the DOS. Operations for the Object are then coded, using a native editor, and a preprocessor is used to build a manager around the operations. The preprocessor is used to generate most of the redundant code (type conversion, communication, etc.) found in most Managers. Standalone programs have been designed and created to allow operations to be directly invoked on objects. These programs, in addition to the native debuggers, are the tools used to debug the new Manager. While there are few tools written primarily for the Cronus DOS in the mentality of the Cronus DOS, we are undertaking contractual efforts to evaluate and recommend any action that should be taken in these areas.

The environment and flavor the of the Cronus DOS is very reminiscent of UNIX, even though the DOS has been ported onto different operating systems. Most of the development work at BBN has been completed on UNIX based systems, which leaves some concern regarding the efficiency and stability of their software that has been written for other operating systems. A Cronus kernel on a VMS system has been implemented, but data on the VMS supported environment and the overall performance will not be available until an evaluation within the DISE is completed in Mid 1987.

There is also a concern about the mortality of replicated Managers. During the initialization phase of a newly started Manager, a replicated Manager will tend to crash if it receives an operation to be processed that is not related to the Manager's initialization. This occurrence has been observed in at least two cases: the Authentication Manager, and a Timing Manager used in a sample application that is distributed with Cronus. The appropriate response would be for the manager to distinguish between classes of messages, so that it can ignore all classes of operations except for those used in its initialization.

## 4. Wide Area Network Access

### 4.1 Connectivity

One of the technical goals for the DISE is to investigate the problems and relationships that occur when remote computer systems attempt to communicate and manage each other when divided by different geographic distances. To facilitate this, the DISE is connected to both the ARPANET and the WIDEBAND Satellite Network.

In choosing a gateway (machine dedicated to handling communication traffic between networks), a choice had to be made between the LSI-11 and the BBN Butterfly. The LSI-11 gateways were stable and proven machines that were reasonably represented throughout the network community. In 1984, there were 24 gateways of this type, with the number rising to 30 at the present time. Based on these numbers and recommendations from outside RADC, it was felt that this would be a safe investment. At the same time, one of the new BBN Butterfly gateway prototypes became available and was installed. With the Butterfly, a main concern was the reliability of a new technology (the multiprocessor system) as well as a new application (gateway service). The benefits were, at least theoretically, greater performance than any of the other gateways in the system. In the final decision, both gateways were selected.

The LSI-11 gateway is intended to be used as a standard benchmark against which we could compare the BBN Butterfly. Also, if the Butterfly does not live up to its reliability claims, the LSI-11 can provide backup capability. The dual cluster will also provide the opportunity to pursue research in the dual connectivity of remote clusters.

With two gateways, it is possible to experiment with the difference between the ARPANET and WIDEBAND networks. ARPANET offers a reliable land-line telephone network, that is widely used and possibly congested. The WIDEBAND network offers a significant increase in bandwidth with little congestion, yet, several drawbacks exist in using WIDEBAND. It is not currently possible to choose a predetermined path through the Internet environment because all network switching information is both contained within and controlled by the Internetwork gateways. Because this capability is needed for experimentation, the best solution would be to dedicate each gateway to a particular network.

Using the Butterfly, in conjunction with the LSI-11 gateway between the ETHERNET and the ARPANET, the strategies and options available in determining the "best" route for message passing in the loosely coupled environment of multiclustered systems will be investigated. Setup time for ground-satellite-ground scenarios must be considered when choosing the WIDEBAND network over the ARPANET. This can be aggravated with packet switching protocols which limit the number of packets that can be "floating" through the network by waiting to receive a packet reception confirmation by the destination host. A small number of floating packets would not effectively utilize the network. A large number may increase problems of packet retransmission or network failure determination.

## 4.2 Butterfly Implementation

The BBN Butterfly is the latest generation of a multiprocessor system that has emerged from BBN. Based on its high speed BBN Butterfly Switch, the Butterfly links the processors to spread the overall workload present on the machine. Each machine configures itself at power up or restart. Dynamic configuration for

processor failure while the machine is running is not available. The Butterfly machine that was provided to us has 5 processors, 4 for applications and 1 for control, connected through the Butterfly Switch. Each processor has one megabyte of memory that is resident on the processor card. To facilitate the multicluster research, the Butterfly has been configured as a 3 legged gateway bridging the ARPANET, WIDEBAND Satellite Net, and the DISE local broadband ETHERNET.

After installation, intermittent failures, lasting about 1 to 1.5 months, were isolated to a faulty communications board that eventually was replaced. The second problem, this time fairly minor, was rooted in the manner that the gateway dealt with message packets that had been corrupted slightly by the WIDEBAND satellite system. This was promptly repaired through a kernel patch, and then standardized in the following release of the gateway software. The third problem, most recent and more costly, has affected gateway operations for the past 3 months. The gateway was unable to function for any appreciable length of time. During the worst period, the software would attempt to restart the machine as much as 30 to 40 times every day before giving up and requiring operator intervention. This problem was found to be an uninitialized variable in the software and was fixed in mid January of 1987. These problems have served to confirm our impression of the Butterfly gateway as a prototype model, with normal reliability problems. Since it remains the biggest factor in any network experiments, continuing efforts will be expended to solve these problems.

## 5. Current Implementation

The DISE currently consists of all of the host machines shown in Figure 3. With the exception of the VAX 8650 and the microVAX II, all of the other hosts are

accessible through either a Broadband Ethernet or the ARPANET. Currently, only the SUN systems have been configured for the DOS by receiving a kernel. We are in the process of getting a working DOS kernel for both our MASSCOMP and VAX 8650 machines. The DECSYSTEM 20 should be integrated into the DOS environment within the 1988 to 1989 frame of time.

There is a working connection into both the ARPANET and the WIDEBAND Satellite Network through a BBN prototype gateway that uses the BBN Butterfly multiprocessor. We are currently working to get the LSI-11 gateway operational.

# Future Research

## 1. General Topics

The primary focus of activity in the past has been devoted to stabilizing the cluster at RADC. Future efforts will extend our research and development goals beyond the Cronus DOS perspective. Investigations will begin to study the effects of instrumentation and the associated mechanisms needed. There are also investigations to be made into the process and procedure for writing distributed applications in an environment tailored for distributed systems. Longer range goals include some experiments within the realm of secure systems, and in particular, the requirements and designs necessary to support levels of security in the DOS environment.

## 2. Intercluster Activity

By tying the DISE cluster to the cluster already present at BBN, approaches, and possibly algorithms, will be developed for protocols capable of handling the dynamic nature of one or more connection paths that are themselves dynamic. In the DOS mechanisms, all communication occurs through message passing. The current method of handling these transfers of information is to use the DOD standard packet switching protocols present throughout the DOD Internet environment. With the advent of another available network, better routing, status, and control mechanisms may be needed to handle the various level of message processing, based on reliablity, time-critical, and congestion factors.

Multicluster resource and configuration management topics will also be investigated. Because both the DISE and BBN clusters will wish to develop their respective distributed systems, configuration control policies and mechanisms must be designed and implemented. Thus, the resources, available at every cluster based upon our DOS principles, can be collectively shared. Configuration control must also allow development that may carry each cluster into explorations of new architectures and capabilities requiring the current standards and criterion to change.

3. Minicomputers vs. Mainframes

Experimentation is underway to investigate the benefits of substituting minicomputers for mainframes whenever additional processing power is needed. An issue to be considered regards the addition of low-cost processing units into an existing integrated system. Would these additions lead to increased performance benefits without the traditional expense of buying a mainframe and growing into it? Moreover, if integrating the DOS environment down to the process level is fully successful, users will not see a traditional machine environment, but a uniform DOS perception of processing capability. In this way, the distributed system would choose the best processing environment for the user's program, and, the user would be unaware of the run-time environment (minicomputer or mainframe).

4. Instrumentation

Instrumentation in a centralized machine is important, but, in a distributed system the need can be critical. While little attention has been paid to this

area in the design of systems, it is desirable to define a set of primitives that allow one to effectively establish, monitor, and control the state of a system. More than a paper study, an attempt will be made to: integrate studied concepts into the distributed system, emerge with a set of principles, and demonstrate implemented capabilities that may be of benefit to all systems. The target is to help the system designers that need some level of validation for their distributed programs. Initially, a passive extracting system will be used for implementation. The flavor will be similar to current debuggers, where the instrumentation can be compiled in or left out.

5. New Machine Integrations

Two major integration activities include bringing a DECSYSTEM 20 and a microVAX workstation into our DOS environment. The DECSYSTEM 20 is a task that has been allocated to our BBN contractors. The DECSYSTEM 20 will serve as a test case for integrating an older technology machine that is radically different from experiences thus far. The microVAX will provide the opportunity to further investigate and determine those DOS kernel functions that are necessary and appropriate. From there, the plan is to use the microVAX as an interface into a IDM 500 database machine and create a database capability within the DOS.

6. DOS Kernel Architectures

To integrate the microVAX VMS system will require an examination of the current DOS kernel architecture in order to see if there are other approaches that are appropriate for implementing the current set of mechanisms. The overall goal is to increase the efficiency of the interprocess communication mechanisms, yielding an increase in the speed of the system. Thus, one of the inhibitions toward

building distributed applications, the concern of communication speed and reliability, will be alleviated.

7. Demand-Scheduling

While heterogeneity inhibits this distributed system from becoming a real-time system, some of the current work in real-time scheduling could be appropriate for the system. Some effort may be directed into developing a model and prototype for a migratory process that is heterogeneous in nature. If able to fully implement a migratory process (mobile rather than replicated), then, the ability to move processes from one host to the next necessitates a need to accurately schedule work among the processors involved.

References:

Institute of Electrical and Electronic Engineers; "Local Area Network (Carrier Sense Multiple Access with Collision Detection - CSMA/CD)"; IEEE Standard 802.3; IEEE Press; 1985

R. Schantz, R. Thomas; "Cronus, A Distributed Operating System: Functional Definition and System Concept"; BBN Report 5879; Bolt Beranek and Newman Inc.; January 1985

R. Schantz, et. al.; "Cronus, A Distributed Operating System: Revised System/Subsystem Specification"; BBN Report 6248; Bolt Beranek and Newman Inc.; May 1986

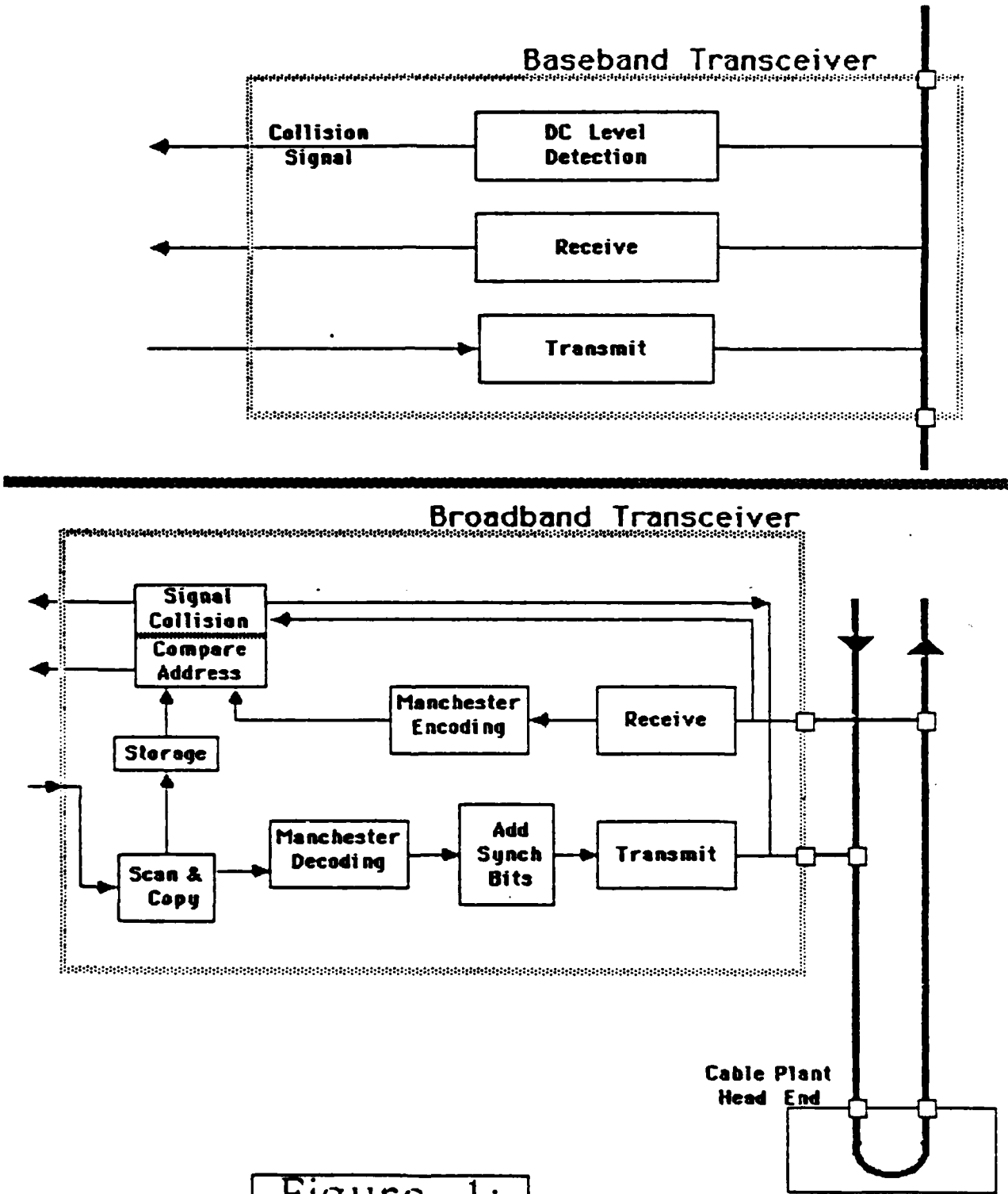
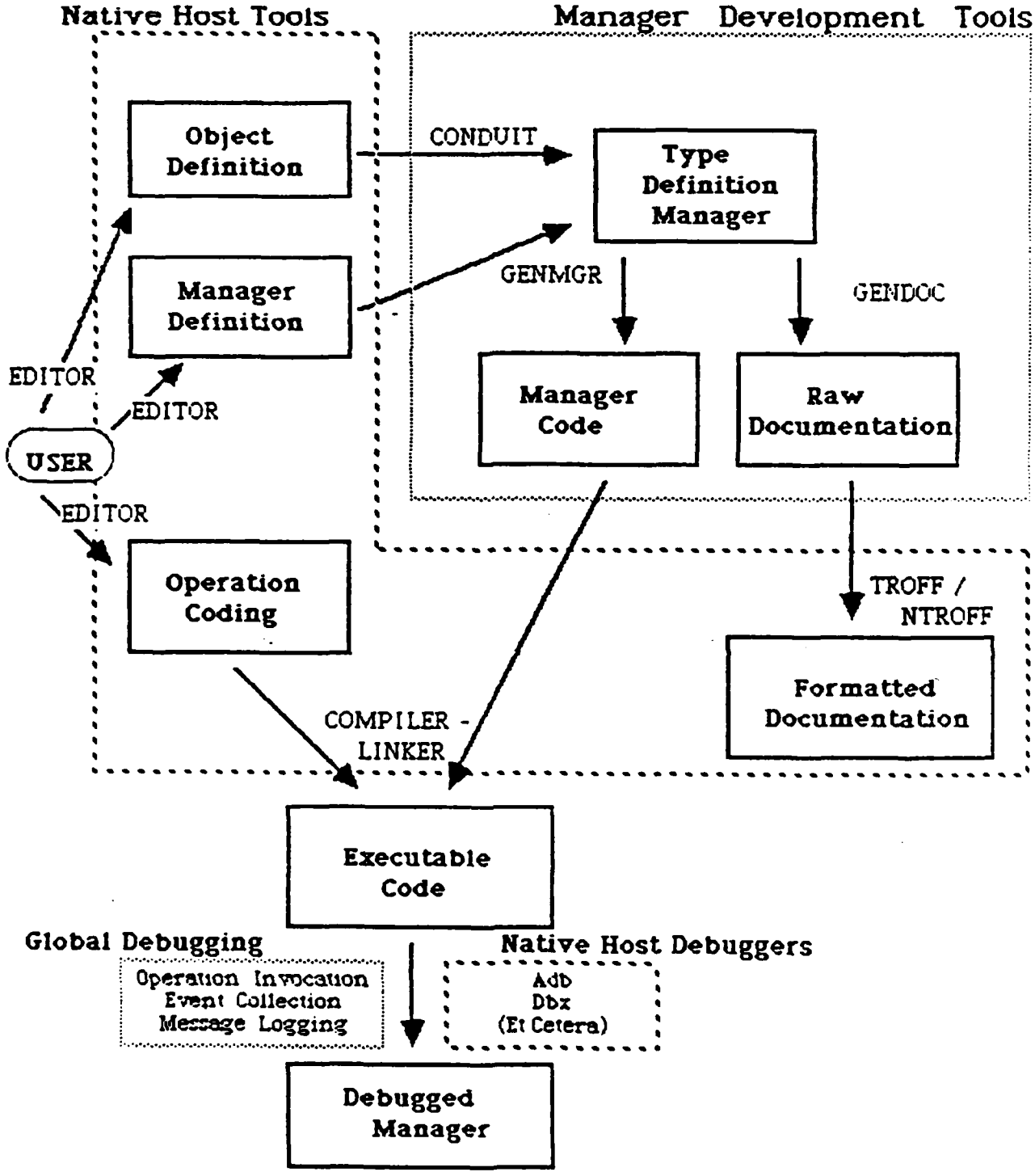
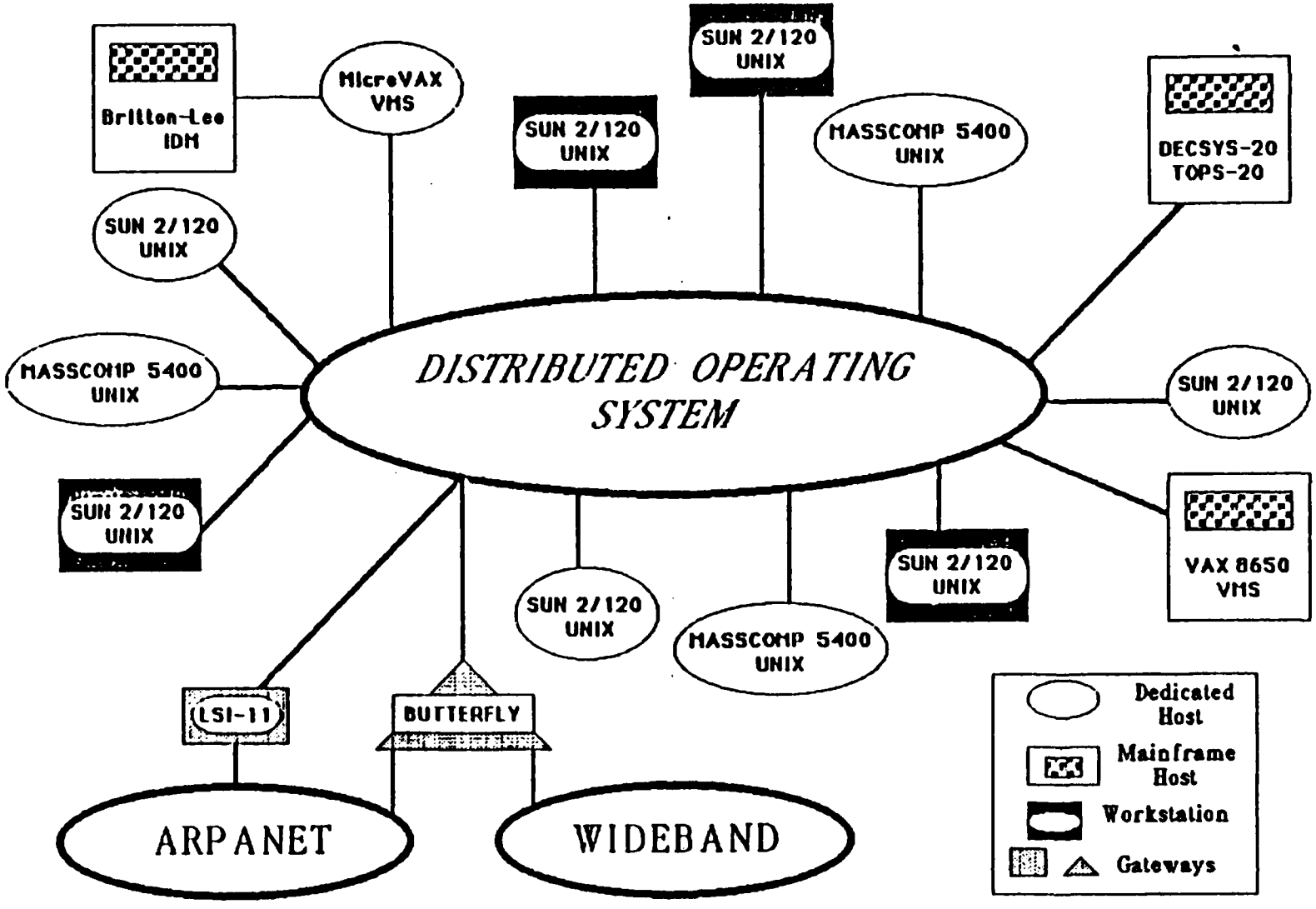

Baseband Transceiver
Collision Signal
DC Level Detection
Receive
Transmit
Broadband Transceiver
Signal Collision
Compare Address
Manchester Encoding
Receive
Storage
Manchester Decoding
Add Synch Bits
Transmit
Scan & Copy
Cable Plant Head End

Figure 1:




Cronus Software Development Environment

Figure 2:




*DISTRIBUTED SYSTEMS EVALUATION ENVIRONMENT (DISE)*

Figure 3:

# *MISSION*
# *of*
# *Rome Air Development Center*

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence ($C^3I$) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of $C^3I$ systems. The areas of technical competence include communications, command and control, battle management, information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic, maintainability, and compatibility.

END

8-87

DTIC