

DTIC FILE COPY

3

**RADC-TR-88-133, Vol II (of three)
Final Technical Report
June 1988**



AD-A200 170

THE C² SYSTEM INTERNET EXPERIMENT Functional Description

BBN Laboratories Incorporated

James C. Berets, Ronald A. Mucci, Richard E. Schantz and Kenneth J. Schroder

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

DTIC
ELECTE
NOV 07 1988
S **D**
D **ck**

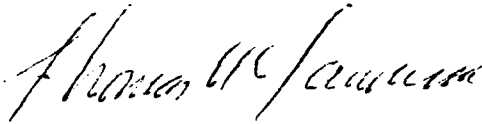
**ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss AFB, NY 13441-5700**

88 11 07 098

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

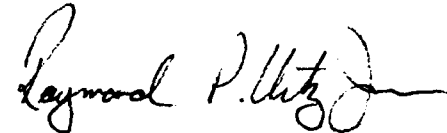
RADC-TR-88-133, Vol II (of three) has been reviewed and is approved for publication.

APPROVED:



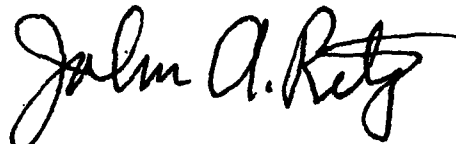
THOMAS F. LAWRENCE
Project Engineer

APPROVED:



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

FOR THE COMMANDER:



JOHN A. RITZ
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 notice on a specific document requires that it be returned.

AD 173

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS N/A		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 5942			5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-88-133, Vol II (of three)		
6a. NAME OF PERFORMING ORGANIZATION BBN Laboratories Incorporated		6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COTD)		
6c. ADDRESS (City, State, and ZIP Code) 10 Moulton Street Cambridge MA 02238			7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Rome Air Development Center		8b. OFFICE SYMBOL (if applicable) COTD	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-84-C-0140		
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700			10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO. 62702F	PROJECT NO. 5581	TASK NO. 21	WORK UNIT ACCESSION NO. 70
11. TITLE (Include Security Classification) THE C ² SYSTEM INTERNET EXPERIMENT - Functional Description					
12. PERSONAL AUTHOR(S) James C. Berets, Ronald A. Mucci, Richard E. Schantz and Kenneth J. Schroder					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM Aug 84 TO Mar 86	14. DATE OF REPORT (Year, Month, Day) June 1988		15. PAGE COUNT 48
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	Distributed Operating System, System Monitoring and Control, Interoperability, Heterogeneous Distributed System, Survivable Application		
12	07				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>This is the Final Technical Report representing work performed for the C2 System Internet Experiment project. The C2 Internet Experiment project is a complementary effort to the CRONUS Distributed Operating System (DOS) project. Under the C2 Internet Experiment, we designed and built a prototype distributed application to demonstrate CRONUS concepts and to provide an environment for evaluating CRONUS' current suitability for supporting distributed applications, especially applications that pertain to the command and control environment. Both the C2 Internet Experiment and CRONUS DOS projects are part of a larger program being supported by the Rome Air Development Center (RADC) to design, develop and support an appropriate technology base for building the types of distributed systems which are expected to be fundamental in future command and control applications and elsewhere.</p> <p style="text-align: center;"><i>Summary include</i></p>					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Lawrence		22b. TELEPHONE (Include Area Code) (315) 330-2158		22c. OFFICE SYMBOL RADC (COTD)	

Table of Contents

1. INTRODUCTION	1
1.1 Background	1
1.2 The C ² Internet Experiment	3
1.3 Selecting an Application	4
1.4 The Cronus Environment	6
1.5 Report Overview	7
2. THE C² INTERNET APPLICATION	8
2.1 Application Architecture	9
2.2 C ² Internet Experiment Components	12
2.3 Monitoring and Control	18
2.4 The User View	18
3. EVALUATION OF THE DISTRIBUTED SYSTEM	20
3.1 Evaluation Criteria	20
3.1.1 The Cronus System Environment	21
3.1.2 The Cronus Model	24
3.1.3 Application Programming Support	31
3.2 Evaluation Methodology	34
4. EXTENDING THE EXPERIMENT	36



Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Date _____	
Availability Codes	
1 2 3 4 5 6 7 8 9 A-1	For and/or Special

FIGURES

C ² Internet Experiment Processing Functions	10
C ² Internet Experiment Architecture	13
Potential Two Cluster Configuration of C ² Application	22
The Cronus Object Model	25
The Cronus Kernel Interprocess Communication Concept	30

TABLE

Distributed System Requirements and Support

33

1. INTRODUCTION

This document describes the command and control application that will be used in the C² Internet Experiment in order to evaluate the utility of Cronus tools and concepts for building complex distributed systems. Functional descriptions of the constituent subsystems of the planned experiment are presented here. Also discussed is how the various components of the application will be used to demonstrate the attributes afforded by a distributed architecture in conjunction with Cronus distributed operating system support.

The implementation of components needed to support the experiment application has already begun. A companion document, **The C² System Internet Experiment: System/Subsystem Specification** [BBN 6248], describes these components and their implementation in more detail. **The C² Internet Experiment: Final Technical Report** [BBN 6251] reviews our experience during this phase of design and implementation of the experimental application, and evaluates the role of Cronus in this work.

1.1. Background

Command and control systems are inherently information intensive and complex; hence it is inevitable that the implementation of many future command and control systems will rely extensively on computers. Furthermore, these systems will be physically distributed and redundant in order to achieve survivability (see, for example, [ADDCOMPE, TRW]); thus they will require a distributed architecture.

Once the idea that future systems will be distributed has been accepted, there are a number of desirable characteristics which the support architecture should exhibit. Functional specialisation generally

necessitates that different types of computers will be available and used to solve different parts of an overall problem, yet the result must continue to perform as an integrated system despite heterogeneity. System distribution should not diminish the usual requirements for effectively controlling the management of system resources or the desire to control access to these resources. However, it does render ineffective current mechanisms in these areas which are limited to a single computer system. Experience has shown that large complex systems cannot be conceived whole and remain static, but rather need to evolve with time, changing requirements, and technology. No matter what support components are chosen for an implementation, there will inevitably be growth and changes that need to be accommodated. The sheer number of components typically involved in a distributed system architecture results in a situation of constant change.

If one were to initiate the development of a distributed system of heterogeneous components using available off-the-shelf technology, one would by necessity be starting with a collection of independent host computers, some available communication medium, and some standard software communication support for transporting uninterpreted data between hosts. The gap between the problem-oriented requirements and the system-oriented support is huge. As a consequence, the application designer is forced to deal directly with a variety of support areas to fill this void. This added burden will at best make such projects much more complex and expensive, and at worst cause many of them to fail. At best, there will be limited interoperability between different systems.

One approach to changing this situation is through the development of a distributed operating system (DOS) and its support functions. A distributed operating system provides computing and communication resources to application programs, while promoting resource sharing among interconnected computer systems, and managing the collection of resources that are shared. A DOS bridges the gap between applications and communication by providing a coherent and integrated system for supporting

the development and operation of distributed applications.

Recognizing that such a gap existed, the Rome Air Development Center (RADC) has for a number of years been supporting research into the design and implementation of distributed operating systems. In 1981, we began the design of the Cronus distributed operating system, which was an attempt to capture and extend much of what had been learned from previous research. The intent was to provide a testbed for application developers to gain hands-on experience with designing, building, and evaluating distributed applications relevant to their particular domain. The Cronus testbed provides application developers with exposure to the Cronus tools for developing distributed applications, and direct experience with the effects of distribution on their application. As a means of testing and evaluating the implementation of Cronus to date, we are now developing a set of exemplary applications.

1.2. The C² Internet Experiment

The C² Internet Experiment is a testbed approach for demonstrating the utility of the Cronus tools and concepts in constructing complex distributed applications. The objectives of the C² Internet Experiment are to design and carry out an experiment demonstrating the applicability of Cronus as a support base for command and control applications. The command and control application should be based on a distributed architecture to demonstrate interoperability, and make use of redundant resources to demonstrate survivability. In addition, there are a variety of other Cronus system properties which derive from system distribution and are desirable attributes for a variety of computer applications, including command and control systems (see [BBN5879]).

Despite almost unanimous agreement on the desirability and applicability of a distributed system architecture for C² applications there is little experience base, and hence very little agreement as to how

to effectively utilize distribution. A testbed and evolutionary system design approach is being taken in both the development of the system support layer and in the development of application systems to account for this uncertainty. In this way, we construct prototypes for immediate evaluation of both the impact of distribution on the application and on the applicability of system tools for constructing such applications. With only a limited investment in any particular approach, modifying or abandoning those that do not seem immediately promising is not a major undertaking. Additionally, positive initial implementation experience with a mechanism or approach gives rise to a higher degree of confidence in making this aspect a base for further development.

Another key aspect of the Cronus testbed approach is the decomposition of large systems into small, manageable units. In this way, developing a large complex system is a two step process: developing a system architecture (and application architecture) which anticipates how the small, manageable units will interact, and developing the small units themselves. Aspects of system distribution and Cronus system support apply equally to both parts. In some cases system distribution applies across functions, while in other cases it may apply within a function. The Cronus distributed system design methodology is fundamental to this type of approach. It supports experimentation with both the structure and the elements of the system. Seen in this way, the distributed system testbed is also evaluating the construction methodology and its applicability to large command and control systems.

1.3. Selecting an Application

The variety of dimensions to which distributed system approaches apply has an impact on the selection of an experiment application. If the application is too limited or focussed, we are sure to miss important areas for evaluation. If it is too large and dependent on details, we will never make any initial progress. An application domain is desirable that includes a rich set of distributed system problems, is

adequately demonstrated without excessive attention to the details of the application, and is extensible so as to incorporate anticipated improvements in technology and future Cronus system developments. To make the test and evaluation more meaningful, it was also important that the application be selected from a context which was immediately recognizable as pertinent to planned or desired usage patterns. On the other hand, it should also be clear that the use of the underlying system is not limited to the specific application selected.

Our approach is to focus on the command and control cycle itself, not any single aspect of it. Most command and control systems may be characterized by a repeated decision-making cycle. The four major actions that comprise this cycle are [Cushman]:

1. Sense the situation and understand it.
2. Consider the situation and decide what to do about it.
3. Execute the decision (or issue instructions for its execution).
4. Continue to sense the situation, thereby starting the cycle again.

Virtually all C² systems may be decomposed into these functions. The C² Internet Experiment encompasses at least parts of each of the elements composing the cycle. It emphasizes extensibility and applicability to a variety of specific command and control contexts. The specific application area is a set of integrated but independently developed battle management functions for detecting and directing an engagement with ground targets. This application suite was selected in part because of its current relevancy to existing projects at RADC and BBN which are attempting to support one or more aspects of this problem.

The suite of application components provides a rich set of current distributed system design and implementation issues, and has a number of naturally occurring extensions in areas planned for future Cronus development. We hope to have both the system and application evolve together in a symbiotic

relationship: extensions to the Cronus system are demonstrated and evaluated in the context of the C² application, which in turn may motivate the selection and direction for some future extensions.

1.4. The Cronus Environment

There are a variety of uses of Cronus in the experimental application. One important role is to support and control interoperability between resources independently developed on various host computers. A second role is to support the development of integrated functions or subsystems in an environment of heterogeneous machines. A third role is to support the gradual evolution from interconnected but largely non-integrated computing services toward integrated services better able to capitalize on their interconnection. Through this experiment we are validating the adequacy of the DOS support mechanisms for constructing systems of this type, and providing insight into making such a testbed more effective for developing other distributed applications.

The Cronus system is designed to support a number of the envisioned system requirements for future automated command and control systems. A previous report [BBN5879] outlines these system requirements, which will be briefly reviewed here. Cronus operates in an environment consisting of integrated clusters of interconnected heterogeneous computer systems. Heterogeneity of both hardware and software resources is both a practical reality and an attribute to support functional specialization envisioned for future command and control applications. The primary requirement for any development activity in this context is to have the distributed system behave as an integrated system despite the distributed, heterogeneous nature of the support base. Toward this end, Cronus develops an integrated system model applicable to both the development of the system and its applications.

The system model is based on decomposing the specific application into interacting functional elements, designing these functional elements to achieve desired global properties, and supporting their interaction. Within this model there are mechanisms and support for survivability through replication of machines, functions, and data; for scalability of functions to support systems ranging from limited needs to large scale requirements; and for global resource management to make effective use of available system resources. These system attributes are supported in a manner which makes their usage customizable to the resource or function in question. In addition to providing support to achieve these properties for application specific functions, Cronus itself also provides a variety of "system" functions, which will be utilized to support the implementation of the selected application. Primary among these are system-wide access control and authentication functions, global naming functions, global file management functions, and system-wide monitoring and control functions.

A major attribute of Cronus system support is that it is designed to be extendable to and customizable for new application systems and evolving system hardware and software. Finally, to complement the available functional support, Cronus also provides a number of high level software development tools which simplify programming in a distributed system environment (see [Cronus], [DCS]). These Cronus development tools will also be evaluated as part of the C² Internet Experiment.

1.5. Report Overview

The remainder of this report will discuss the particular application chosen for the C² Internet Experiment, and how it will be used to demonstrate and evaluate the facilities provided by the Cronus distributed operating system. The proposed evaluation methodology will also be briefly discussed, as will the current Cronus testbed.

2. THE C³ INTERNET APPLICATION

The C² Internet Experiment application is loosely based on the Ground Attack Control Capability (GACC) concept which is being developed by RADC "to function as the principal agency of the Tactical Air Control System for decentralized execution of attacks against selected ground targets. The GACC is responsible for the management and control of assigned air assets engaged in the detection, classification, attack and destruction of selected time-sensitive targets on a time urgent basis." [GACC]

Of particular importance in the development of the GACC is the rapidity with which decisions can be made. Present ground attack capabilities accumulate a long delay between the time a situation is sensed and the time that action is taken on this situation. However, many ground targets are time-sensitive. That is, the usefulness of attacking these targets depends on acting quickly. Currently, much of this delay is incurred as information moves from place to place in the command and control hierarchy. The main goal of the GACC is to decrease the response time for handling certain ground targets by increasing the speed of information flow. Consistent with this major goal of the GACC is one important objective of the C² Internet Experiment which is to demonstrate a survivable distributed architecture capable of effectively managing the required transfer of information in a reliable and timely manner.

The core of the GACC's function is to supervise operations during the current operations period. Briefly, this includes the following functions:

- performing ground surveillance of the assigned area of operations,
- selecting the targets for attack,
- selecting weapons to attack the targets,
- pairing the selected weapons and targets,
- tasking the missions,

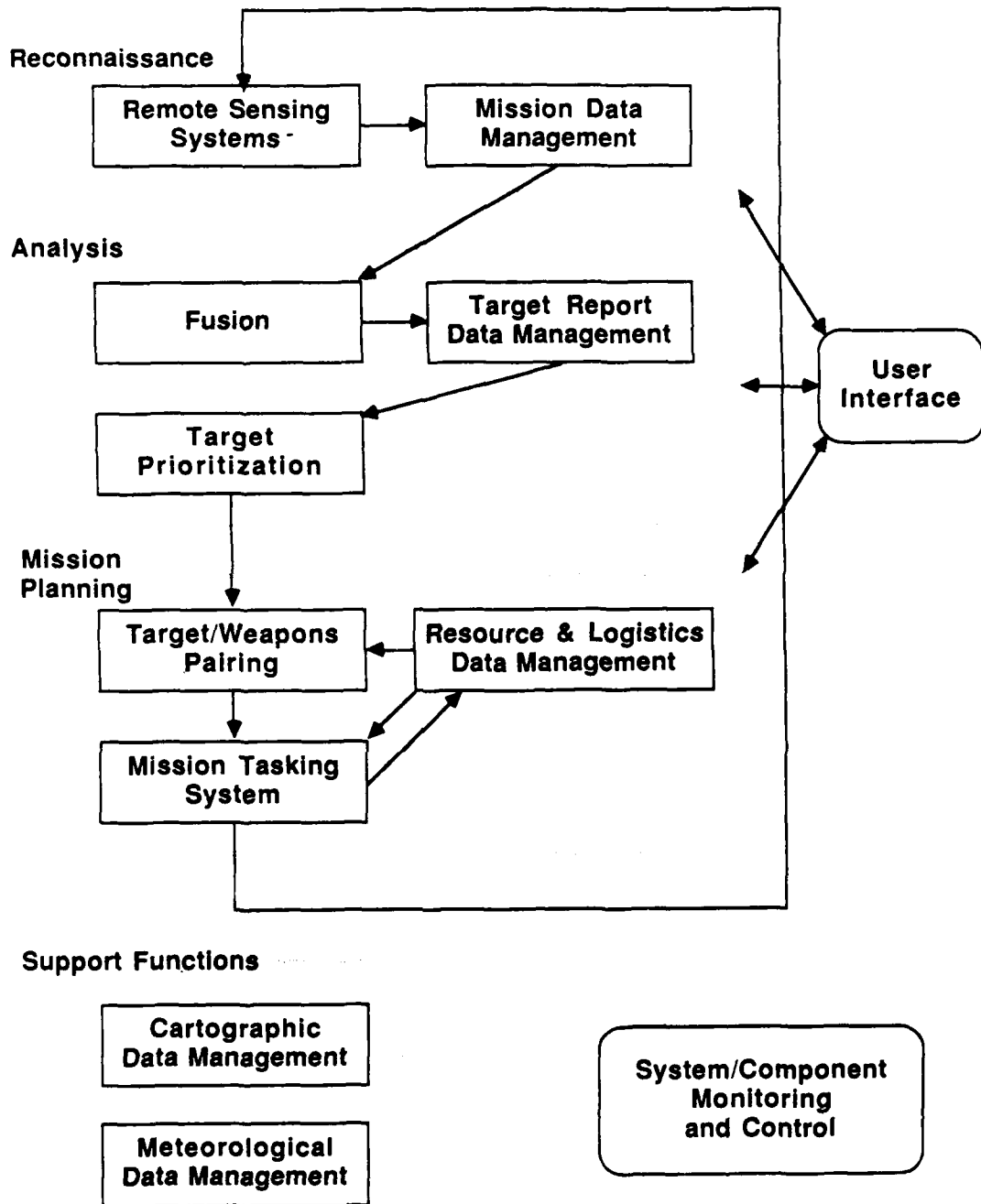
- and managing and controlling their execution.

A distributed system of this sort is believed to be inherently heterogeneous. That is, different parts of the system will probably be built by different contractors using different machines and at different times. The tendency will thus be one of incompatibility. The Cronus system architecture is intended to deal with problems of this nature by providing a means for tying together independent-but-cooperating heterogeneous application systems. In addition, the experiment is intended to highlight the use of Cronus in developing a distributed solution to a single C^2 functional area, such as target sensing and detection.

It is important to emphasize that Cronus is expected to be useful in a wide variety of applications, C^2 and otherwise. The experiment is largely a test and evaluation of Cronus capabilities as applied to a typical C^2 problem domain. As a result of the experiment we anticipate that many aspects of the Cronus distributed system architecture will be judged adequate as is. Others may be functionally adequate but deficient in performance or scaling to more operational environments and hence need extending, while others may be judged deficient and need replacement.

2.1. Application Architecture

The functions of the selected C^2 Internet Experiment application are shown in Figure 1. They consist of a collection of interconnected sources of data and processing elements which are the relevant constituent parts of a tactical air control system supervising attacks against selected ground targets. The overall processing is divided into three phases: reconnaissance, where the field situation is surveyed; analysis, where the reconnaissance data is processed to identify and categorize targets; and mission planning, where tactical resources are evaluated and assigned to deal with chosen targets. Support functions provide information needed for analysis and decision making procedures. User interfaces



C² Internet Experiment Processing Functions
Figure 1

provide access to the data, and support control of data collection and planning functions.

The actual processing begins as reconnaissance data is obtained by a variety of remote sensing systems. These datasets are processed to detect suspected target positions; these detections will include both actual targets and *false alarms* which cannot be distinguished without further processing. All detections are reported to the Mission Data Management System to ensure data availability regardless of the status of the sensor system which provided the data.

The target data recorded by the Mission Data Management System is then subjected to fusion processing, which combines information from several sensor missions and performs analytical processing to eliminate false alarms reported by the target detectors, to identify target movements, and to predict future target trajectories. The resulting information is stored in the Target Report Data Management System for use in later processing. Targets identified by the fusion processing are assigned priorities, to distinguish their overall strategic importance.

Weapons are assigned to incoming targets based upon available resources and the importance of the targets. Information to support this decision process is maintained by the Resource and Logistics Data Management System. The end result is the dispatch of weapons to deal with the incoming target. Additional reconnaissance missions may be dispatched to monitor the movement and condition of the targets, or to resolve uncertainties regarding the characteristics and mission of the targets. These weapons and sensor missions are reported to the resource and logistics data management system so that they will be available for later review.

Additional support data for all the processing functions is provided by the Cartographic and Meteorological Data Management Systems. These systems provide map and weather information for use in analyzing the tactical situation and for mission planning both by automated processing and by human

analysts. Monitoring and control functions are provided so an operator can review the health of the overall system and of individual system components.

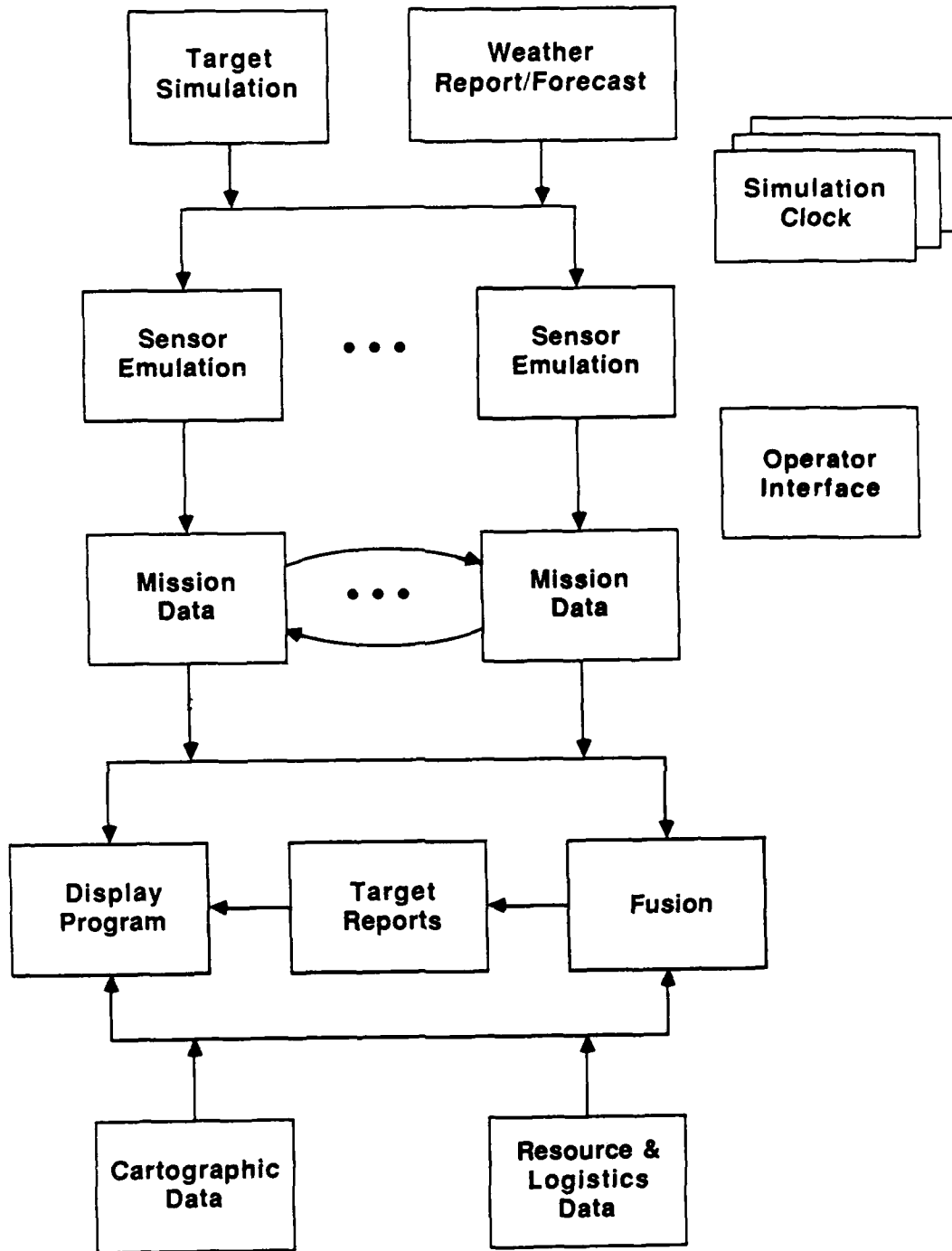
The functions described above will be implemented and distributed over the two clusters of computer resources available. Presently one cluster is operational at BBN. The second cluster is under development at RADC, Griffiss Air Force Base.

We plan to use the ARPANET to provide intercluster communication. The distribution of the application data sources and processing elements will be arranged to test fully the attributes afforded both by a distributed system architecture and the Cronus software tools and concepts within the constraints of the communication networks.

2.2. C² Internet Experiment Components

This section describes the constituent components of the C² Internet Experiment. These components support the functions of the selected application architecture, described in the previous section. In addition, the experiment architecture also includes components required to simulate elements and aspects of the application environment, such as the movement of targets and the effects of weather conditions on sensor performance, and components required to coordinate and control the progress of simulation. The components of the C² Internet Experiment architecture are shown in Figure 2.

It is important to note that the intent of the experiment is to demonstrate the utility of distributed system technology for a diverse set of command and control problems, not to achieve fidelity, at this time, with GACC or any other existing C² problem. The experiment components include existing software, "stubbed" software that provides or emulates essential functions of components which will be further developed in the future, and software designed specifically to operate in a multi-computer



C² Internet Experiment Architecture
Figure 2

environment. In this way, we can demonstrate our approach to evolutionary system development, where various components are at different stages of development, as well as demonstrating the interworking of the various constituent systems.

Target Simulator: In the absence of real data, a Target Simulator is used to simulate the existence and activities of numerous targets that might be anticipated under real conditions. For each target, the target simulator maintains a simple schedule of target movements. Simulated target positions and trajectories are computed by the target simulator and then provided for use by the sensor emulation, or for display by the user interface.

Remote Sensing: The Remote Sensing Systems are a heterogeneous collection of autonomous sensing systems. The primary function of each system is to provide target-related data from which the overall target situation can be assessed. For assessment of the ground targets situation, the majority of systems considered for the experiment operate from an airborne platform and employ optical, infrared or radar sensing technologies.

The sensor components of the C² Internet Experiment emulate the selected sensing systems described above. The sensor system emulator obtains target position data from the Target Simulator and introduces distortions into the data in a manner which is consistent with the effects of weather and other environmental conditions on the characteristics of the overall sensing system. An application console can remotely monitor and control the sensing systems resources and display the resulting target data.

Mission Data Management: Some of the heterogeneous data produced by the various Remote Sensing System contains information vital to the success of this and similar C² applications for functions such as target identification. Therefore, it is important that this data be made available in a timely and reliable manner. This is accomplished by the Mission Data Management System, which services requests

to store and retrieve such data. The requests for data storage originate from the sensor systems; the requests for data retrieval originate primarily from the fusion processing component. Mission data is also accessible for examination and evaluation by human operators on a console that supports graphical display of target data presented against a cartographic background.

Fusion Processing: To support ground force situation evaluation, fusion processing is performed on the aggregate target information collected by the Mission Data Management function. The fusion procedure, which associates or combines the target data from the various sensor systems, is referred to as the target data correlation process. The correlation procedure is based on a set of rules which are used to infer the coincidence of target related data from two or more sensor systems with overlapping coverage, in order to discriminate false alarms from true targets. This correlation process requires the examination of primarily the uncorrelated target data; however it may require the use of other sources of information (for example, cartographic data). The process is sufficiently complex to warrant specialized computer resources, such as symbolic processors. Also, in an actual system, there will be cases requiring operator review and intervention. Hence, a console is required which provides access to the uncorrelated and correlated target data, as well as to the rules used to infer the correlated results.

Target Report Data Management: The Target Report Data Management System is similar to the Mission Data Management System: its primary function is to provide a repository for target-related data. However, not only does it maintain data generated by the fusion process, it also stores data associated with particular targets that have been generated by other elements within the experiment (for example, the particular priority or target/weapons pairing associated with a target).

Target Prioritization: Following the detection, identification, and classification of targets, prioritization of targets occurs on the basis of their importance. This will permit priority-based resource allocation during the mission planning stage. The Target Prioritization System relies on cartographic

data, as well as anticipated target track to determine the strategic importance of a particular target.

Target/Weapons Pairing: The function of the Target/Weapons Pairing System is to determine an effective weapon system for deployment against a chosen target. The Target/Weapons pairing is based on information retained about weapon systems capabilities and availability, in conjunction with relevant target information, such as the anticipated track, and meteorological and cartographic information. The C² Internet Experiment will provide an operator with the resources to display, manage, and manipulate the relevant information.

Resource and Logistics Data Management: A fairly substantial amount of information relevant to the Target/Weapons Pairing and Mission Tasking functions must be retained: this task is the function of the Resources and Logistic Data Management System. This system will maintain capability and availability data for such resources as personnel, equipment, and facilities. This data management system must maintain data such as equipment capability, that is relatively static, as well as maintaining availability data that changes frequently as resources are reserved for use, expended and replenished. Of the various data management systems included in the application architecture, this one will be the most similar to a classical database.

Mission Tasking: The Mission Tasking System is responsible for the preparation of Air Tasking Orders (ATOs): assigning a number of resources to a number of sorties, subject to a group of constraints (logistic, meteorologic, etc.). Each resultant ATO includes such information as the number, type, and location of the assigned aircraft, the location of the target, and a textual description of the major mission objectives. After the ATO is developed, it is disseminated to the appropriate units for mission execution. In order to develop the Air Tasking Order, the Mission Tasking System will make extensive use of the information stored in the Resource and Logistics Data Management System, and lesser use of the information stored in the Target Reportand Meteorological Data Management Systems. The resulting

ATO will be retained in the Resource and Logistics Data Management System. Information associating a particular mission with a given target will be retained in the Target Report Data Management System as well.

Support Functions: Supporting the processing functions planned for the experiment requires the storage and maintenance of a variety of data to support functions performed in both clusters. The Cartographic Data Management System includes such data as ground height, vegetation and surface type. Since the total volume of cartographic data is quite large, it would be inefficient to store it in one cluster and repeatedly transmit it to the other cluster when requested. Therefore, the cartographic support will be replicated and exist in both clusters. This use of replicated cartographic data will offer a means to include Cronus replicated object management in an environment where updates are performed infrequently and with few changes compared with the total amount of data stored.

The Meteorological Data Management System provides and retains weather data, such as ceiling, visibility, wind speed, and temperature, collected from various reporting sites. In addition, it will maintain forecasts that have been made for particular geographic areas. This information will be available for use by operators performing the various function of the planned experiment. This system also provides simulated weather information used by other systems to emulate the effects of weather on the environment.

Simulation Clock: The simulated time of day coordinates the progress of the experiment simulation. The activities of targets, sensors and other simulated elements are described by schedules. The experiment operator can start, stop and vary the rate of the time of day used by the experiment components, thereby controlling the progress of the simulation.

2.3. Monitoring and Control

In order to fully exercise the experimental configuration in a straightforward manner, we anticipate the need for an Experiment Monitoring and Control System. The Experiment Monitoring and Control System will be used to instrument and control the experiment. It will be used to perform such functions as:

- the initialization and configuration of the experiment;
- the control of scenarios during the course of the experiment;
- the generation of experiment failure modes;
- the introduction of artificial load;
- and the acquisition and processing of application performance data.

Similarly, an Application Monitoring and Control System is envisioned. The Application Monitoring and Control System will provide support for the role of an operator of the C² system. It will provide information on the status of the various components of the distributed application system. From an application monitoring station, the operator can aid recovery operations and control in real-time the allocation of application resources.

2.4. The User View

Many of the subsystems comprising the C² Internet Experiment need one or more human operators to perform critical decision-making tasks. For example, the intervention of an operator familiar with battle plans would undoubtedly be necessary in the formulation of target priorities. Alternatively, the skill of a trained expert may be necessary to assist in the differentiation of actual and spurious sensor system data. In order to perform these functions, an operator must have available all information relevant to his specific decision-making task. The need therefore arises for operators to have an

appropriate interface into the facilities provided by the distributed C^2 system, and interoperator communication facilities for the timely transfer of urgent messages.

The type of interaction and the appropriate access control needed to support the variety of "users" will vary. In some cases, the operator may merely serve to monitor the system, determining if automated processing is being performed correctly. In other cases, the operator may act as a sort of valve on the flow of information, filtering it as it passes from one subsystem to another. In still other cases, the station may be a data entry point. For example, an operator controlling the Resource and Logistics Data Management System may inform the system of the arrival of new supplies.

The sophistication of the user interface provided will undoubtedly vary as well. An operator controlling target correlation functions may need a topographic display overlaid with the positions of potential targets, so that obviously spurious ones could be eliminated using the knowledge of the operator. *On the other hand*, an operator controlling the Resource and Logistics Data Management System may not need such a sophisticated interface; a pre-formatted text-only display would probably be entirely adequate.

User interfaces are often quite costly. For the C^2 Internet Experiment, the user interface will not be an area of emphasis. We will provide, within the context of the experiment, functional user interfaces for a number of the application subsystems. However, our ability to provide application fidelity in this area will be limited by the level of effort for the project.

3. EVALUATION OF THE DISTRIBUTED SYSTEM

One of the major goals of the C^2 Internet Experiment is to demonstrate the use of distributed systems technology in C^2 applications. The application framework described in the preceding section serves as the basis for exercising the facilities of the Cronus distributed operating system. By choosing a large application context consisting of many different interconnected parts at the outset, the ability to perform evaluation of many specific Cronus facilities is enhanced. The level of detail to which various parts of the experiment are implemented or stubbed will be determined by the usefulness of those parts in testing the distributed system.

3.1. Evaluation Criteria

Conventional operating systems are typically evaluated by techniques that include benchmarks, direct instrumentation, simulation, and analytic models, to produce performance measures such as response time, throughput, capacity, and utilization. Although distributed system evaluation could measure the performance of analogous statistics, it is not evident that such techniques are immediately relevant. In evaluating a distributed system, particularly the effectiveness of a distributed system in supporting the development and use of distributed applications, it is more sensible to place importance on identifying how well the system achieves the attributes which initially prompted the development of the distributed system. The remainder of this section will discuss the issues which motivate consideration of a distributed architecture in the context of the C^2 Internet Experiment, the context which will be used to evaluate the characteristics of the distributed system. Also discussed are i) the manner in which the Cronus software tools and concepts are used to achieve the attributes of a distributed architecture relevant to the specific applications, and ii) the manner in which the Cronus software development tools

are used to build the specific application.

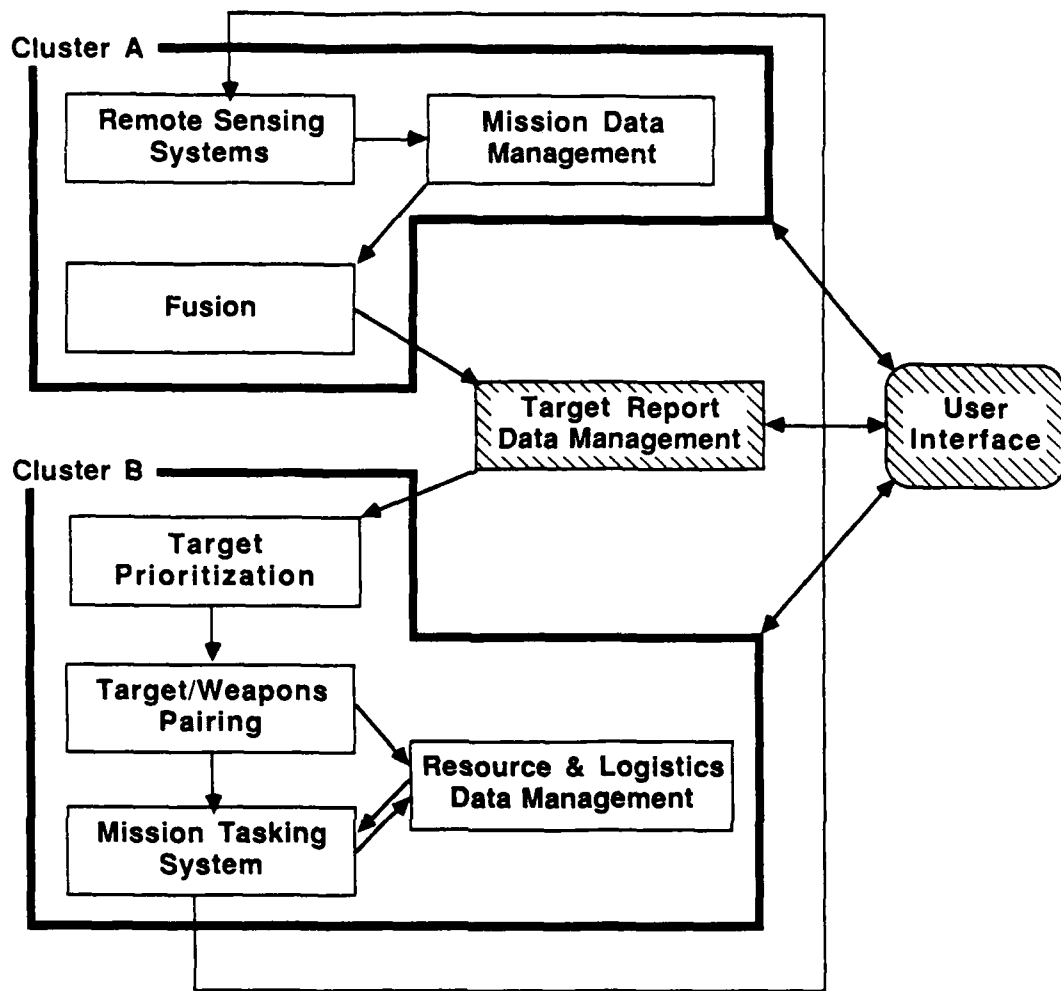
3.1.1. The Cronus System Environment

The Cronus distributed operating system is intended to promote resource sharing among interconnected computer systems, and manage the collection of resources which are shared. Its main purpose is to provide a coherent and integrated system to support the development and use of applications requiring the complexity of a distributed architecture.

Cronus exists in an environment of heterogeneous processors (or *hosts*) interconnected by computer communication networks. A Cronus *cluster* consists of some number of logically (and probably physically) grouped and interconnected hosts. Intracluster communication typically is characterized by relatively high available bandwidth and relatively low delay. Clusters may be interconnected; intercluster communication usually exhibits lower available bandwidth and higher delay than intracluster communication.

For the C² Internet Experiment, we are implementing the various functional units of the application described in Section 2 on a number of hosts. The experiment will be organized into a two-cluster configuration (see Figure 3). The current partitioning includes sensing functions and preliminary target situation analysis in one cluster, and battle planning and management functions in the other. This clustering is one of any number of plausible configurations and is motivated by the idea that the functions within a cluster will generally be functionally related.

In the development of Cronus, we have taken the view that processing elements are heterogeneous. This assumption rests on the premise of functional specialization. That is, certain processors and systems will be more capable of certain functions than others. It is, however, highly desirable for these



Support Functions

Cartographic Data Management

Meteorological Data Management

System/Component Monitoring and Control

Shaded functions are replicated in both clusters

Potential Two Cluster Configuration of C² Application
Figure 3

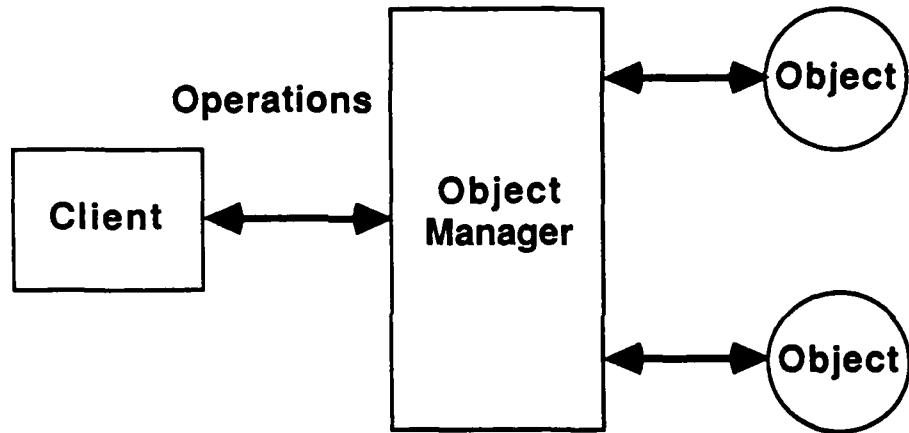
functionally specialized units to interwork so that they can cooperatively perform some larger task. Often each of the processors has its own native or *constituent operating system* (COS; for example, DEC's VMS). Instead of a tight coupling to the COS, Cronus is typically made minimally reliant on COS services. As a result, Cronus is largely independent of both the processor hardware and the constituent operating system upon which it is implemented. This increases the portability of lower-level Cronus routines, while minimizing changes to the COS due to the installation of Cronus. The system hardware and software base supporting Cronus can be replaced in whole or in part without global impact on the application. However, we also retain for efficiency the use of existing COS mechanisms for strictly "local" processing while adding Cronus mechanisms for processing which is intended to be "global" in nature. Applications which directly utilize COS mechanisms trade-off host dependency for efficiency.

The experiment will make use of the heterogeneous environment of existing Cronus host types and constituent operating systems. The Cronus processor and constituent operating system base currently consists of DEC VAX systems running VMS and 4.2BSD UNIX, Sun workstations running Sun UNIX, BBNCC C/70 processors running BBN UNIX, and Motorola 68000-based Generic Computing Elements (GCE) running a real-time operating system developed at BBN. Where appropriate, the application subsystems will take advantage of the functional specialization offered by underlying resources. For example, the Fusion Processing System and will be implemented on a machine (or group of machines) capable of meeting the required (potentially large) processing requirements, each remote sensing system may be simulated on a separate GCE, and the SUN workstations may be primarily used for monitoring and control purposes.

3.1.2. The Cronus Model

In order to provide the desired integrated view of diverse resources, a system model has been developed. The development of the model has been motivated by a number of factors including: the need for a coherent system decomposition, the need for the customization of various components, and the need to provide easy to use "off-the-shelf" mechanisms for a wide range of potential applications. A model which meets these needs is the object model on which Cronus is based. This Cronus model allows resources to be viewed abstractly, shielding consumers from the actual implementation details that determine how the services are provided. Using the Cronus object model, the distributed system is thought of as consisting of a collection of typed and uniquely-identified *objects*. Associated with the objects of a particular type are a number of permissible operations. The operations on an object type are implemented within an *object manager*. An object manager is responsible for all objects of a particular type on a given host; one or more managers collectively manage a given object type within the Cronus environment. In order to perform tasks, clients (or other managers) invoke operations on objects through the object managers (see Figure 4). The Cronus system provides mechanisms for clients to manipulate objects without concern for their location in the network or for the implementation of the operations within the manager. This allows the evolution of the mechanisms implementing various operations without necessitating changes to their external view. Resources within the distributed system are viewed as objects, for example files, processes, and access control groups. The benefit of the object model is that it is, by nature, easily extensible through the addition of new object types. Application resources can be cast as objects and make use of the same mechanisms used to support the system objects.

For the experiment, we will define appropriate application object types. For example, a "target" object type can be viewed as the collection of features, characteristics, parameters, etc. associated with a particular target. A "sensor" object type can be viewed as a process characterized by the qualities



The Cronus Object Model
Figure 4

associated with a given sensing system. The ability of Cronus to accommodate evolutionary development within managers will be used throughout the experiment. We anticipate initially stubbing many of the application functions. As the experimental development progresses, the extent of this stubbing will decrease as more complete alternatives are substituted.

In the Cronus model, objects may be classified as *primal*, *replicated* or *migratory*. Primal objects are always located on the host on which they were created. Migratory objects may move from host to host as situations change largely to support resource management objectives. For example, when the storage resources on a given host are exhausted, a migratory file could be moved to a host with more available space. Replicated objects are duplicated on one or more hosts and are used to enhance survivability. By having more than one active copy of an object, failures can be mitigated by using the redundant copy (or copies) of the object. Managers of replicated objects cooperate to maintain the desired degree of consistency between the copies of the objects. The location of objects is, in general, transparent: performing "local" and "remote" operations are functionally equivalent. However, clients have the option of exerting control over object location in order to perform application-oriented optimization. The facilities for transparently locating and operating on both primal and non-primal objects within the operating environment are supported by the Cronus kernel. In the C² Internet Experiment, we plan to make extensive use of the primal, replicated, and migratory attributes of objects as appropriate.

Survivability is a critical issue in C² systems. The Cronus object model provides facilities to make applications survivable. Functional redundancy is offered by simultaneously providing more than one object manager (on different hosts) for a given object type; data redundancy is provided by using replicated objects. This redundancy of both function and data can be used to minimize the effect of *single (or multiple) failures*. The distributed system is susceptible to various types of failures for which some degree of survivability of an overall function is feasible. These include:

- Host and manager failures
 - Part of a replicated data resource fails
 - Part of a replicated function resource fails
- Communication failures
 - Degraded intercluster communication
 - High delay
 - Low bandwidth
 - No intercluster communication
 - Partitioned local area network
 - Degraded local area network
 - Network jammed

In order to exercise the Cronus facilities for survivable data, two environments must be considered. Within a cluster, data replication is used to insure that critical data generally will remain accessible when various failure modes occur. In the C² Internet Experiment, data replication facilities within a cluster will be used in a number of places. As one example, replicated data will be used to insure the integrity of the Meteorological Data Management System. Some degree of reliability can also be achieved as part of the application itself. For example, uncorrelated target data (target object types) from two sensors with overlapping coverage can be managed so as to reside transparently on separate devices. If one device fails, some target data relevant to the region of coverage is accessible.

Some data is used in more than one application cluster. In order to insure its survivability, it is necessary to replicate this data in the intercluster environment. Replicating data in more than one cluster will guard primarily against intercluster communication failures, and secondarily against host failures. In the C² Internet Experiment, our current plans are to demonstrate the use of survivable data in the intercluster environment by replicating the Cartographic Data Management System, and the Mission and Target Report Data Management Systems in each cluster. The replication of such data also relaxes the

demand for intercluster communication that might otherwise be necessary.

The Fusion Processing System is illustrative of the need for survivable processing. Survivability of this function will be enhanced by distributing its processing to a number of hosts. Then, should one of these hosts fail, the function can still be performed (although potentially with reduced capacity). Within the Fusion Processing System, the overall function will be performed using different machines to process targets from different geographic regions. This partitioning of the task into similar but smaller subtasks is also relevant to distribution of processing in a multiprocessor architecture.

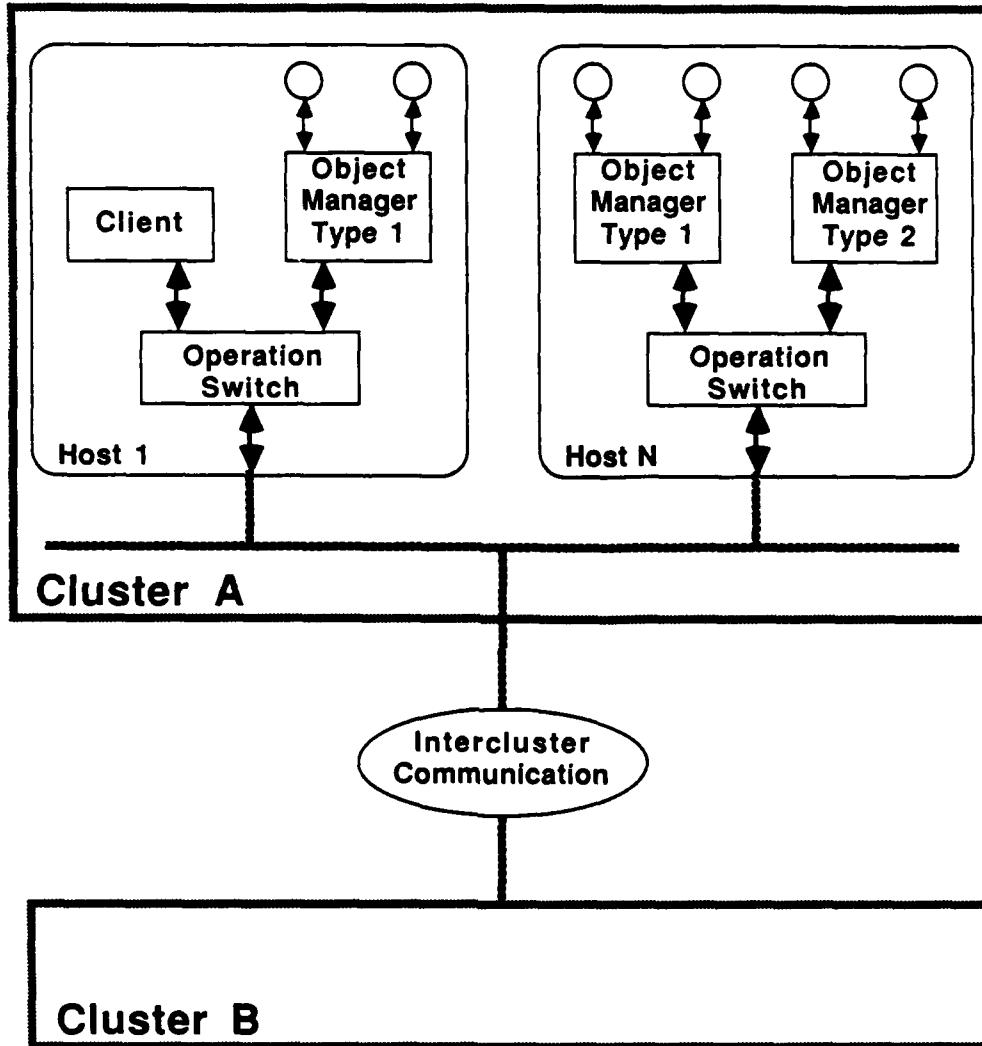
In a distributed system, the need arises for the formulation of global resource management policies, and for the mechanisms to implement these policies. These strategies are necessary in order to effectively and conveniently control redundant resources. The Cronus resource management model is based on providing a set of mechanisms capable of supporting a variety of policies. These mechanisms are based on making resource allocation as transparent as possible to the client and tailorable to the resource being managed; they include: the ability of managers to redirect requests to other managers of the same type (i.e., peer managers), the ability of managers to accumulate information about the status of their peers, and the ability of users or applications to indicate preferred hosts.

The redundant resources provided for reasons of survivability and to handle increased capacity in the C² Internet Experiment will need to be managed. Such management will be provided within the context of the Cronus resource management model eluded to above. Should one machine providing a replicated function (or data) fail, the global resource management offered by the distributed system will be used to perform a real-time workload redistribution and prioritization. Global resource management will also be used to ensure effective use of redundantly available storage resources to avoid creating system bottlenecks. These mechanisms will also be used to demonstrate scalability of an application function in making effective use of additional resources. At minimum, these system attributes will be

utilized in the target data management subsystem.

In virtually all computer systems, access control mechanisms are necessary to prevent the unauthorized use of services and data, and to preserve system and component integrity; Cronus is no exception. In fact, robust access control techniques are particularly crucial in Cronus, since by design Cronus users have access to a potentially large number of hosts and facilities. Therefore, all Cronus operations are subject to access control restrictions. The access control mechanisms provided by Cronus will be used in the C² Internet Experiment to prevent unauthorized access to data and functions. For example, an operator of the C² system performing updates to the Resources and Logistics Data Management System should not be able to access or perform operations on uncorrelated target data since it is not relevant to his job function. Similarly, the ability to redirect remote sensing units will be limited to application operators designated to control this task.

Object managers communicate with both clients and other managers using standardized message formats and representations. These *canonical* representations allow convenient communication between heterogeneous hosts with differing internal host-specific representations. Clients and object managers communicate using an *interprocess communication* (IPC) mechanism contained within the Cronus kernel (see Figure 5) [BBN5884]. The Cronus kernel provides the object-based message passing facility required for the C² Internet Experiment. It implements such functions as manager-to-manager message routing, transparent location of migratory and replicated objects, and access control support. The Cronus IPC, in turn, relies on standard, off-the-shelf datagram and virtual circuit services of the underlying communication network to provide host-to-host communication. The Cronus kernel message passing facility will also be considered for communicating time urgent messages among system operators supported at various sites.



The Cronus Kernel Interprocess Communication Concept
Figure 5

Presently, Cronus uses the TCP family of protocols (TCP, UDP, IP) over Ethernet and the ARPANET to provide the intra- and intercluster communication facilities respectively. The communication system design, like the rest of Cronus, is modular, providing the ability to easily substitute different technologies subject to a small number of constraints. For the experiment, the IPC mechanism and underlying communication services will be used in their current form.

3.1.3. Application Programming Support

Yet another aspect of distributed system evaluation is the ease with which distributed applications can be built. Within Cronus, there is little difference between "application" and "system" services. Thus, building new Cronus applications is roughly equivalent to building new managers which provide additional services. Handling all of the details associated with building a component of a distributed system is a potentially time-consuming task. Providing Cronus libraries to perform commonly used functions such as conversion from internal to canonical representation relieves some of the burden. However, even taking advantage of these libraries, application programming is very detailed and tedious. In order to reduce the effort involved with using the facilities provided by the Cronus kernel and libraries directly, high-level tools are provided for the development of distributed applications [Cronus, DCS].

The initial approach to simplifying Cronus application development has been to develop high-level tools that, starting with a specification of the object types, operations, and access control requirements of an object manager, provide much of the code that would otherwise need to be written manually to utilize many of the low-level Cronus mechanisms. In building a distributed application then, the programmer must provide this specification, and provide subroutines that implement the operations specified.

The software generated by the tools, in turn, invisibly provides message parsing and validation, conversion of arguments from canonical to host-specific format, and a subroutine interface to operation invocations for use by clients. The run-time support additionally includes multitasking (multiplexing operations within managers), access control verification, object management, and a number of other facilities.

The high-level Cronus tools will be used in the construction of the C^2 Internet Experiment application subsystems in order to evaluate the utility of Cronus application development tools.

Table 1 summarizes how various requirements are accommodated by the distributed system and its associated tools.

Requirement	Cronus support mechanism(s)
Heterogeneous environment; Functional specialization	Standards (e.g., canonical message formats); transparency of distributed operations; availability of COS facilities.
Survivability of data	Replicated objects.
Survivability of function	Redundant managers for the same object type on different hosts.
Global resource management	Uniform mechanisms (e.g., migratory objects, managers can redirect requests) support customizable policies.
Evolvability; Substitutability based on standards.	Managers hide implementation details within high-level operations, implementation
Integrity of data and services	Access control restrictions on all operations.
Extensibility	Introduction of new managers and types.
Simplified application programming in a distributed environment	Software development tools; ability to use "off-the-shelf" software with little modification.

Distributed System Requirements and Support
Table 1

3.2. Evaluation Methodology

Beyond the layout, design, and construction of a demonstration C² system, we are also interested in evaluating system performance using the developed application. The command and control system described will be used as an evaluation environment for testing Cronus facilities, the overall goal being the evaluation of Cronus performance. The performance of various application elements is a secondary consideration except insofar as they are influenced by the performance of the underlying system architecture. Therefore, an efficient application design, per se, is not critical at this time. Rather it is the performance of the distributed system facilities that is being evaluated and the subsequent impact on the distributed system performance has the interoperability of the application subsystems.

Performance may be judged on two types of criteria: qualitative and quantitative. In order to perform meaningful quantitative measurements, it is necessary to tightly and repeatably control experimental conditions. Since many of the resources that constitute Cronus are not easily controllable, and realistic workload and operating conditions are not available, quantitative tests could prove to be extremely difficult. Thus, we will initially concern ourselves with the qualitative evaluation of the distributed system: whether or not the system works, and subjectively whether it works reasonably well under limited load.

Some simple quantitative performance measurements will also be made. A reasonable performance evaluation technique is to isolate different parts of the application and Cronus, and measure their performance independently to determine where performance bottlenecks and saturated resources exist within the system under various workloads. One approach to isolating these parts is to identify and instrument across significant logical boundaries during the execution of Cronus operations. For example, a simple Cronus request can be roughly broken down as:

client → Cronus kernel → tcp → network → tcp → Cronus kernel → manager

Determining the contribution of each part to overall system performance measurements allows an analyst to focus on the points of maximum leverage. High-level interactions will be benchmarked using a variety of criteria. Some examples are:

- response time of system to application
- related time urgent requests;
- number of operations per module;
- elapsed time per module;
- cpu time per module.

Of course these measurements would still have a certain amount of variability due to such factors as host type, configuration and load, the potential for operation redirection, and the status of the underlying communication facility. Nonetheless they could form the basis for a more exhaustive system evaluation.

4. EXTENDING THE EXPERIMENT

Some aspects of the C^2 Internet Experiment application domain cannot be adequately carried out within the current Cronus system; a number of system extensions are warranted. We will be considering extensions in a variety of areas including software, hardware, and communications. It is clear that many command and control systems require the storage of, access to, and complex manipulation of a variety of data. The Resources and Logistics Data Management System of the experiment is one example. Toward the efficient management of that data, we are considering the integration of a commercial database management system into Cronus. We are also considering integration of additional processor and constituent operating system types, including multiprocessors and symbolic processors. Integration of a multiprocessor would offer increased access to a wealth of processing power at low cost. This would facilitate the implementation of partitionable computationally intensive application functions, such as target detection based on image like sensor data. Symbolic processors are currently being used in the development of many sophisticated application systems. One major application is the implementation of expert systems and other AI-related concepts. In the context of the present experiment this capability is especially relevant to automating target data fusion and other decision making related processes in the C^2 cycle. Hence the introduction of machines with such new architectures would extend the current potential application domain.

Present plans call for the interconnection of the two experimental Cronus clusters using the ARPANET. However, the existence of other communication networks offer the opportunity to experiment with networks exhibiting a variety of potentially more desirable characteristics. We are investigating the use of the DARPA Wideband network as such an alternative intercluster network: it offers a much higher available bandwidth than the ARPANET.

Our plans for the future are oriented toward both the progression of the C² Internet Experiment and the refinement of Cronus. In the long run, we expect future application development and future Cronus development to have a symbiotic relationship. As distributed applications evolve, undoubtedly additional requirements for the underlying distributed system will surface. These needs will be met by evolving Cronus to meet them. Furthermore, as the development of Cronus proceeds, it is expected that further test and evaluation of the distributed system will be necessary; our approach here is to evolve the prototype applications in areas that exercise the new functions.

REFERENCES

- [ADDCOMPE] M. Frankel, C. Graff, L. Dworkin, T. Klein, R. desJardins, "An Overview of the Army/DARPA Distributed Communications and Processing Experiment," IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 2, March 1986.
- [BBN5879] R. Schantz, R. Thomas, "Cronus, A Distributed Operating System: Functional Definition and System Concept," BBN Report 5879, January 1985.
- [BBN5884] "Cronus, A Distributed Operating System: Revised System/Subsystem Specification," BBN Report 5884, January 1985.
- [BBN6248] J. Berets et. al., "C² Internet Experiment: System/Subsystem Specification", BBN Report 6248, May 1986.
- [BBN6251] J. Berets et. al., "C² Internet Experiment: Final Technical Report", BBN Report 6251, May 1986.
- [Cushman] J. Cushman, "The User's Viewpoint," Signal, v. 37, n. 12, p. 47, August 1983.
- [Cronus] R. Sands, K. Schroder, eds., "Cronus User's Manual," BBN Report 6180, February 1986.
- [DCS] R. Gurwitz, M. Dean, R. Schantz, "Programming Support in the Cronus Distributed Operating System," Sixth International Conference on Distributed Computing Systems, May 1986.
- [GACC] J. Dussault, "Ground Attack Control Center," RADC briefing, November 15, 1984.
- [TRW] M. Mariani, R. Turn, B. Johnson, "Architectural Study for Tactical C³ Construct," RADC Technical Report 80-304, September 1980.



*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 (C³I) 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 C³I 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.