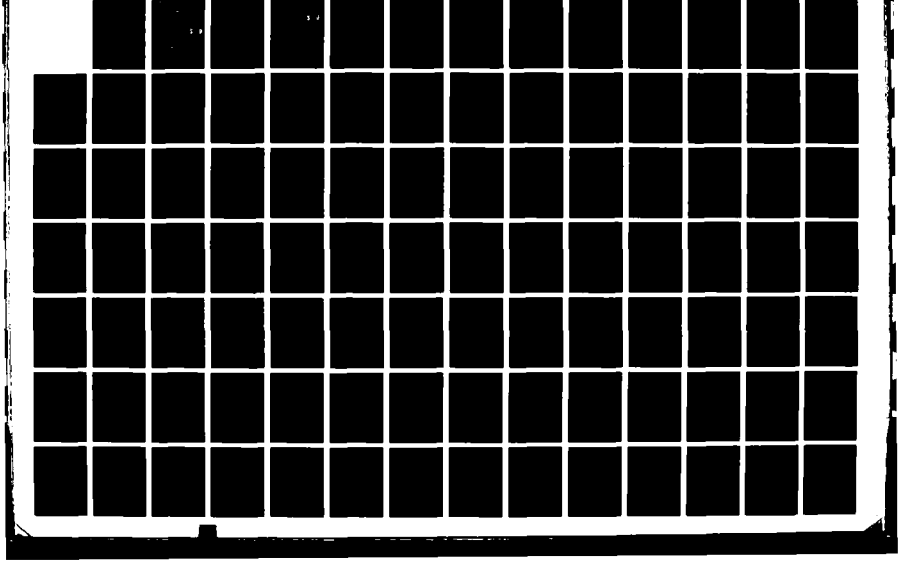
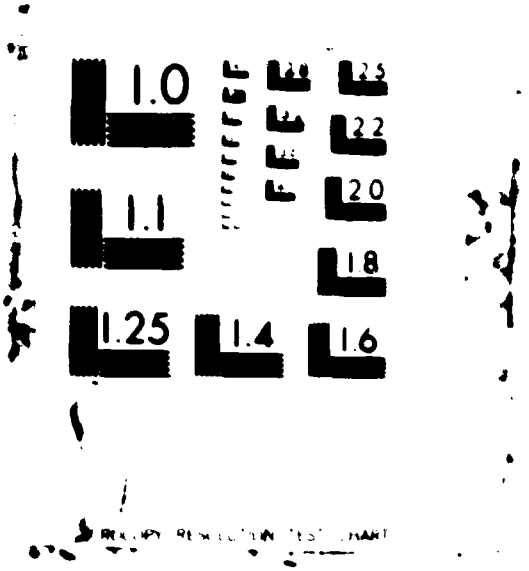
AD-A179 577 VIRTUAL COMMUNICATIONS TO ASD'S (AERONAUTICAL SYSTEMS DIVISION'S) CRAY CO..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. T W TASSEFF 1/2

UNCLASSIFIED DEC 86 AFIT/GCS/ENG/86D-10 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
2.0
2.2
2.5

(1)

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

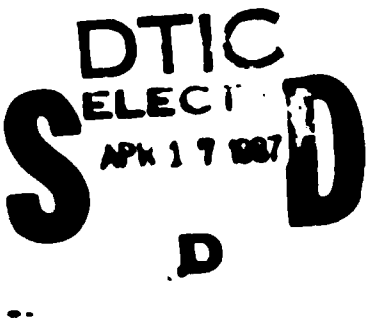
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

List of Tables

(1)

AFIT/GCS/ENG/86D-10



VIRTUAL COMMUNICATIONS TO ASD'S CRAY COMPUTER VIA AFIT DATA COMMUNICATIONS RESOURCES

THESIS

Todd W. Tasseff
Captain, USAF

AFIT/GCS/ENG/86D-10

AFIT/GCS/ENG/86D-10

VIRTUAL COMMUNICATIONS TO ASD'S CRAY COMPUTER VIA AFIT DATA COMMUNICATIONS RESOURCES

THESIS

Presented to Faculty of the School of Engineering

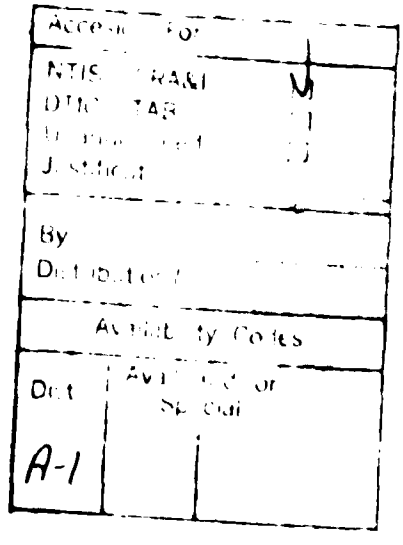
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science



Todd W. Tasseff

Captain, USAF

December 1986

Approved for public release; distribution unlimited

Preface

The task of this thesis project was to study, create, and evaluate a virtual communications capability to ASD Computer Center's Cray supercomputer from the AFIT School of Engineering using existing AFIT computer and data communications resources. The system will be used to support research and coursework for AFIT faculty and students.

The challenge of working with five separate computer systems, three operating systems, and two different communications links all at the same time grew more intense as I struggled to keep the system I created organized and workable. But having done so, I learned a great deal about the importance of good systems programming.

I would like to offer my thanks to Hal Carter, my thesis advisor, for his help and encouragement throughout the course of this thesis project. I would like to thank Kirk Horton as I used his 1984 thesis as model for my own. I would also like to thank AFIT/SI, especially Joe Hamlin, Bob Filer, and Jim Ware, and ASD/SI, especially Ken Johnson, for their assistance and expertise during the system programming phase of the thesis. I would especially like to thank my wife, Patsy, for her support and patience during this thesis project. And, I would like to thank God for the strength He gave me throughout my thesis project and my entire time at AFIT (Phil. 4:13).

Table of Contents

List of Figures

Abstract

The AFIT-ASD Cray System, a virtual communications capability from the AFIT School of Engineering to ASD's Cray computer, was studied, created, and evaluated. The SADT design, detailed design, and System program code are included in the study. A user enters the System's user command sendcray which initiates a Cray job from a remote or central AFIT host computer, and transfers the user's input file to the central AFIT host computer. The input file is then sent first to the Cyber, which is a front-end to the Cray, and then to the Cray itself where the input is processed. Output from the Cray is sent to the Cyber and then back to the AFIT central computer. Finally, the output, with an accompanying job log file, is transferred to the user at the originating AFIT host computer.

The System uses a combination of 4.2 BSD UNIX, Cyber NOS, and Cray operating system commands, and makes use of an AFIT Ethernet network and a VAX-UNIX/Cyber RACP/modem link. The System operates over a series of UNIX C-shell programs, most of which are executed automatically. The System provides the user with simple, error-free, and automatic access to the Cray, with the potential of improved turnaround time for compute-intensive jobs at AFIT.

VIRTUAL COMMUNICATIONS TO ASD'S CRAY COMPUTER VIA AFIT DATA COMMUNICATIONS RESOURCES

I. Introduction

Background

In recent years, the Department of Electrical and Computer Engineering (ENG) at the Air Force Institute of Technology (AFIT) School of Engineering has become a center for Very Large Scale Integration (VLSI) circuit research within the Air Force and the VLSI research community (5). In order to conduct VLSI architecture research, AFIT/ENG performs logic design and circuit simulation, and checks integrated circuits and systems with integrated circuits (5).

The VLSI simulation and checking is computationally intense, and requires a relatively large amount of processing time. For instance, a typical circuit simulation using the SPICE software package uses up to 24 hours of CPU (central processing unit) time on one of AFIT's DEC VAX-11/785 computers (5). Such a long turnaround time makes the VAX unwieldy for use in the timely development of VLSI circuit chips.

As a partial solution to the above problem, AFIT/ENG has recently acquired an ELXSI 6400 multiprocessor computer to be used for VLSI research. The ELXSI can execute 7 MIPS (millions of instructions per second) versus the VAX's 1.5 MIPS (5), and is therefore about 5 times as fast. The previous SPICE circuit simulation job would run for a minimum of five hours of CPU time

on the ELXSI versus 24 hours on the VAX. Although the CPU time is a large part of the overall job turnaround time, another time factor is that the job must share CPU time with other ELXSI user jobs. Thus, a job on the ELXSI could consume most of an average workday, still unacceptably slow. A much faster computer is still needed to speed up the development of complex VLSI designs.

One nearby facility with more advanced computer support than AFIT is the Aeronautical Systems Division's (ASD) Computer Center, which has recently installed a Cray XMP-12 supercomputer. The Cray can execute at over 200 MIPS (5) which is about 30 times as fast as the ELXSI. A 24-hour VAX SPICE job would take a minimum of 11 minutes of CPU time on the Cray. For this reason, the Cray would be the more desirable compute. for AFIT's large VLSI design simulations. If AFIT could gain easy access to the Cray, the Cray would remedy AFIT's problem of long turnaround times for simulating large VLSI designs.

Problem

However, the main problem with gaining access to the Cray for AFIT VLSI research is how to do so in a simple, error-free manner. At present there is no easy way to run VLSI simulations on the Cray. One alternative is to make a tape with both a copy of the SPICE software package and a VLSI design data file on it, hand-carry the tape to ASD Computer Center, and load it on the Cray. But this procedure would have to be repeated for every SPICE job. Any errors created in the data due to the loading of the tape at AFIT and its subsequent unloading on ASD's Cray may go undetected. This alternative is also time-consuming. Besides creating the tape, transporting it to the Cray,

and loading it on the Cray, the total turnaround time would have to include running the SPICE program with the input, loading the output back on to tape, and transporting the tape back to AFIT. In addition, AFIT's tape format might be incompatible with ASD's tape drives.

Fortunately, a data communications link currently exists between an AFIT DEC VAX-11/785 computer and ASD's CDC Cyber 750 (a dual-system) computer (1), which is in turn connected to the Cray (15) (see Figure 1-1). But this link only services users on the VAX, meaning that any VLSI designs developed on AFIT/ENG's ELXSI would have to be transported to the VAX, sent to the CDC Cyber, then transmitted to, and run on, the Cray, with output returning to the Cyber. To get output from the Cray back to the ELXSI, the user would have to transport the output from the Cyber to the VAX and then back to the ELXSI. Most of these steps require some user intervention, and the overall process is still time-consuming and cumbersome for the user.

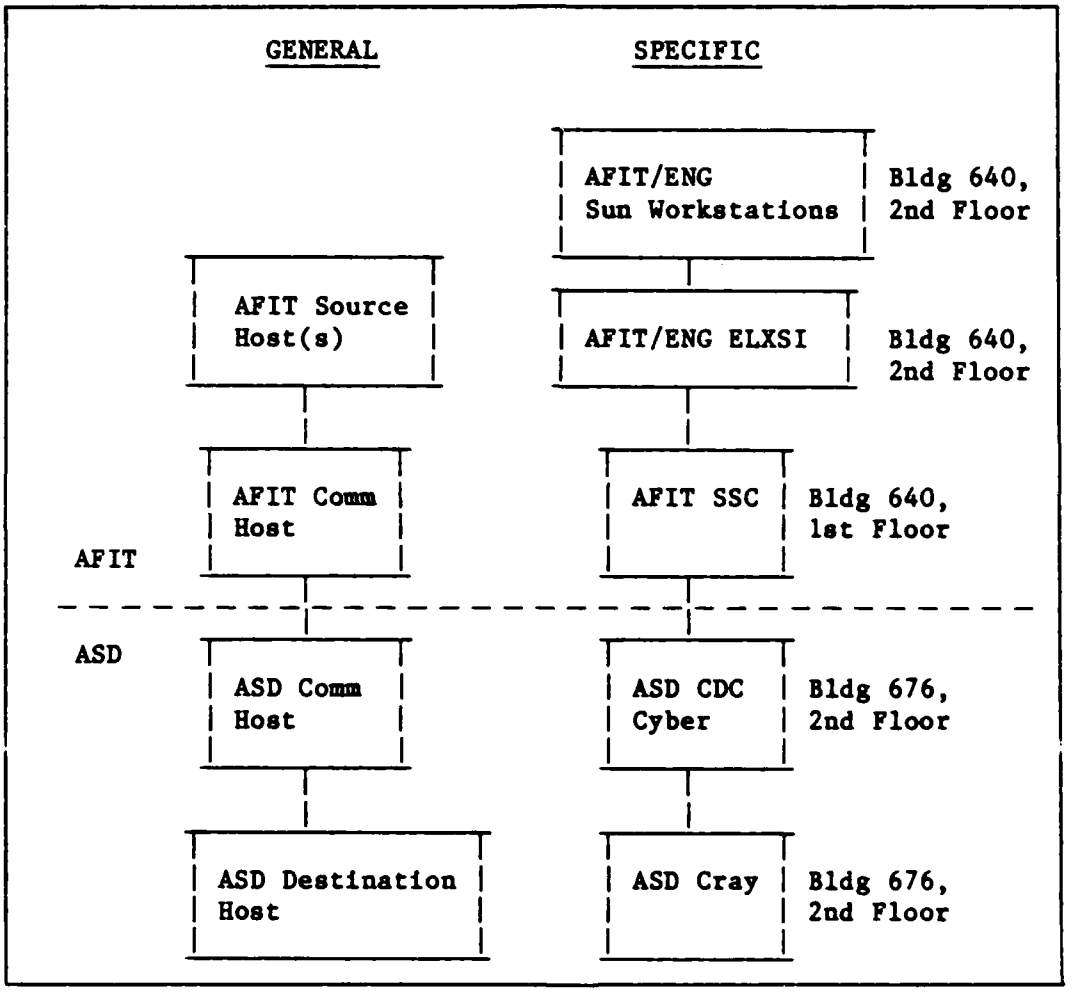


Figure 1-1 Overall System Model

In sum, any attempt to make use of ASD's Cray supercomputer for AFIT VLSI research would be extremely cumbersome and prone to error. The process requires that the AFIT user know something about each of the computer systems involved. The intermediate steps are not all automatic and require some user intervention. And the steps require a great amount of time (i.e., hours), which negates the advantage of using such a fast computer as the Cray.

Scope

The scope of this project is to choose, implement, and analyze the best available method for allowing an AFIT host computer to access a faster non-AFIT host computer. Specifically, such a data communication facility should be a flexible remote job entry (RJE) system capable of sending AFIT/EN programs (in particular, VLSI jobs) from an AFIT host computer to ASD Computer Center's Cray computer. Once at the Cray, the jobs would be run and the results sent back to a specified AFIT host computer. The specific AFIT host computers to be used, as test cases, would be the AFIT SSC (UNIX-VAX 11/785), and AFIT/ENG's ELXSI computer and associated Sun computer workstations (see Figure 1-1). The method chosen would then provide "virtual' communications to ASD's Cray. In other words, the system would provide access to the Cray even though there is no direct link from an AFIT computer to the Cray.

Summary of Current Knowledge

The ISO OSI Reference Model. Most data communications systems, whether consisting of a single link between two similar computers or a network of many different computers, adhere to some kind of computer network architecture. The architecture describes the specific data communications functions that are

required and how the system should perform them. In an attempt to standardize network architecture, the International Standards Organization (ISO) has created the Open Systems Interconnection (OSI) reference model (16:15).

The OSI reference model (see Figure 1-2) contains a set of "protocols"

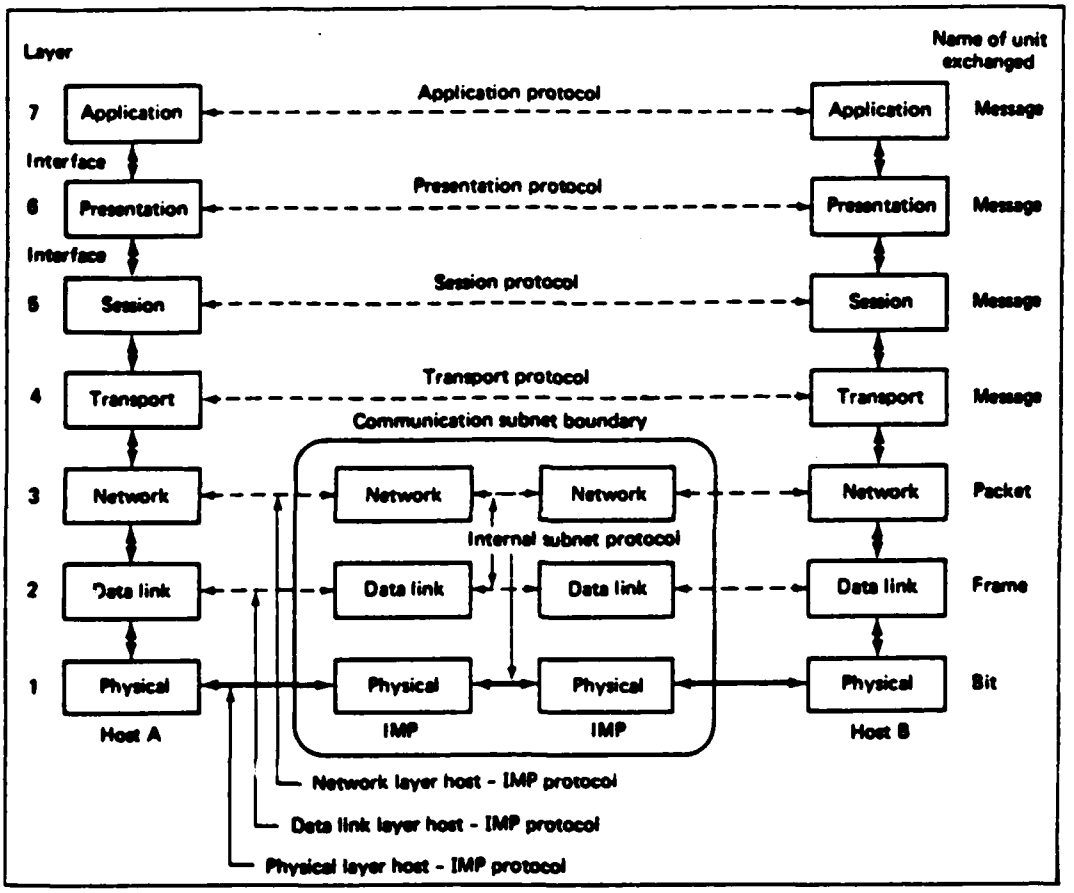


Figure 1-2. ISO OSI Network Architecture Reference Model (16:16)

which correspond to certain function "layers." Each protocol layer uses the layer below it to form a coordinated, cohesive data communications system (i.e., network). The basic layer is the physical layer, Layer 1. This layer transmits only "raw bits" of data over some transmission medium (wire, radio, etc.); it is concerned with the mechanical, electrical, and procedural

interfacing of data communications equipment (16:16).

The next highest layer is the data link layer, Layer 2; it is responsible for sending "frames" (predetermined number and order of data and control bits) sequentially to a destination node (data communication device). It then waits for acknowledgement ("transmission O.K.") frames to return from the destination node. This layer ensures error-free transmission over the transmission link (16:17).

The layer above the data link, the network layer, Layer 3, handles the node-to-computer interface and the routing of "packets" (frames with more control bits) within the "subnet" (16:17). The subnet comprises the physical, data link, and network layers. The network layer takes data messages from the source host computer, splits them up into packets, ensuring "that the packets get directed toward the destination" computer, and controls the traffic (and congestion) of packets in the subnet (16:18).

The next layer, Layer 4, the transport layer, makes sure that the data arrives at the destination computer correctly, making it a "true source-to-destination or end-to-end layer." This layer creates a unique network connection for each transport layer connection needed by the next highest layer (16:18). At the next highest layer, Layer 5, the session layer, the user (human user or application computer program) interacts with the computer network (16:19). At this level, the user establishes a connection with a program on a remote computer, providing the user knows the remote computer's network address (16:19). The layer above the session layer, the presentation layer, provides a "transformation service" to the user in the form of text compression, encryption, character-code conversion, file format conversion, etc. (16:20-21).

The uppermost layer, Layer 7, is the application layer. The messages sent at the application layer are determined by a human user or some application program at the source and destination host computers (16:21), depending on the individual requirement.

Remote Job Entry Protocols. According to Day (7:107,114), many of the data communications requirements of a "production-oriented data-processing" office are for terminal-to-computer access, or remote job entry (RJE) capability. Day points out that remote job entry protocols (RJEP) allow a human user or applications program "to execute a job on one of several computers" from a remote location, using a single RJE software package that runs on different computer systems. In relation to the ISO OSI reference model, RJEP corresponds with the protocol layers from session layer on up. An RJEP supports the same types of functions found in batch (non-interactive) computer systems: submitting jobs (programs to be run), determining job status, retrieving output, controlling the routing of jobs and output to the desired locations, for example (7:107).

Day defines the function of RJE in a network of computers by stating that RJE permits loadsharing (sharing computer resources) across more than one computer system (7:114). Day gives three reasons for RJE: lower cost or faster turnaround time; need to access special software that exists on a remote computer; or having too large a program to be run locally. Each RJE requirement, though, should be carefully researched before committing resources to develop a loadsharing operation (7:115). Day also mentions that if "a small transfer of data generated a large amount of computation," then remote job entry (to a more powerful host computer) can be feasible.

A Resource-Sharing Network Link. In his 1980 AFIT thesis, Design of a Resource-Sharing Network Link, Capt Thomas McLeod describes an example of RJEP (13). The project was an initial step in developing the ASD Automated Management System (AMS) resource-sharing (R-S) computer network. McLeod provided the design of one R-S link between two Texas Instruments model 990 computers (13:2,11). Some of the R-S functions that were required were: transferring sequential (data) and program files; transferring intersite messages; and executing remote programs (programs on remote computers) (13:14-15).

McLeod's thesis is similar in functionality to the type of system proposed in this thesis. Also, his thesis had the similar goals of developing an R-S link that was simple to operate and would use existing hardware and software (13:14).

AFIT SSC VAX to ASD Cyber RJE Link. Another more recent example of RJEP is the RJE link now operating between AFIT's SSC (UNIX-VAX 11/785) and ASD's CDC Cyber Computer. Currently, batch (non-interactive) jobs may be sent, error-free, from the AFIT SSC to ASD's Cyber via a hardware device called the COMBOARD which is installed on the AFIT SSC (1). The COMBOARD is then connected to ASD's Cyber computer via a standard synchronous modem (data communications transmission device) link. The RJEP that the COMBOARD and the Cyber use is the HASP (Houston Automatic Spooling Program) protocol.

A user on the AFIT SSC places the appropriate "control cards" (Cyber job-control commands) into the desired batch file, and uses an SSC command to send the batch file to the Cyber (1). Once at the Cyber, the batch job is run. Depending on the SSC commands used, the output from the job is either physically printed by the AFIT SSC, or it is stored in a file on the AFIT SSC

(1). The AFIT SSC user can also store and recall files from ASD's Cyber (i.e., file transfers) using the same RJE link (18).

Approach

The first step in solving the problem was to analyze the system requirements in order to identify what functions the system should perform and how well it should perform them. This analysis included the system specifications and justified, as necessary, those specifications. The second step in solving the problem was to translate the system requirements into an overall system design which incorporated the functions specified under the system requirements. The overall design was described using the Structured Analysis and Design Technique (SADT). The system functions were outlined using SADT charts and accompanying documentation.

The third step in solving the problem was to create a detailed design of the system from the overall design. This step involved translating the individual system functions into the proper set of programming commands. These programming commands formed program modules which became the basis for implementing the system.

In general, the proposed RJE system operates over the currently available AFIT data communications facility as described in previous sections, which in turn is connected to ASD's data communications facility. Specifically, the RJE system uses the existing RJE link between the AFIT SSC and ASD's CDC Cyber computer (1), which is in turn connected to the Cray (15).

Since the AFIT hosts and workstations involved operate under the UNIX operating system, the RJE system was implemented using UNIX C-shell commands to create the appropriate system programs (9). Computer-to-computer

communications among these UNIX machines required the use of standard TCP/IP data communications protocol commands over the existing Ethernet local computer network link (5).

The system required that Cyber job-control language commands be used within the job files sent from the SSC to the Cyber. In addition to the Cyber commands, Cray job-control commands were also embedded in the SSC's job files since the job files were ultimately interpreted and processed at the Cray.

Sequence of Presentation

Chapter II contains an analysis of the system requirements including identification of system function and performance specifications.

Chapter III describes the overall system design using the functions specified in Chapter II, while Chapter IV contains the detailed design of the system.

Chapter V gives the results of the implemented system which consists of an analysis of the system's functions and performance as compared to the system requirements.

Finally, Chapter VI presents the summary, conclusions, and future recommendations.

II. System Requirements and Justification

This chapter describes and analyzes the requirements of the AFIT-ASD Cray system. The chapter describes the objectives, description, operation, and specifications of the system so that there is a clear explanation of the system itself and how the system is to perform. Any necessary justification accompanies the stated requirements.

Objectives

The overall objective of the system design is to provide a flexible, automatic, and virtual communications system capable of sending AFIT/EN programs (jobs) from any of several AFIT host computers to the ASD Computer Center's Cray computer, and receiving the results sent back.

Specifically, the system must meet the following objectives:

1) It must be simple to use: The system must be simple for the human user to use, in terms of understanding: (a) how to operate the AFIT-ASD Cray system, and (b) what results to expect. This will allow users to quickly learn how to operate the system and to reduce user-induced errors.

2) It must be error-free: The system must be able to transfer data to and from ASD's Cray computer without error. This will ensure that no unprocessed or erroneous data returns to the user, thus reducing the number of repeat jobs needed and maintaining a consistent turnaround time.

3) It must allow fast turnaround: The system, as it transmits jobs to and returns output from ASD's Cray, must be faster than the current processing capability of AFIT's host computer resources (especially for large, compute-intensive jobs). This objective is the driving force behind the need for the system itself.

4) The system must use existing computer software and hardware to minimize development time and system complexity, and to simplify maintenance and future upgrades. This objective includes existing data communications software and hardware.

The above objectives are reflected in the requirements definition that follows. First, an overall system description is given followed by an outline of the system's operation. Next, the implementation requirements are listed and the performance requirements are expanded. Finally, an overview of the functional requirements for the system is presented.

Requirements Definition

System Description. The AFIT-ASD Cray system takes a user's input file, containing program data, from either the AFIT SSC, AFIT/ENG's ELXSI computer, or one of AFIT/ENG's Sun workstations, and automatically transmits that input file to ASD's Cray computer for processing. Once at the Cray, the input file is processed and the output file is automatically transmitted back to a previously designated AFIT host computer. For example, the input file could contain AFIT/ENG VLSI design data and the output file would then consist of VLSI design analysis data (5) that may be output on a printer. The program that would process the input data on the Cray would be a FORTRAN version of SPICE, a VLSI circuit simulation program.

The AFIT-ASD Cray system should be flexible enough to allow jobs to be entered from any of the above AFIT hosts or workstations and the output sent to any of the above AFIT hosts or workstations. The system will notify the user via an online screen message or off-line electronic mail message when output from the Cray has returned. The entire process, from Cray job initiation at an AFIT source computer to the Cray job output's return at an

AFIT destination computer, will require no user intervention.

System Operation. In general, the AFIT-ASD Cray system operates over the currently available AFIT data communications facility, which is connected to ASD's data communications facility. Specifically, the system makes use of the existing RJE link between the AFIT SSC and ASD's CDC Cyber computer (1), which is currently connected to the Cray (15). The system also communicates with AFIT/ENG's ELXSI computer and Sun workstations and the AFIT SSC via a dedicated, local computer network link (5).

Input data files created on the ELXSI or the Sun workstations and destined for ASD's Cray computer are transmitted over an Ethernet (coaxial cable) network link to the SSC (see Figure 2-1). While at the SSC, the ELXSI, Sun or SSC input data files destined for the Cray have the appropriate Cyber and Cray job-control commands placed into the input data files (15).

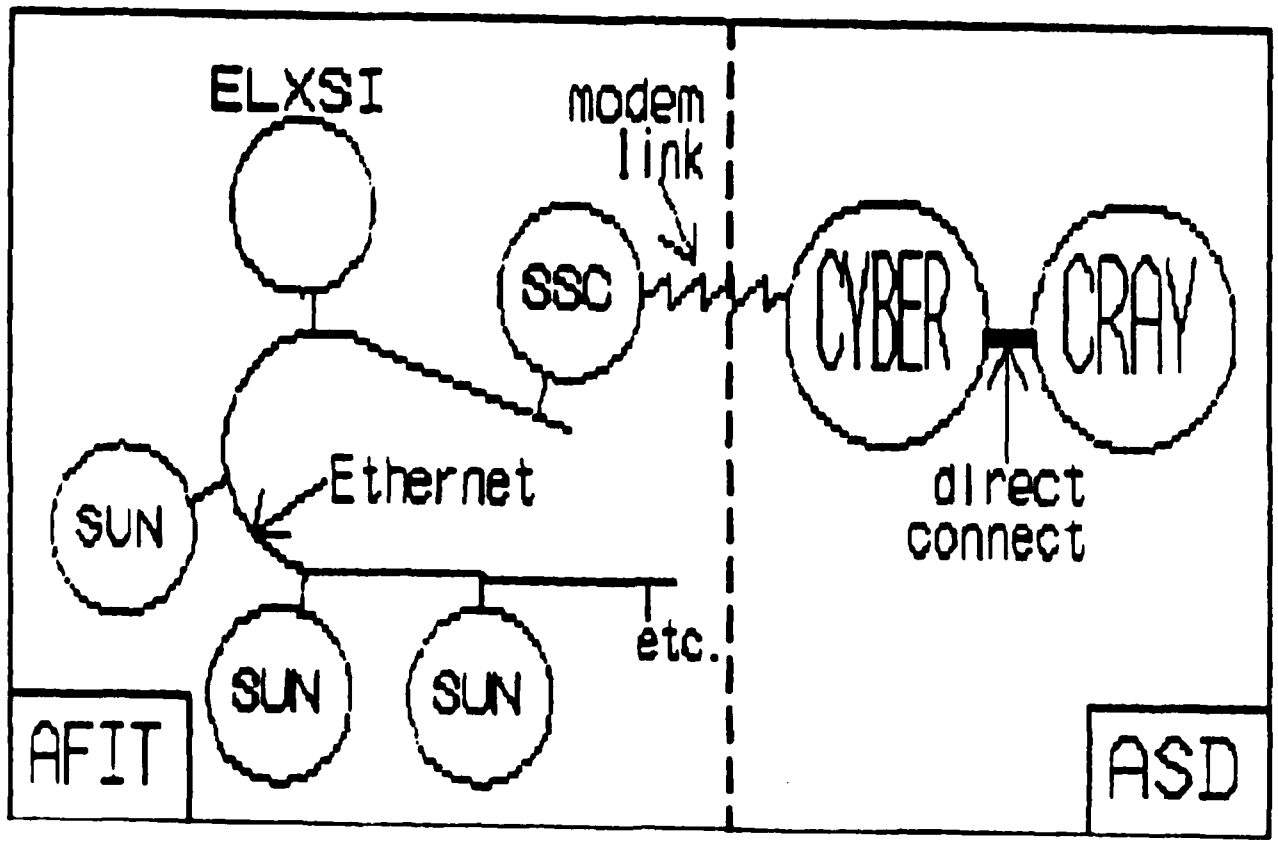


Figure 2-1 AFIT-ASD Cray System Diagram

The SSC then takes the Cray job file and transmits it to the Cyber via the existing SSC-Cyber RJE link. Once at the Cyber, the job file is transferred to the Cray, the job processed, and the output returned to the Cyber, which in turn returns the output to the AFIT SSC.

Once the SSC receives the Cray output from the Cyber, the SSC stores it or passes it on to the ELXSI or Sun workstations, using the local computer network link, depending on where the user specified the output should be sent.

The system will notify the user that the output file has returned via an on-screen message at the source computer. If the user has logged off the source computer, the system will place a message in the user's "mailbox" on the source computer using the UNIX mail command (12:114).

The system also has an audit capability which keeps track of the progress of a job as it passes from one computer to the next. A log file accompanies both the input file and corresponding output file through the system. The system sends to the user the log file along with the output file. The log file contains date/time stamps of important events (e.g., file transfer from one computer to the next) and any pertinent system messages and error messages sent by the AFIT and ASD host computers involved.

The user has the responsibility to first create an input file for the AFIT-ASD Cray system. This is accomplished by using the file editor of choice on any one of the AFIT host computers or workstations involved. Once the input file is created, the user may enter the command sendcray, a set of options, and the input filename, in that order. If no options are given, then the default options are assumed. After the input file has been sent to the Cray and processed, an output file will return to the designated AFIT host and user directory (file space).

In addition to the output file, a companion log file containing the history of events that transpired from the time the Cray job left the AFIT source computer or workstation and returned to its AFIT destination will be sent.

Implementation Requirements. The system must use existing computer software and hardware in order to minimize development time and system complexity, and to enhance maintainability. The system must also use existing data communications software and hardware to link together the computer systems involved.

Since each of the AFIT hosts and workstations involved can operate under the UNIX operating system, the system will require the use of UNIX C-shell commands in order to create the appropriate RJE commands (9). Computer-to-computer communications among these UNIX machines will require the use of standard TCP/IP data communications protocol commands over the Ethernet local area network (5).

The AFIT SSC to ASD Cyber hardware link is already established. However, the system will require that Cyber and Cray job-control commands be embedded in the AFIT user's data files.

The specific AFIT EN job to be used for an initial prototype system is a FORTRAN version of AFIT ENG's SPICE VLSI circuit simulation program. The specific AFIT host computers to be used in the prototype are the AFIT SSC (UNIX-VAX ... 785 , AFIT ENG's FLXSI computer, and the Sun workstations.

Performance Requirements. The performance criteria for the system are at the heart of the system's objectives. The need for the system grew out of a requirement to provide a simple, error-free way to access a faster non-AFIT host computer, i.e, the Cray supercomputer at ASD Computer Center.

The system must be simple to use. The user must be able to understand quickly how to operate the system and to remember how to operate it after only a brief introduction. The system will accomplish this through the use of obligatory error and usage messages (onscreen) and the use of hardcopy and online user's guides.

Accompanying the system will be a hardcopy user's guide explaining the operation of the AFIT-ASD Cray system. In addition to the hardcopy guide, an online user's manual entry will be available on each of the AFIT hosts and workstations involved. Under the UNIX operating system, the user will use the "man" manual command (12:31) to review the online user's manual entry. Besides explanation in the user's guide of the function and options for the system, the guide will also explain what outputs are produced and what they contain.

One other requirement for simplicity of use is that the transmission and running of a Cray job should be transparent to the user, i.e., should not require any user intervention.

The system must be able to transfer data to and from ASD's Cray computer without error--no missing or garbled data. This will be done by connecting each AFIT host computer or workstation involved to a local communications network. The local network is the Ethernet baseband (single-channel), coaxial cable network. The network will ensure simple, coordinated data transfers among the AFIT host computers. The Ethernet's established error-handling features will also ensure error-free transmission among the AFIT hosts. In addition, the existing features of the AFIT SSC-ASD Cyber link will ensure that the transfer of data files between AFIT and ASD will be simple and, again, error-free.

The system must have a relatively fast turnaround time. The system, as it transmits jobs to and retrieves output from ASD's Cray, must be faster than the current processing capability of AFIT's host computer resources, especially for large, compute-intensive jobs. Output from the Cray must return to a designated AFIT destination computer within a maximum of one hour.

The turnaround time will start from the time a Cray job leaves an AFIT host computer or workstation until the Cray output returns to its final AFIT destination. The turnaround time includes not only the program run-time (execution-time) at the Cray, but also the time taken to transmit input data to and output data from the Cray.

One major constraint of the system is the speed at which data is transmitted between the AFIT SSC and ASD's Cyber. The synchronous modem link between the SSC and the Cyber transmits data at 9600 bits/second. This is relatively slow compared to the maximum 10 Mbits/second transmission rate of AFIT/ENG's Ethernet local computer network. The 9600 bits/second rate is a maximum rate--with the HASP communications protocol executing between the SSC and the Cyber, the actual rate is somewhat lower. The expected large (one-half to one million characters) file transfers between the SSC and the Cyber, whether input or output files, could become a major bottleneck for the system. A careful analysis of the system turnaround time must follow system implementation to determine the effects of this constraint.

The system will be validated by demonstrating that any of the AFIT hosts or workstations involved will be able, without error, to transmit a job to ASD's Cray computer, run the job at the Cray, and return the output to any one of the involved AFIT computers or workstations. The system should operate properly under normal conditions, and recover from any abnormal conditions

such as a computer going down or a communication link being disconnected.

An attempt to install the before-mentioned SPICE circuit simulation program on ASD's Cray computer was made as a test case. The program would execute VLSI circuit design simulations on the Cray using realistic circuit design input data.

Functional Requirements. This section describes the AFIT-ASD Cray system's user command and the functions of each of the user command options, including default options.

Besides entering the system's command name and the name of the input data file, the user may enter other optional information: (1) the Cray job type; (2) the estimated Cray job run time (time to run while on the Cray, in minutes); and (3) the output filename. If no option information is entered, the default options are exercised.

The system is initiated when the user enters the following command:

sendcray [options] input_filename

The system provides the user with the following options:

(1) Cray job type. The job type can be either FORTRAN or SPICE. A FORTRAN job indicates that the input data file is a FORTRAN program, while a SPICE job indicates that the input data file is input data for a VLSI circuit simulation. The default Cray job type is FORTRAN.

(2) Cray job time. The job time is the estimated time in minutes that the user expects the Cray job to run while at the Cray. The system uses this time value to place an upper limit on how long ASD's Cyber should wait for output to return from the Cray, the maximum time being 15 minutes. The default Cray job time is 5 minutes.

(3) Output Destination. The user should be able to specify which AFIT host computer or workstation will be the final destination of a Cray job's output. The choices are the AFIT SSC, AFIT/ENG's ELXSI, or any one of AFIT/ENG's Sun workstations. The default output destination is the source computer or workstation (the machine that originated the Cray job).

(4) Output filename. The user can also specify the exact name of the Cray job's output filename. However, if no output filename is given, the default output filename will be the same as the input filename followed by a ".crayout" suffix. Accompanying the output file will be the log file for the Cray job. The name of the log file will be the same as the specified output filename followed by a ".craylog" suffix, with the default being the input filename followed by that same suffix.

After the user has entered the above information, the system performs automatically without giving the user any additional information until the output file returns. Once the output file arrives at the destination host, the system notifies the user via an online screen message (if the user is still logged-on to the source computer) or an electronic mail message (if the user has logged-off).

These functional requirements along with the other requirements set forth in this chapter are the foundation for the AFIT-ASD Cray system. These requirements also form the basis for the detailed system design, which is described in the next chapter.

III. System Design

This chapter on system design translates the requirements presented in the previous chapter into a master diagram of the system. This overall design will incorporate the functions specified under the system requirements and will show how they and their sub-functions interrelate to produce a working system.

The system design is described using the Structured Analysis and Design Technique (SADT). System functions are described using SADT charts with accompanying text.

Overview

The SADT charts in the following figures describe the functions, inputs, outputs, and controls of the AFIT-ASD Cray system. The system, though depicted as operating from a single machine (see Figure 3-1), is actually implemented on several AFIT computers: AFIT/ENG's ELXSI computer and Sun workstations, and the AFIT SSC (DEC VAX-11/785). Each of the AFIT computers uses the UNIX operating system which executes UNIX C-shell programs to implement the system described here. Some of these programs are similar for each machine while some are unique to a particular computer or set of computers.

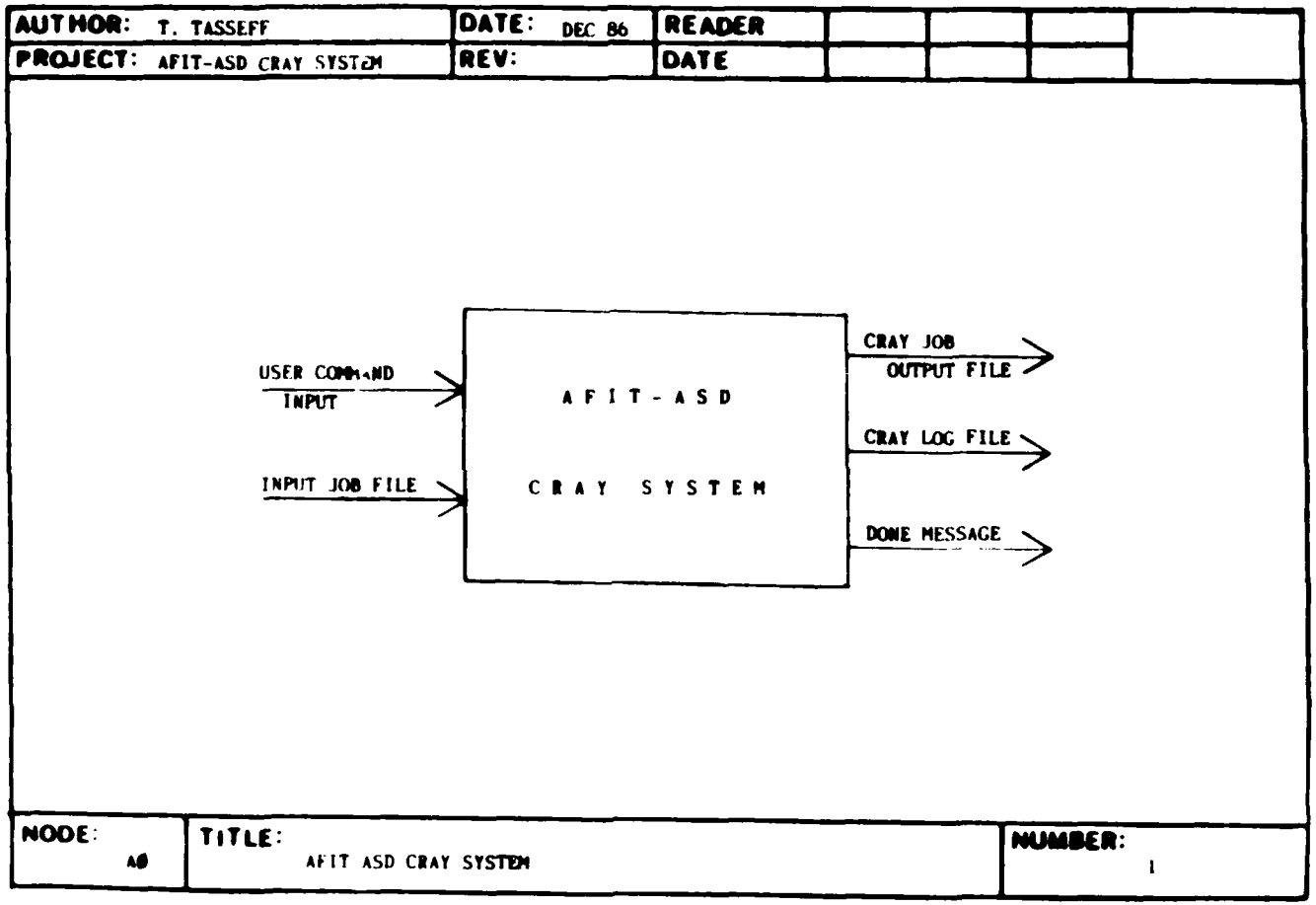

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
USER COMMAND INPUT
INPUT JOB FILE
A F I T - A S D
C R A Y S Y S T E M
CRAY JOB OUTPUT FILE
CRAY LOG FILE
DONE MESSAGE
NODE: A0
TITLE: AFIT ASD CRAY SYSTEM
NUMBER: 1

Figure 3-1

The task of initiating a Cray job is virtually identical in all the AFIT computers involved. Required user-inputs are the same no matter which computer is used, and the system tasks of creating a Cray job file and the associated log file are primarily the same.

Differences occur in the way the job and log files are submitted to the Cray and in the way those same Cray log files and associated output return to their final destinations. If the Cray job originates from or is destined to return to the AFIT SSC, the task is relatively simple since the SSC is connected directly to ASD's Cyber (and, in turn, the Cray). However, if the job originates from or is destined for any other of the involved AFIT computers, the task becomes more complex, since the job and log files must travel through the SSC enroute to the Cray.

Another important difference among the system's programs is that since only the SSC can communicate with the Cyber and Cray computers, there are specific programs resident on the SSC for sending files to and receiving files from the Cyber and Cray. The differences outlined above will be described in more detail in the next chapter which contains the detailed design.

The AFIT-ASD Cray system is split up into four main sections, Initiate Cray Job, Route Cray Job to Cray, Execute Cray Job, and Route Cray Job Output to Destination (see Figure 3-2).

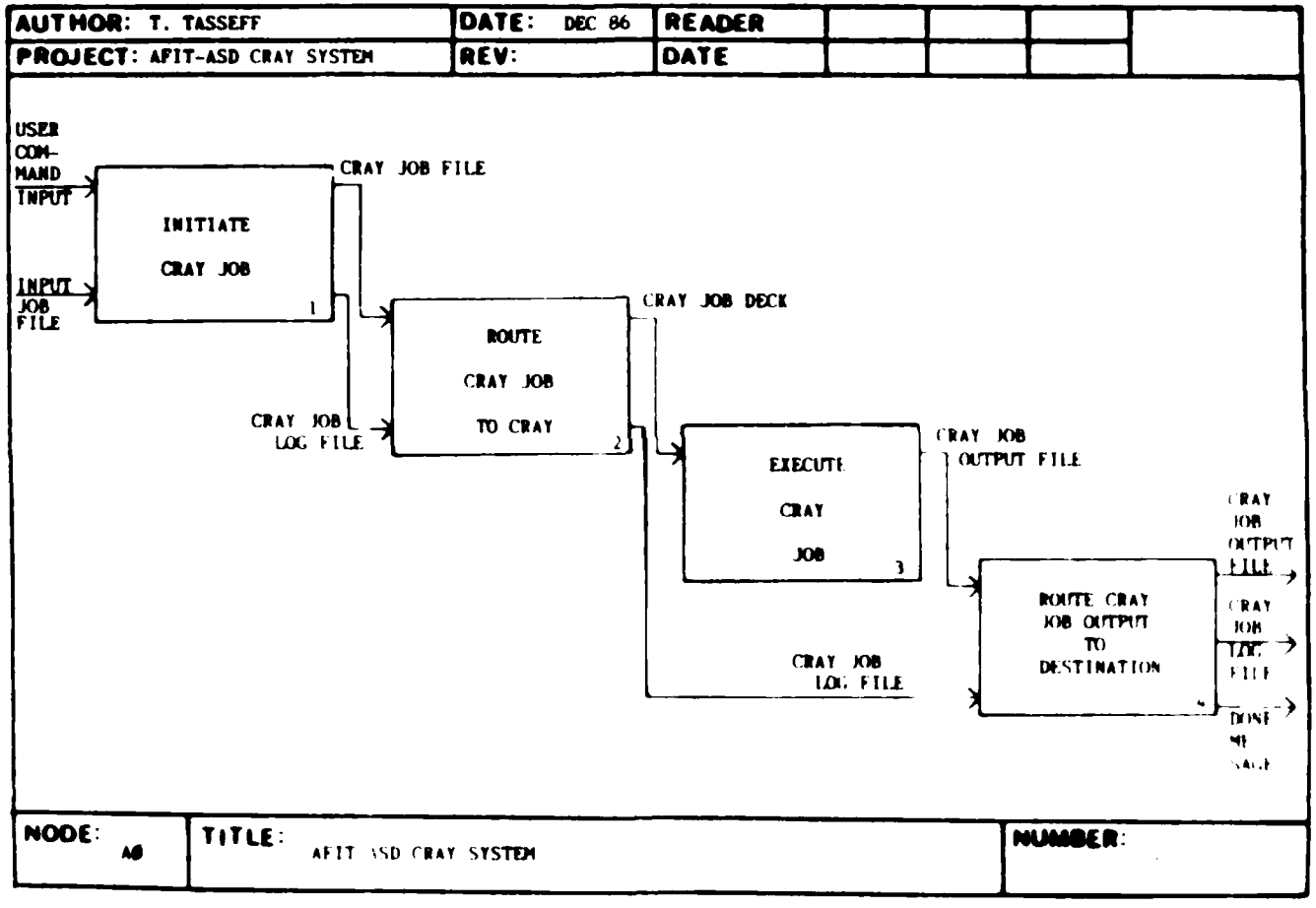

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
USER COM-MAND INPUT
INPUT JOB FILE
INITIATE CRAY JOB
1
CRAY JOB FILE
CRAY JOB LOG FILE
ROUTE CRAY JOB TO CRAY
2
CRAY JOB DECK
EXECUTE CRAY JOB
3
CRAY JOB OUTPUT FILE
CRAY JOB LOG FILE
ROUTE CRAY JOB OUTPUT TO DESTINATION
4
CRAY JOB OUTPUT FILE
CRAY JOB LOG FILE
NODE: A8
TITLE: AFIT ASD CRAY SYSTEM
NUMBER:

Figure 3-2

The function, inputs, outputs, and controls of each section or module as well as any submodules are described below.

Initiate Cray Job

The Initiate Cray Job module (Figure 3-3) takes the user's keyboard input and the user's input job file and creates a Cray job file and an accompanying Cray job log file. The input filename is identified by the user's input. The Cray job file is the same as the input file except that it has a new filename which uniquely identifies it.

Besides the input filename, the user may choose the Cray job type, the job time, an alternate output destination, and a unique output filename. The system software then places these four items into the Cray job log file.

The Cray job and log files are used by the next system module, Route Cray Job to Cray.

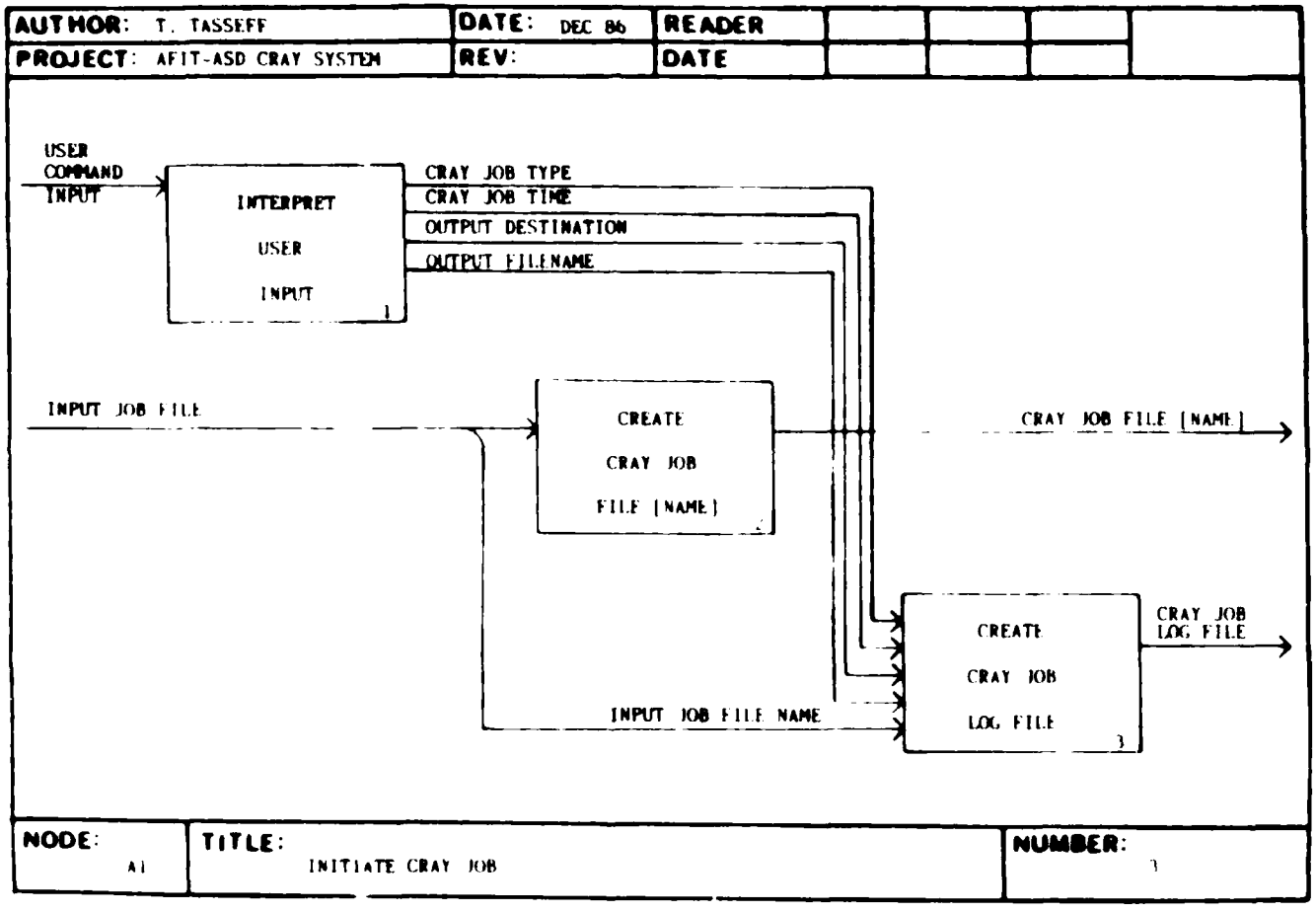

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
USER COMMAND INPUT
INTERPRET USER INPUT
1
CRAY JOB TYPE
CRAY JOB TIME
OUTPUT DESTINATION
OUTPUT FILENAME
INPUT JOB FILE
CREATE CRAY JOB FILE [NAME]
2
CRAY JOB FILE [NAME]
CREATE CRAY JOB LOG FILE
3
CRAY JOB LOG FILE
INPUT JOB FILE NAME
NODE: A1
TITLE: INITIATE CRAY JOB
NUMBER: 3

Figure 3-3

Route Cray Job to Cray

The Route Cray Job to Cray module (Figure 3-4) is really two modules in one. The first submodule is responsible for updating the Cray job log file and routing the Cray job and log files to the SSC (Figure 3-5). Once at the SSC, the second submodule (Figure 3-6) converts the Cray job file into a job deck file and sends the job deck to the Cyber and Cray for processing. The job deck contains the original input file data from the AFIT computer that initiated the Cray job plus special Cyber and Cray job control language (JCL) commands that control the flow and execution of the input file data at the Cyber and Cray. The type of JCL commands that are created will vary depending on the Cray job type that was specified by the user and stored in the job log file. The log file remains at the SSC and is updated when any transfer (or arrival) of job files occurs.

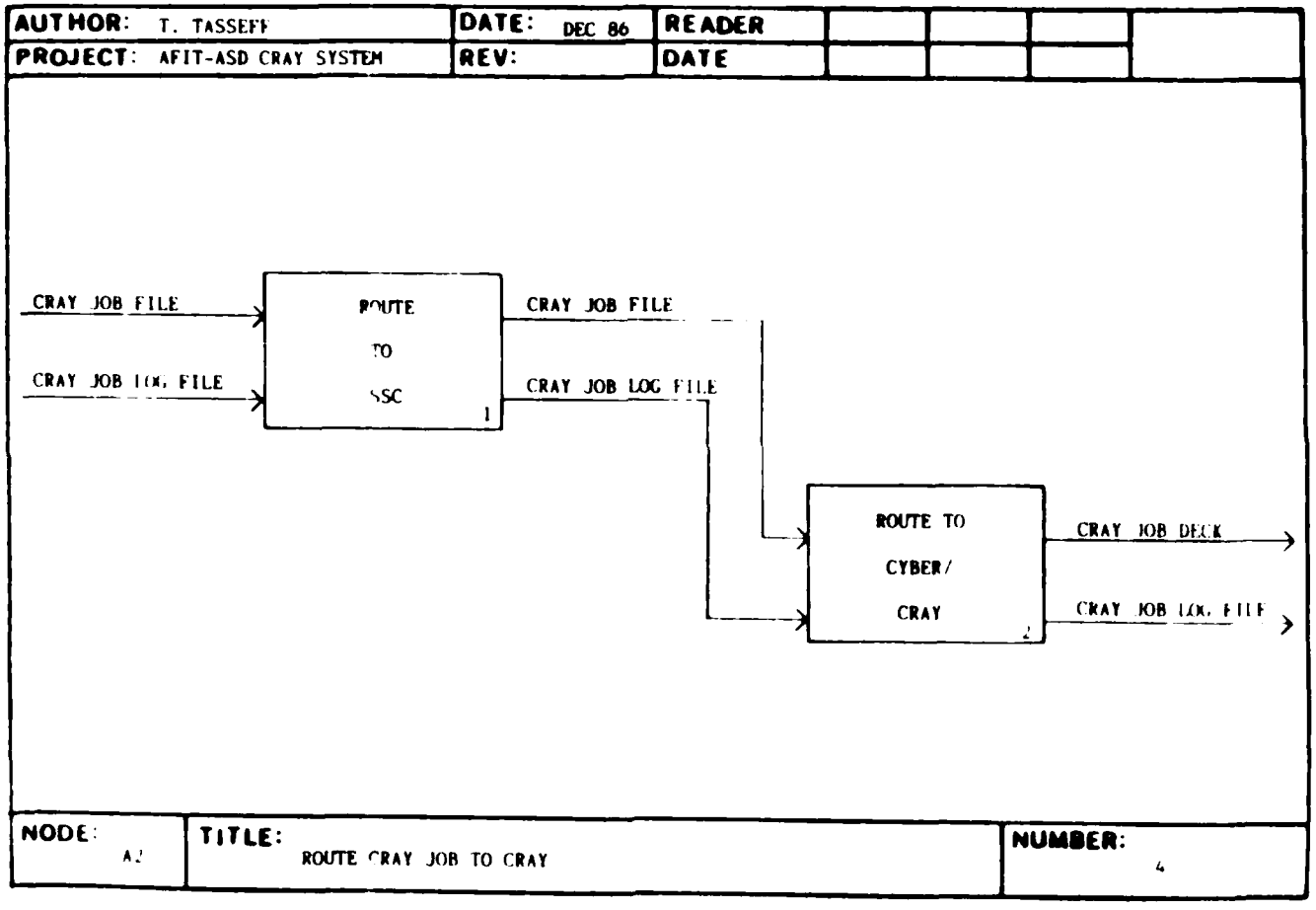

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
CRAY JOB FILE
CRAY JOB LOG FILE
ROUTE TO SSC
1
CRAY JOB FILE
CRAY JOB LOG FILE
ROUTE TO CYBER/ CRAY
2
CRAY JOB DECK
CRAY JOB LOG FILE
NODE: A2
TITLE: ROUTE CRAY JOB TO CRAY
NUMBER: 4

Figure 3-4

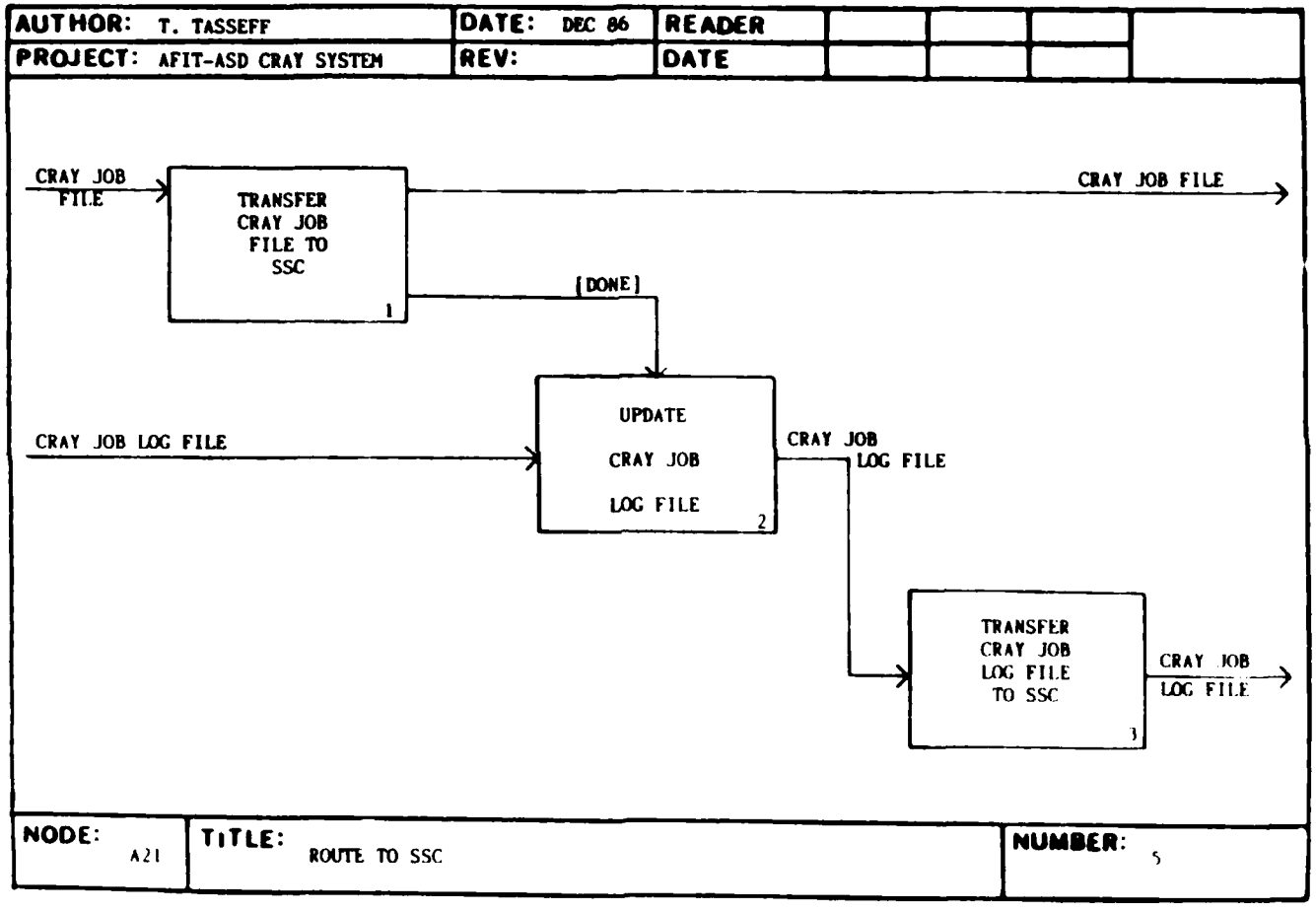

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
CRAY JOB FILE
TRANSFER CRAY JOB FILE TO SSC
1
CRAY JOB FILE
[DONE]
CRAY JOB LOG FILE
UPDATE CRAY JOB LOG FILE
2
CRAY JOB LOG FILE
TRANSFER CRAY JOB LOG FILE TO SSC
3
CRAY JOB LOG FILE
NODE: A21
TITLE: ROUTE TO SSC
NUMBER: 5

Figure 3-5

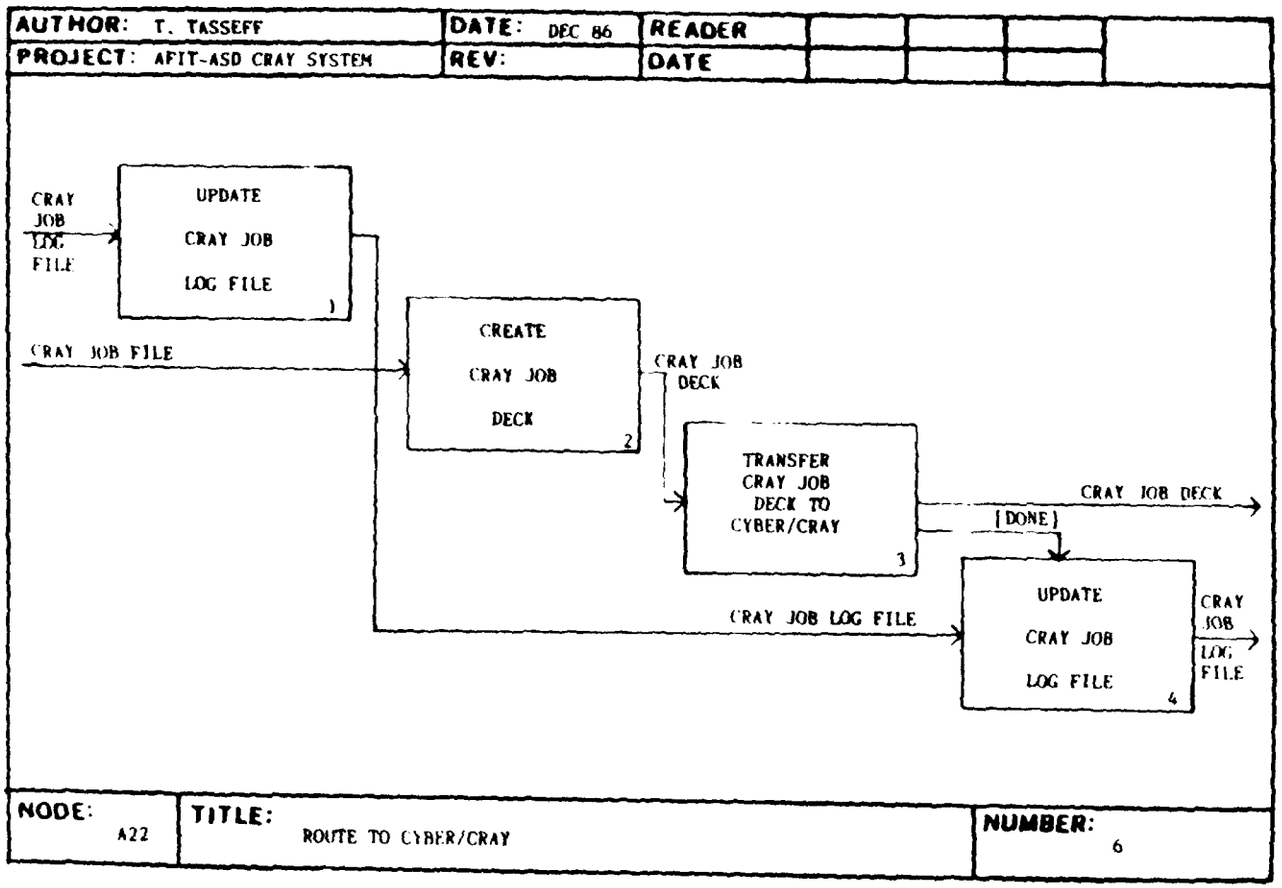

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
CRAY JOB LOG FILE
UPDATE CRAY JOB LOG FILE
1
CRAY JOB FILE
CREATE CRAY JOB DECK
2
CRAY JOB DECK
TRANSFER CRAY JOB DECK TO CYBER/CRAY
3
CRAY JOB DECK
[DONE]
CRAY JOB LOG FILE
UPDATE CRAY JOB LOG FILE
4
CRAY JOB LOG FILE
NODE: A22
TITLE: ROUTE TO CYBER/CRAY
NUMBER: 6

Figure 3-6

Execute Cray Job

This module (see Figure 3-7) is self-explanatory because the Cyber in concert with the Cray executes the JCL commands found in the job deck sent from the AFIT SSC. The Cyber JCL commands submit Cray JCL commands plus the AFIT input file to the Cray, and wait for output from the Cray to return. The Cray JCL commands then process the input file data that originated from an AFIT source computer, and return the output to the Cyber. Output from the Cray along with the Cyber output (which contains Cyber job history information) return to the SSC automatically.

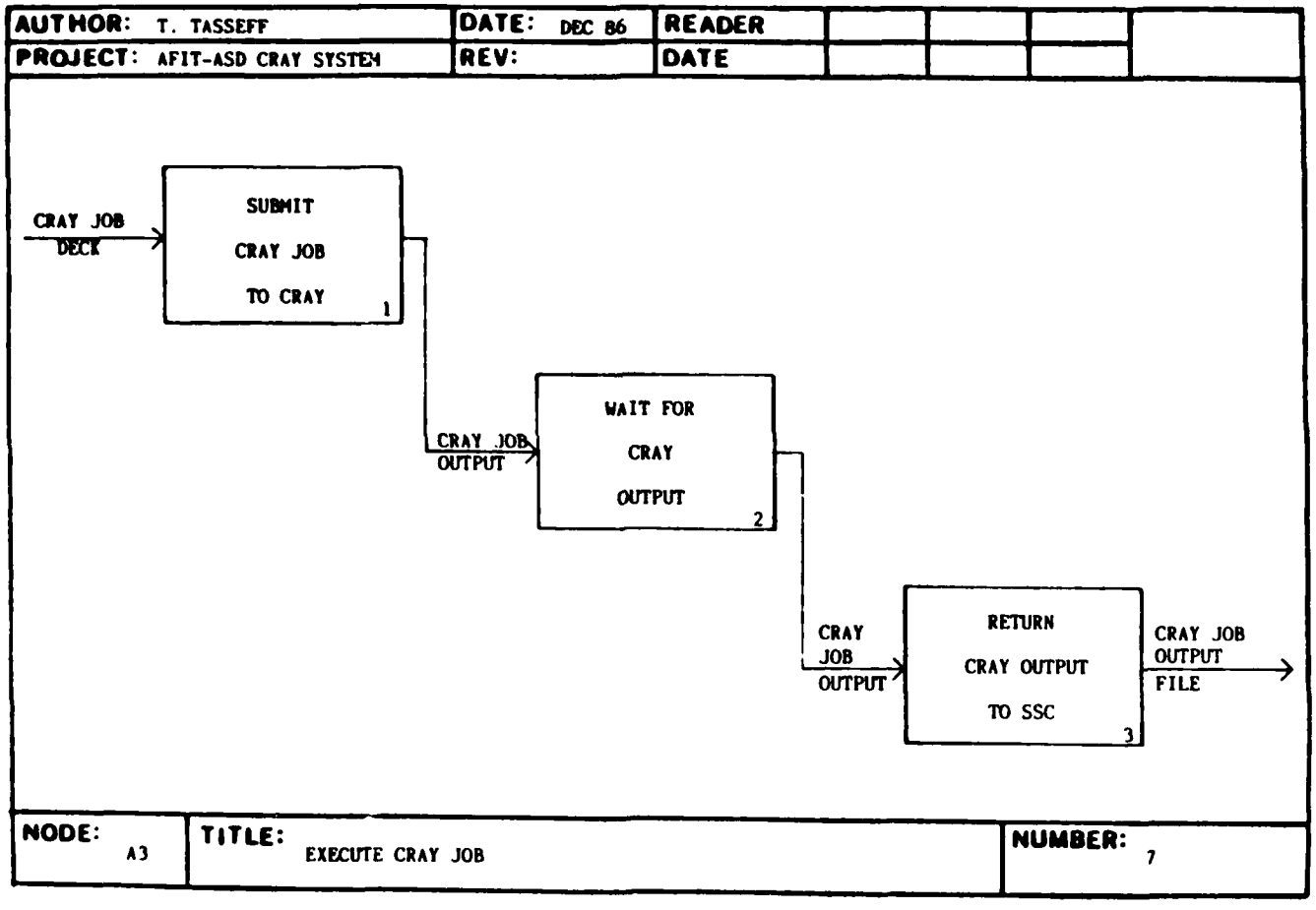

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
CRAY JOB DECK
SUBMIT CRAY JOB TO CRAY
1
CRAY JOB OUTPUT
WAIT FOR CRAY OUTPUT
2
CRAY JOB OUTPUT
RETURN CRAY OUTPUT TO SSC
3
CRAY JOB OUTPUT FILE
NODE: A3
TITLE: EXECUTE CRAY JOB
NUMBER: 7

Figure 3-7

Route Cray Job Output to Destination

The Route Cray Job Output to Destination is made up of three separate submodules (see Figure 3-8). The first submodule (Figure 3-9) updates the Cray job log file when the Cray job output arrives. Then, the output destination is extracted from the log file and the output file is transferred to the output destination. Finally, the Cray job log file is updated once more and then transferred to the output destination.

Once at the output destination, the Cray log file is updated and the Cray output and log files are delivered to the user's output filename which was designated by the user initially and stored in the log file. After the Cray output and log files are delivered to the user, a "done message" is generated. The "done message" will take the form of an onscreen message if the user is logged on to the AFIT source computer, or will take the form of a computer mail message if the user is not logged on to the AFIT source computer.

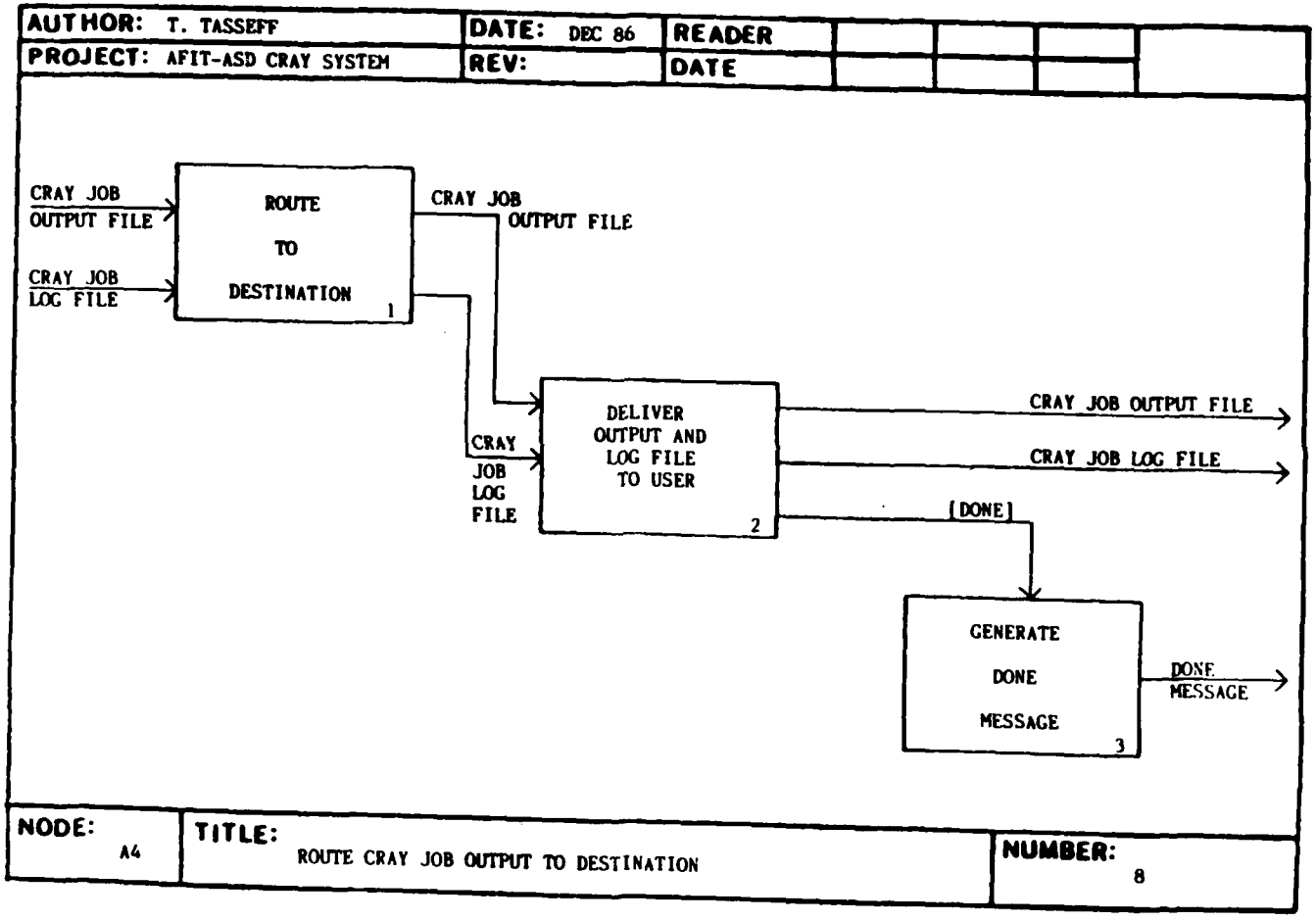

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
CRAY JOB OUTPUT FILE
CRAY JOB LOG FILE
ROUTE TO DESTINATION
1
CRAY JOB OUTPUT FILE
CRAY JOB LOG FILE
DELIVER OUTPUT AND LOG FILE TO USER
2
CRAY JOB OUTPUT FILE
CRAY JOB LOG FILE
[DONE]
GENERATE DONE MESSAGE
3
DONE MESSAGE
NODE: A4
TITLE: ROUTE CRAY JOB OUTPUT TO DESTINATION
NUMBER: 8

Figure 3-8

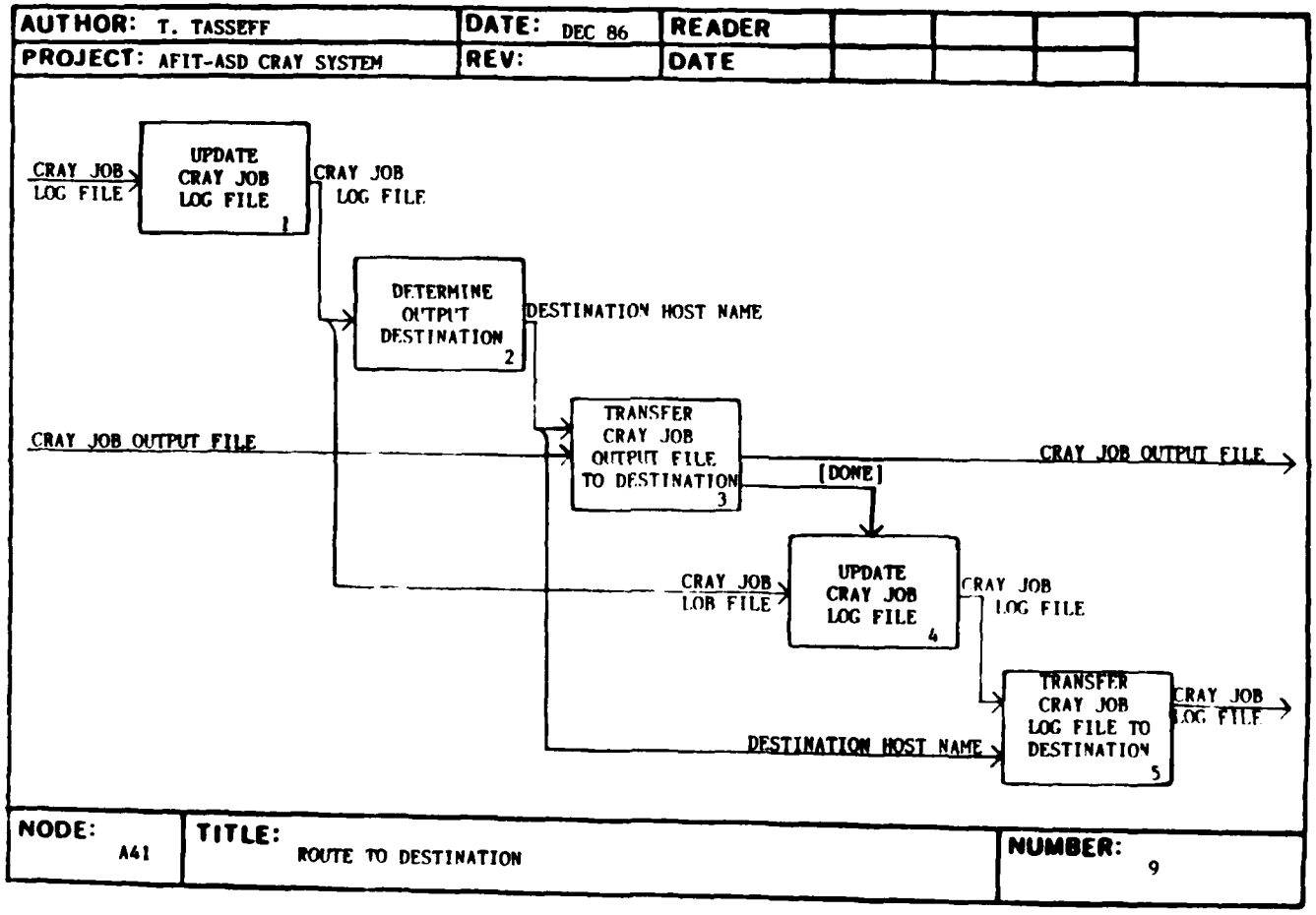

AUTHOR: T. TASSEFF
DATE: DEC 86
READER
PROJECT: AFIT-ASD CRAY SYSTEM
REV:
DATE
CRAY JOB LOG FILE
UPDATE CRAY JOB LOG FILE
1
CRAY JOB LOG FILE
DETERMINE OUTPUT DESTINATION
2
DESTINATION HOST NAME
CRAY JOB OUTPUT FILE
TRANSFER CRAY JOB OUTPUT FILE TO DESTINATION
3
CRAY JOB OUTPUT FILE
[DONE]
CRAY JOB LOG FILE
UPDATE CRAY JOB LOG FILE
4
CRAY JOB LOG FILE
TRANSFER CRAY JOB LOG FILE TO DESTINATION
5
CRAY JOB LOG FILE
DESTINATION HOST NAME
NODE: A41
TITLE: ROUTE TO DESTINATION
NUMBER: 9

Figure 3-9

IV. Detailed Design

This chapter on detailed design develops the initial, overall system design of the AFIT-ASD Cray System into a more complete system design. The detailed design describes the overall structure of the System's implementation so that the design may be translated into appropriate computer system-programming instructions.

Design Goals

The design objectives of the System have already been described in Chapter 2: (1) must be simple to use, (2) must be error-free, (3) must allow fast job turnaround, and (4) must use existing computer software and hardware. Each objective has an impact on the design goals.

The System must be simple to use and operate. Given that the System runs on AFIT computers that use the UNIX operating system, the System's user command format must conform to the UNIX style of user command-lines, i.e., a user command followed by "dash-options" (single-character options prefixed by a single dash) and ending with some kind of input (e.g., a filename). In this way, a user already familiar with UNIX command formats will be able to understand how to use the AFIT-ASD Cray System command.

The System must be able to detect invalid options, invalid option inputs, and invalid input filenames and reporting these errors back to the user in an orderly and graceful manner, i.e., without causing some abnormal or undetected System program termination. The System must also be able to detect, recover from, and report back to the user any errors in the internal operation of the

System itself (host computers unavailable, data communication links down, etc.) and do so in an orderly and graceful manner.

There is nothing inherent in the system design that will notably improve job turnaround time. ASD's Cray computer is the component that executes actual job. The System is merely a user of the Cray's exceptionally fast computational facilities. Since the System's programs are UNIX C-shell command programs, internal documentation (comments) in the production version of the System software must be kept to a minimum. Commands in UNIX C-shell programs are interpreted one at a time including the comments, although the comments are not processed in any way. Reducing the number of comments, and extraneous commands, in each of the System's programs would then decrease the execution time of the System, though not significantly. The only notable "software" improvement that could be made would be to increase the speed in which the System detects and processes data files. This improvement is done by increasing the frequency in which some of the System programs are scheduled to run by AFIT's UNIX host computers. Also, since the JCL (job-control language) commands of ASD's Cyber and Cray computers are interpreted by their respective operating systems, these commands should also be kept to a minimum.

The only other components that might be improved in order to provide faster job turnaround time are the AFIT host computers themselves and the data communications lines involved. However, this would violate the objective of using existing software and hardware to develop the System.

Design Procedure

The detailed design of the System began by expanding the information imbedded in the SADT charts of the previous chapter which described the

overall system design. The process modules in the SADT charts were converted to actual UNIX C-shell programs by implementing module functions using a series of C-shell commands. Each module was preceded by a module header, a set of comment lines describing a module's function and contents. Since several files were created to implement the modules, headers began each file. Also, comments throughout the modules explained their inner workings (though most or all of the comments, including the headers, are not used in the production version of the System).

In general, prototypes of each module, including prototypes of parts of the modules, were created first as tests to insure that the modules would function as intended. Both the syntax of the C-shell commands and the logic used within the modules were tested before the modules were allowed to process any sample Cray jobs files.

The prototyping of System modules, in terms of what functions were successfully tested and added, proceeded in this fashion: (1) FORTRAN job sent to the Cray from the AFIT SSC; (2) Cray job output received at SSC; (3) user command line created; (4) output delivered to a SSC user directory; (5) modules run automatically using SSC's cron (job scheduler) facility (explained in greater detail in the Program Descriptions section); (6) modules (with minor modifications) automatically run on AFIT/ENG's ELXSI computer and Sun workstations under each system's cron facility; and (7) attempted to send a SPICE job, as a test, to the Cray with output returning to AFIT.

Design Format

Given the relative simplicity of the resultant C-shell programs, the only format necessary was the use of the module headers as described above. An example of a module header, with an explanation of each line-item, follows:

```
*****************************************************************
*                                                               *
*     DATE:  date of latest version                             *
*     VERSION:  current version number                          *
*                                                               *
*     MODULE NUMBER:  module number (from SADT charts)          *
*     NAME:  name of module (corresponds to SADT charts)        *
*     FUNCTION:  description of module's basic function         *
*     INPUTS:  user-supplied inputs                             *
*     OUTPUTS:  user-directed outputs                           *
*     GLOBALS READ:  global (common) variables read             *
*     GLOBALS WRITTEN:  global (common) variables written       *
*     FILES READ:  files read by the module                     *
*     FILES WRITTEN:  files written by the module               *
*     HARDWARE INPUT:  hardware devices read by module          *
*     HARDWARE OUTPUT:  hardware devices written to by          *
*                         module                                *
*     CALLING MODULES:  modules that call this module           *
*     CALLED MODULES:  modules that this module calls           *
*                                                               *
*     AUTHOR(S):  person(s) who wrote the module                *
*     HISTORY:  history of the development of this module       *
*                                                               *
*****************************************************************
```

The above module header, though a part of the documentation of this System, will not be included in the production version.

Detailed Design

The detailed design follows the same order and flow of logic as did the overall system design. Each module corresponds to an SADT module as described in the previous chapter. Though all the modules in the SADT charts appear to be operating on one AFIT host computer, the System modules do not all execute on the same computer. The following descriptions of the system modules will

indicate the computer or computers in which each module resides.

The one common element of all the modules, no matter in which computer the module resides, is that there is a unique job identifier name that serves as prefix for all the files created by the System, e.g., input file, log file, output file, etc. This job identifier is created when the System is initiated for each job. The job identifier has the format:

xx99999 where xx represents the source host from which the job originated (sc=SSC, el=ELXSI, mc=Sun-Mercury, etc.)

99999 represents a 1-5 digit process (job) identifier number, furnished by the source host computer, and is assigned when the job is initiated

The purpose of the job identifier is to distinguish one set of System files from the next. This distinction prevents the System modules from using, overwriting, or otherwise destroying files belonging to another job. The size of the job identifier is restricted to seven alphanumeric characters because the identifier is also used by the System as a filename on the Cyber and the Cray, both of which restrict filename lengths to seven characters (2:p.6-5).

Program Descriptions

The AFIT-ASD Cray System consists of several 4.2 BSD UNIX C-shell programs, distributed among the above-mentioned AFIT hosts. The same program may run on different hosts and may operate basically the same, except for minor modifications based on the AFIT host in which the program resides. Each program is contained in a file with a name identical to that of the program. A program contains one or more System modules as found in the SADT charts of the previous chapter. Also, both Cyber-NOS and Cray job control language

(JCL) commands are placed in the Cray job deck before the deck is sent to the Cyber and Cray. A variety of publications were used to create the programs and job decks (1) (2) (3) (4) (6) (12) (17) as well as the expertise of both AFIT and ASD computer center personnel (8) (9) (11).

Sendcray. The sendcray program (see Appendix A) is both the System's command-line program and System initiator. Its main function is to interpret the user's command inputs and create a Cray job input file and a Cray job log file. It also has the task of routing the input and log files to the AFIT SSC. sendcray incorporates SADT module A1, "Initiate Cray Job," and submodule A21, "Route to SSC." A21 is part of module A2, "Route Cray Job to Cray." A version of sendcray runs on all the AFIT host computers involved.

Sendcray begins with a list of variables, preset to some numeric or text values. Most notable among these variables are the default values for the System's command options. Another set of important variables are the source-host variables. In order to make sendcray portable to the other AFIT hosts, the source-host variable SOURCEHOST is set to the host computer's name (e.g., ssc) while the variable SOURCEID is set to the host's two-letter identifier (e.g., sc). Also, the variable PROCESSID is set to the value of the program's 1-5 digit process (job) identifier number as furnished by the host computer.

The first items that the sendcray program processes are the command-line options and the Cray job input filename. After these input items are read, the program determines whether or not the items are usable, valid, or in range. Most importantly, the input file, the file given as input for the job, must exist. Concerning the job type and job destination options, sendcray checks these option inputs against a jobtypes.sendcray file and a

hosts.sendcray file which contain the valid job types and destination hosts, respectively (see Figure 4-1). Concerning the job time option, the amount of

```
::::::::::::::
jobtypes.sendcray
::::::::::::::
## list of valid Cray job types for the sendcray command
## the format is:  jobtype followed by list of aliases (all on same line)
fortran FORTRAN Fortran
spice SPICE Spice
::::::::::::::
hosts.sendcray
::::::::::::::
## list of valid AFIT hosts for the sendcray command; the format is:
##   primary hostname followed by list of aliases (all on same line)
zssc ssc SSC
zelxb bsd afitbsd
dsun2-1 dsun2 apollo
vsun2-1 vsun2 mercury
psun2-1 psun2 venus
isun2-1 isun2 zeus
```

Figure 4-1 Jobtypes and Hosts Support Files

time that the user expects the job to run while at the Cray must be less than 15 minutes. If any error occurs, an online error message is written to the user together with a "usage" message showing the proper command-line format. Note that a potential administrative problem with the assignment of usernames among the AFIT SSC, AFIT/ENG ELXSI and Sun computers (i.e., same person with two different usernames or same username for two different people), prevents implementation of the job destination option; however, initial attempts to program-in that option will be left in some of the programs for future use, but will be "commented-out" to prevent their accidental use.

Next, sendcray creates the Cray job identifier by combining the SOURCEID and the current PROCESSID together to form one unique identifier; it then

"echoes" the job identifier to the user as an online message (see Figure 4-2).

```
% sendcray craytestjob
sendcray: jobfile ID, sc16349, created: 19:04:23 EST
sendcray: log file created
sendcray: input and log files transferred to SSC
```

Figure 4-2 Sample Sendcray Session

The System will use this identifier for the duration of the Cray job. Sendcray then updates the file sendcray.log (see Figure 4-3) with information

```
## this is the local sendcray log file
## the format is:  date unique_jobfile_ID source destination jobtype jobtime
##   inputfile outputfile user inputfile_directory process_ID
Thu Nov 6 19:04:27 EST 1986 sc16349 ssc ssc fortran 5 craytestjob
craytestjob.crayout ttasseff /en0/gcs86d/ttasseff/cray 16349
```

Figure 4-3 sendcray.log File

about the current Cray job: date, time, job identifier, source and destination host, job type and job time, input and output filenames, the user's name and current file directory, and the PROCESSID.

Sendcray then begins to form the Cray job log file. Using the job identifier as a prefix and .craylog as the suffix for the log filename, sendcray enters Cray job information into the log file (same as for sendcray.log), then "echoes" to the user that the log file has been created (see Figure 4-2).

The next task that sendcray performs is to route the Cray job input and log files to the SSC. If the sendcray program is running on the SSC, then the input and log files are simply moved to a special "inbound" subdirectory where

they wait for the next System program to pick them up. If the sendcray program is running on the ELXSI or one of the Suns, then the input and log files must be transferred to the "inbound" subdirectory on the SSC via the AFIT/ENG Ethernet. The transfer command used in this case is called ftp (file transfer program). Any errors during the ftp transfer are noted in the log file and "echoed" to the user. Also, if there are any such transfer errors, the program will exit (terminate). Note that the ftp command works within the program only because a .netrc file was used (see Figure 4-4). The .netrc file

```
On the SSC...
machine zelxb, login ttasseff, password xxxxxxx
machine vsun2-1, login ttasseff, password xxxxxxx

On the Sun-Mercury...
machine ssc, login ttasseff, password xxxxxxx
```

Figure 4-4 .netrc File Examples

contains the name(s) of the AFIT host computer(s) to which files are to be transferred, along with the username and password for the AFIT-ASD Cray System account on that particular host. ftp uses the .netrc file to login automatically to a remote host from within a C-shell program. Every System program that uses ftp must use the .netrc file.

While being moved or transferred, the Cray job input file will take on a new filename, one with a Cray job identifier prefix and a .crayin suffix. Also, once the move or transfer is complete, a final message is "echoed" to the user indicating that the input and log files have been successfully transferred to the SSC (see Figure 4-2).

Send cyber cray. The send_cyber_cray program (see Appendix B) is the most important of the System programs. Its main function is to create a Cray job deck from the Cray job input file and to send it to ASD's Cyber, and in turn to the Cray. Send_cyber_cray incorporates SADT submodule A22, "Route to Cyber/Cray." A22 is part of module A2, "Route Cray Job to Cray." Send_cyber_cray runs solely on the AFIT SSC. Also, this program runs via the UNIX cron facility. The cron facility can execute a program of choice at a particular time or set of times throughout a regular 24-hour day within a regular 7-day week (12:536-537). By placing execution time options and the program name into a UNIX crontab file, UNIX will execute the program as indicated (see Figure 4-5). For this System implementation, send_cyber_cray

```
0,15,30,45 8-23 * * 1-6 /en0/gcs86d/ttasseff/cray/send_cyber_cray
0,15,30,45 8-23 * * 1-6 /en0/gcs86d/ttasseff/cray/send_destination
```

Figure 4-5 crontab File Example

will execute on the SSC Monday through Saturday, from 0800 to 2345 hours (which matches the availability of ASD's Cyber and Cray computers), and do so every 15 minutes. One important feature of cron is that any program run from the crontab file actually runs at the "root" level of UNIX, which has the authority to manipulate files in any user's directory, but needs full path (directory) names in order to reach a file or to execute a given command.

First, send_cyber_cray updates its send_cyber_cray.log file with a date/time stamp and a message indicating that the program has been initiated. Then the program checks to ensure that no other send_cyber_cray program is running (if, for some reason, the previous send_cyber_cray program was held

up). If another send_cyber_cray program is running, the program updates the send_cyber_cray.log with an appropriate message, and exits (terminates) in order to maintain System integrity and to avoid two programs processing the same files.

Next, the program moves into the "inbound" subdirectory to check for the existence of Cray job log files. If they exist, the program continues by determining the unique Cray job identifier from the prefix of the job log filename. If they do not exist, the program will exit.

Send_cyber_cray then checks for the existence of the Cray job input file (i.e., has a .crayin suffix) with a filename prefix matching the job identifier. If there is no matching input file, then the program places an error message in the log file, creates a dummy Cray job output file (with filename prefixed by job identifier and suffixed by .crayout), moves both the log and output files to an "outbound" subdirectory, and continues to check for the existence of other Cray job log files.

If the above check is successful, send_cyber_cray will continue by checking to see if the user who originated the Cray job is a valid user. The program extracts the user's name from the Cray job log file and compares it against a table of authorized users stored in a file called users.sendcray (see Figure 4-6). If the user is not authorized, the same error-recovery routine described for failing to find a matching input file is executed, except that different error messages are used.

```
::::::::::::::
users.sendcray
::::::::::::::
## this is the sendcray valid user list; the format is  username  followed by
##    cyberuser cyberbatchpasswd crayacct crayacctpasswd crayuser
 crayuserpasswd
##    (all on same line)
ttasseff T888888 XXXXXX T999999 ZZZZZZZ T999999 ZZZZZZZ
```

Figure 4-6 Users Support File

If the above check is successful, then send_cyber_cray will extract the user's Cyber and Cray account and user numbers, with their accompanying passwords, from the users.sendcray file. The Cyber and Cray accounts information is used to form the job deck that is sent to the Cyber and, in turn, the Cray (see Figure 4-5). The program also extracts the Cray job type (FORTRAN or SPICE) and the job time (in minutes; time that job is expected to execute in while at the Cray) from the Cray job log file. Send_cyber_cray then edits a template file called crayjobtop and replaces dummy variable names with the Cyber and Cray account numbers, user numbers, and passwords, and the job type and job time. The program also replaces any dummy filename variables with the Cray job identifier to prevent any confusion with other Cray job files on the Cyber and Cray that the System is still using. Finally, the program places the new, updated copy of the crayjobtop file on top of the Cray job input file and places an existing crayjobbot file on the bottom of the input file to complete the Cray job deck (see Figure 4-7). Send_cyber_cray

This is the crayjobtop file:

```
CRAYJOB,P1,STCSB.
USER,T888888,XXXXXX.
CHARGE,*.
COPYEI,,sc16349.
REWIND,sc16349.
CSUB,sc16349,MF=CRX,US=T888888,PW=XXXXXX.
SET,R1=0.
SET,R2=0.
SET,R3=5.
WHILE,R1=0.AND.R2.LT.R3,LOOP.
ATTACH,sc16349=sc16349/NA.
IF,FILE(sc16349,AS),CHECK.
REWIND,sc16349.
COPYEI,sc16349.
PURGE,sc16349.
SET,R1=1.
ELSE,CHECK.
ROLLOUT,60.
SET,R2=R2+1.
ENDIF,CHECK.
ENDW,LOOP.
/*EOR
JOB,JN=sc16349,US=T999999.
ACCOUNT,AC=T999999,APW=ZZZZZZZ,US=T999999,UPW=ZZZZZZZ.
CFT,L=0.
LDR.
DISPOSE,DN=$OUT,SDN=sc16349,MF=CB,DC=ST,DF=CB,DEFER,^
TEXT='DEFINE,sc16349.CTASK,ALL.'.
/EOF
```

This is the Cray job input file (FORTRAN):

```
      PRINT 1000
 1000 FORMAT(///,5X,'THE CRAY TEST MADE IT.')
      STOP
      END
```

This is the crayjobbot file:

```
/EOF
/*EOI
```

Figure 4-7 Simple Cray Job Deck

then removes the new copy of the crayjobtop file (the old one is retained) as well as the Cray job input file.

Send_cyber_cray is now ready to send the Cray job deck to the Cyber and the Cray. The program then executes the send command to send the job deck to the Cyber from the SSC as if it were a normal job, and to designate that the output file (with a filename prefixed by the Cray job identifier and suffixed by .crayout) be placed in the "outbound" subdirectory. Send_cyber_cray will then check for any errors from sending the job deck to the Cyber. If there are any errors, the program will run the same error-recovery routine described for failing to find a matching input file, except that different error messages are used. If there are no errors, then the Cray job deck file is removed, and the log file is moved to the "outbound" directory.

Once the job deck arrives at the Cyber, the particular Cyber commands in the job deck (from the crayjobtop file) instruct the Cyber to submit the Cray job input data, along with the necessary Cray commands, to the Cray as a normal job, wait for output to return from the Cray, then return the output to the SSC.

Send destination. The send_destination program (see Appendix C) is one of the last of the System programs. It's main function is to determine the Cray job files' destination, transfer the job files there, and move them to the user's file directory. Send_destination incorporates SADT module A4, "Route Cray Job Output to Destination," and runs soley on the AFIT SSC. Also, send_destination is one of the programs that runs via the UNIX cron facility (according to the same schedule as send_cyber_cray).

First, send_destination updates its send_destination.log file with a date/time stamp and a message indicating that the program has been initiated.

Then the program checks to ensure that no other send_destination program is running (if, for some reason, the previous send_destination program was held up). If another send_destination program is running, then the program updates the send_destination.log with an appropriate message, and exits (terminates).

Send_destination then initializes several variables which are preset to some numeric or text values. Most notable among these variables is the SYSTEMHOST variable, which contains the host computer's name (e.g., ssc). The other variables will be explained below.

Next, the program moves into the "outbound" subdirectory to check for the existence of Cray job log files. If they exist, the program continues by determining the unique Cray job identifier from the prefix of the job log filename. If they do not exist, the program will exit.

Send_destination then checks for the existence of the Cray job output file (i.e., has a .crayout suffix) with a filename prefix matching the job identifier. If there is no matching .crayout file, then the program begins keeping a count of how many times it has checked for the output to return (i.e., every 15 minutes the program advances the count by one).

When the count becomes equal to the value of the preset variable SEND_CHK, then send_destination checks to see if the Cray job deck is still in the send queue (i.e., checking to see if the job deck has not left the SSC). If the job deck is in the send queue, then the program places an error message in the log file, creates a dummy Cray job output file (with filename prefixed by job identifier and suffixed by .crayout) in the "outbound" subdirectory, and continues to check for the existence of other Cray job log files. If the job deck is not in the send queue, then the program will also continue to check for the existence of other Cray job log files.

When the count becomes equal to the value of the preset variable OUTPUT_CHK, then send_destination executes the same error-recovery routine described when finding that the Cray job deck is still in the send queue, except that the error messages indicate that the timeout for Cyber/Cray output is exceeded.

Once the existence of the Cray job output file is detected, send_destination extracts the user name and destination host name from the Cray job log file. If the destination host is the SSC, the program also extracts the user's file directory and the final name for the Cray output file from the log file, then moves the the Cray job log and output files into the user's file directory. The output file will have the final output filename, while the log file will have the final output filename as a prefix and .craylog as a suffix. Send_destination then checks to see if the user is presently logged on to the SSC. If so, the program will write a "done message" to the user as an online (on-screen) message. If the user is not logged on, the program will mail the user the same message via computer mail (see Figure 4-8). The "done message" has an important feature in that it

```
From root Tue Nov 11 20:12:08 1986
From: root (Charlie Root)
To: ttasseff
Status: RO
Message from sendcray: Cray output and log files for sc16349 have returned
```

Figure 4-8 Sample Done Message via Mail

contains the unique job identifier which signifies to the user which job's output has arrived, which is helpful if the user had initiated more than one job.

If the destination host is not the SSC, send_destination uses virtually the same ftp transfer routine as in the sendcray program in order to transfer the Cray job log and output files to their destination host. One difference is that if the transfer fails for any reason, no error message is "echoed" to the user but is placed in the Cray job log file nevertheless, and the transfer attempt must be repeated the next time that send_destination executes. If the transfer is successful, the Cray job log and output files are removed from the SSC.

Note that in order for the ftp to work within a program it must use the .netrc file of the System username or account. Since the send_destination program runs under the cron facility at the "root" level (or account), the program must switch the user account, using the su command (9), from "root" to the System account. This allows ftp within the program to access the System .netrc file so that ftp can successfully communicate to a fellow System account on another AFIT host computer. Any System program running under cron or "root" must use the su command before using the ftp command.

One additional feature of send_destination is that it removes any stray output files that belatedly return to the AFIT SSC from ASD's Cyber and Cray (i.e., the System has already returned the accompanying job log file to the user, reporting that the output file was lost). When a stray output file is removed from the SSC, a message signifying that action is placed in the SSC's sendcray.log file.

Send user. The send_user program (see Appendix D) is the final System program. Its main function is to move Cray job log and output files to the user's file directory on a non-SSC host computer. Send_user incorporates SADT module A4, "Route Cray Job Output to Destination," except for submodule A41,

"Route to Destination," since the Cray job log and output files are already at their destination, namely AFIT/ENG's ELXSI or one of the Sun workstations. Send_user runs only on the ELXSI and the Suns. Also, send_user is one of the programs that runs via the UNIX cron facility (according to an every-five-minute schedule).

Send_user is basically the same program as send_destination. But, since there is no other destination host to which to route the Cray job log and output files, send_user needs only to deliver the log and output files to the user's file directory and generate the appropriate "done message" (using write or mail).

To conclude this chapter, two additional important items require further comment. One important thing to note is that the Cray job log file is being updated with a date/time stamp and a message for every significant event that occurs in each of the above programs (see Figure 4-9), including any System error messages. Also, any Cray job files that are processed, or other files that are created in these programs to help process the job files, are protected from being read or altered in any way in order to maintain System integrity and to prevent possible errors or faults.

```
************** Cray job sc23925 log file **************
Wed Dec 3 11:58:17 EST 1986 : actual sendcray start time
source:  ssc     destination:  ssc
jobtype:  fortran     jobtime:  5
inputfile:  fpbench.f
outputfile:  fpbench.f.crayout
user:  ttasseff
directory:  /en0/gcs86d/ttasseff/cray
processid:  23925
Wed Dec 3 11:58:42 EST 1986 : Cray log file creation completed at ssc
*****************************************************
Wed Dec 3 11:58:46 EST 1986 : Cray job input and log files
  being transferred to SSC
Wed Dec 3 12:00:28 EST 1986 : Cray job input and log files received at SSC
Wed Dec 3 12:00:35 EST 1986 : Cray job deck being formed at SSC
618
551
Wed Dec 3 12:00:49 EST 1986 : Cray job deck send-to-cyber/cray started at SSC
Job number #1533 entered in queue `asdcyber'
Wed Dec 3 12:01:37 EST 1986 : Cray  job  deck  send-to-cyber/cray completed at
SSC
Wed Dec 3 12:15:20 EST 1986 : Cray job output received at ssc
Wed Dec 3 12:15:33 EST 1986 : Cray job output and log files sent to directory
  /en0/gcs86d/ttasseff/cray  and user notified at ssc
```

Figure 4-9 Sample Cray Job Log File

V. Analysis

This chapter analyzes the AFIT-ASD Cray System in order to report on the success of the System in meeting the prescribed specifications and its performance. This chapter covers the System's good points as well as the bad, and makes suggestions where applicable.

The analysis covers the following areas of the System design and operation: system design, functional requirements, system performance, and the current System status.

AFIT-ASD Cray System Design

The System's design, from the initial, overall design to the detailed design, covered all areas of the System's operations and functions. The use of SADT charts to document the flow of data and control provided the input needed to create the various System programs. Each program contains appropriate header information and internal documentation so that each program file and module is well-documented and explained.

The programs themselves are highly structured so that the flow of control is evident by inspection. Proper indentation of the program commands reveals the structure within the programs. Also, the use of "go-to" commands are used sparingly to prevent confusion. Despite these guidelines, however, some parts of the programs prove to be hard to read, primarily because the UNIX commands did not accept indentation properly. In general, the programs are written as clearly and efficiently as possible.

The use of consistent variable names is enforced from program to program to promote naming integrity among the programs. The names of the variables and the programs, as much as possible, are meaningful and descriptive.

Functional Requirements

The System implements the functional requirements found in Chapter II. The System takes a user's input file from the AFIT SSC and automatically transfers it to ASD's Cray computer for processing. Once at the Cray, the input file is processed, and the output file is automatically transferred back to the user at the AFIT computer from where the input file came. A log file is also implemented to keep a history of significant events over the life of a job, and accompanies the output file back to the user. After the output is sent back to the user, an onscreen message is sent to the user if the user is still logged on to the computer, or a mail message is sent instead if the user has logged off. The options of providing job type (FORTRAN or SPICE), job [execution] time [on the Cray], and output filename have been implemented.

The [alternate] job destination option was not implemented because of a potential administrative problem in assigning usernames among the involved AFIT host computers. No SPICE programs were run at the Cray because AFIT lacked a complete set of SPICE FORTRAN modules. Some of the SPICE modules were written in the C programming language, but the Cray as of yet has no C compiler. Due to a fatal system error when running ftp within a C-shell program, the AFIT/ENG ELXSI could not be used to test the System. Also, due to repeated breakdowns of the Ethernet connection to the SSC, neither AFIT/ENG's ELXSI or Sun workstations could be used in the System. Thus, only FORTRAN jobs originating at the AFIT SSC can be run on the Cray using the

present System.

Once the SPICE program is loaded on the Cray, a different set of Cray JCL commands are needed in the Cray job deck (see Figure 5-1). These commands

```
ACCESS,DN=SPICE,PDN=SPICE,ID=systemcrayacct,UQ.
ASSIGN,DN=$IN,A=FT5.
LDR,DN=SPICE.
```

Figure 5-1 Sample Cray SPICE Job Deck Commands

would replace the CFT (Cray FORTRAN compiler) and LDR command lines in the FORTRAN job deck. Note that the executable SPICE code must have open (public) access so that all users will have permission to use it.

The System is simple to use in that the user uses a familiar-looking UNIX-like command to start a job through the System. Also, onscreen usage and error messages were created to guide the user if there were any problems. In addition, hardcopy and online users guides are available to aid the user.

The System was tested to ensure that it could operate on the AFIT host computers involved, and could accommodate the options specified at job initiation. The System was also tested for various simulated error conditions (e.g., files missing in transit, output not returning from ASD Cyber/Cray, etc.) with the System consistently being able to recover from the errors, and the error messages correctly recorded in the job log file.

System Performance

The System performance was evaluated by compiling and running the same sample FORTRAN test program (see Figure 5-2) using the AFIT SSC and using the AFIT-ASD Cray System. The compile-and-run of the test program was initiated

almost at the same instant in time on both the SSC and the System via a UNIX C-shell program. This same C-shell program placed both the SSC and System test runs individually into "background" (non-interactive) mode so they could run on their own until completion. The compile-and-run on the SSC was timed by using the /bin/time command (12:473-474) while the System time was calculated by subtracting the final time stamp in the Cray job log file from the first time stamp (actual sendcray start time). The timed runs were made at various times of day over a consecutive three-day period.

```
C      FLOATING POINT BENCHMARK TEST
C      G. SCOTT OWEN 1/83
C*     MODIFIED BY CAPT TODD TASSEFF, AFIT/GCS-86D, DEC. 1986
       DIMENSION X(9)
       PI = 3.14159
C*     WRITE(*,100)
C*100 FORMAT(' ENTER THE NUMBER OF ITERATIONS - FORMAT I4 ')
C*     READ(*,200) N
       N = 500000
  200 FORMAT(I4)
       DO 20 I = 1,N
        Y = PI/2.1
        X(1) = SIN(Y)
        X(2) = COS(Y)
        X(3) = Y**2
        X(4) = SQRT(Y)
        X(5) = EXP(Y)
        X(6) = ALOG(Y)
        X(7) = Y
        X(8) = Y*PI
        X(9) = SIN(Y) * 2
        DO 10 J = 1,9
   10    A = X(J) * X(J) / PI
   20 CONTINUE
       WRITE(*,300)
  300 FORMAT(1X,'  ** FPBENCH EXECUTION FINISHED ')
       END
```

Figure 5-2 FORTRAN Test Program (14)

Table 5-1 below shows the comparison between test run times on the AFIT SSC vs. the AFIT-ASD Cray System. The SSC run times ranged from approximately

TABLE 5-1

AFIT-ASD Cray System Performance

	Time taken to run FORTRAN test (hrs:min) at various times of day				
Computer System	1000	1200	1500	1800	2300
AFIT SSC	04:14	04:36	02:05	06:51	04:00
AFIT-ASD Cray System	00:31	00:17	00:19	00:16	00:19

two to seven hours per run, while the AFIT-ASD Cray System maintained a consistent run time close to 15 minutes. Except for the 1000 run, each run was initiated within five minutes before the hour. The 1000 run was initiated almost right on the hour and incurred a longer run time (see explanation below).

The variations in time for the SSC test runs were most likely due to the computing load that the SSC was experiencing during the run times. The AFIT-ASD Cray System test run times were fairly consistent due to the timed nature of the System programs and the effeciency of the compile-and-run time on the Cray itself (approximately one-half minute of total Cray time per run).

Since the send_cyber_cray and send_destination programs of the AFIT-ASD Cray System execute every 15 minutes (on the quarter hour), a System run would be expected to complete in approximately 15 to 30 minutes. In this scenario the send_cyber_cray program begins running at the quarter hour, receives the input file (in this case the FORTRAN test program) at the SSC, packages it as

a job deck and sends it to ASD's Cyber and Cray. Within a few minutes the output returns to the SSC. At the next quarter hour, the send_destination program begins running, receives the Cyber/Cray output, and transfers the output to the user.

Therefore, since all of the test runs were made within five minutes before the hour (hence, before the quarter hour), the AFIT-ASD Cray System test runs completed in 15 minutes plus a few extra minutes (the extra minutes being the amount of time before the hour when the test runs were initiated). Since the 1000 test run was made too close to the hour, the send_cyber_cray program missed receiving the input file. The 1000 test run then had to wait an additional 15 minutes before the send_cyber_cray program finally received it, making for a longer overall run time.

System Status

The AFIT/SI computer center staff are ready to assume the responsibility for the System. They will be able to operate the System well into the future. In addition to the user's guide shown in Appendix E, an administrator's guide is included in Appendix F to instruct the staff on various items which include how to set up the System, add users to the System, interpret error messages, and to be aware of possible problems.

VI. Conclusions and Recommendations

Conclusions

The AFIT-ASD Cray System as a whole successfully accomplished its task of providing a virtual communications capability to ASD's Cray given the existing computer hardware and software at AFIT. It proved to be simple to use and was robust enough to tolerate and recover from various errors and fault conditions. And as an alternative to running time-consuming, compute-intensive jobs, such as the SPICE VLSI circuit simulation package, on AFIT computers, it has the potential of providing a faster turnaround time than what is possible with AFIT computer resources.

The System proved that such a system was feasible to implement, despite the three different operating systems and two different types of communications links. And the System operated automatically -- without user intervention. This made the inner workings of the System transparent to the user, hiding such details from the user so that the user was left with the simplest interface to the System possible.

Recommendations

Some improvements to the hardware support are required in order to improve the System's capabilities. First, the 9600 bit/sec link between the AFIT SSC and ASD's Cyber needs to be upgraded to a higher speed, at least somewhat approaching the Ethernet's 10Mbits/sec capability, so as not to remain the communications bottleneck in the System. One possibility is to upgrade the SSC-to-Cyber link with a fiber-optic cable in order to approach

speeds in the Mbit range. Second, the more often System programs running under UNIX's cron can be initiated the faster the System can turn around Cray jobs. Third, once the administrative problem of assigning usernames to users on the involved AFIT host computers is solved and becomes more streamlined, then the option to choose an alternate job destination for a Cray job's output could be implemented.

The problems that occurred during the implementation of the System demonstrate that AFIT must obtain a complete FORTRAN version of SPICE in order to run SPICE on the Cray as was originally intended. Also, care should be taken when implementing the System over a wide range of supposedly compatible UNIX computers. As was the case with the ELXSI, there was some error in the ELXSI's version of 4.2 BSD UNIX that prevented ftp from being executed within a C-shell program, even though it worked on the AFIT SSC and the AFIT/ENG Sun.

The System was designed with the latent goal that the System needed to be fully implemented on the various AFIT hosts involved, and be done at a privilege (authority and priority within the computer) higher than the user, that is, at the UNIX "root" level. This means some program commands were written in such a way as not to completely preclude the programs from being loaded into their own system file directories on the involved AFIT hosts, and run at the higher privilege level. By following the comments within the programs, and by following the Administrator's Manual for the System, a computer system's administrator should be able to implement the System on any of the involved AFIT computers.

The System itself serves as a model for other such remote job entry requirements at AFIT. Other software packages could be loaded on the Cray, and with a few minor changes to one of the System programs, those packages

could also be run automatically on the Cray using the System. If AFIT develops requirements to initiate jobs on physically separate computers, the System could be used as an example for system programmers. Specifically, the System could be used by those looking for a solution to implement and control the execution of programs on a different computer than what the users normally use.

Bibliography

1. Air Force Institute of Technology Information Systems Directorate. "Remote Job Entry Is Back." AFIT SSC News (computer message). 7 January 1986.

2. ASD Information Systems Center. ASD Computer Center CDC NOS ser's Guide. Revision B. September 1983.

3. ASD Information Systems Center. On-Line Information System, Section 12, Cray Information. 6 June 1986.

4. ASD Information Systems Center. NOS Procedure User Guide. May 1983.

5. Carter, Lt Col Harold, Professor, Department of Electrical and Computer Engineering. Briefing and personal interviews. AFIT/ENG, Wright-Patterson AFB, OH, January through July 1986.

6. Cray X-MP and Cray-1 Computer Systems. Cray-OS Version 1 Ready Reference Manual, SQ-0023. Revision E. Cray Research, Inc., 1984.

7. Day, John D. "Terminal, File Transfer, and Remote Job Protocols for Heterogeneous Computer Networks," Protocols and Techniques for Data Communications Networks, edited by Franklin F. Kuo. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

8. Filer, 1Lt Robert, Systems Analyst. Personal Interviews. AFIT/SIO, Wright-Patterson AFB, OH, June through September 1986.

9. Hamlin, Joseph, Systems Analyst. Personal Interviews. AFIT/SIO, Wright-Patterson AFB, OH, April through November 1986.

10. Horton, Capt Kirk S. A Microcomputer-based Program for Printing Check Plots of Integrated Circuits Specified in Caltech Intermediate Form. MS thesis, AFIT/GE/ENG/84D-35. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1984.

11. Johnson, Kenneth, Systems Analyst. Personal Interviews. ASD Information Systems Resource Center, Wright-Patterson AFB, OH, July through November 1986.

12. McGilton, Henry, and Rachel Morgan. Introducing the UNIX System. New York: McGraw-Hill Book Co., 1983.

13. McLeod, Capt Thomas M. Design of Resource-Sharing Network Link. MS thesis, AFIT/GE/EE/80D-30. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1980, (AD-A100793).

14. Owen, G. Scott. "Benchmarking the 8087 Numeric Coprocessor," Personal Computer Age, March 1983, p. 58.

15. Reinhard, Ray, Systems Analyst. Personal Interview. ASD Information Systems Resource Center, Wright-Patterson AFB, OH, 14 April 1986.

16. Tanenbaum, Andrew S. Computer Networks. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

17. University of California, Berkeley. UNIX Programmer's Manual: The C Shell, Seventh Edition, Virtual VAX-11 Version. Computer Science Division, Department of Electrical Engineering and Computer Science, Berkeley CA, November 1980.

18. Ware, Jim, Systems Analyst (on contract from SRL, Inc.). Personal Interviews. AFIT/SIO, Wright-Patterson AFB, OH, April 1986.

System Program Listing - sendcray

```
#!/bin/csh -f
#************************************************************************
#*                                                                      *
#*      DATE:  December 5, 1986                                         *
#*      VERSION:  1.0                                                   *
#*                                                                      *
#*      TITLE:  AFIT - ASD Cray System                                  *
#*      FILENAME:  sendcray                                             *
#*      OWNER:  ttasseff (Capt Todd Tasseff, GCS-86D)                   *
#*      SOFTWARE SYSTEM: AFIT SSC - 4.2 BSD Unix Operating              *
#*                       System                                         *
#*      USE:  for Lt Col Carter, EENG Thesis Project                    *
#*      CONTENTS:  Modules A1, Initiate Cray Job, and A21,              *
#*                   Route to SSC                                       *
#*                                                                      *
#************************************************************************
#
#************************************************************************
#*                                                                      *
#*      DATE:  December 5, 1986                                         *
#*      VERSION:  1.0                                                   *
#*                                                                      *
#*      MODULE NUMBER:  A1                                              *
#*      NAME:  Initiate Cray Job                                        *
#*      FUNCTION:  Interprets the user command inputs, creates          *
#*                   a Cray job input file and log file.                *
#*      INPUTS:  From keyboard                                          *
#*      OUTPUTS:  Status messages (on-screen)                           *
#*      GLOBALS READ:  None.                                            *
#*      GLOBALS WRITTEN:  job file identifier                           *
#*      FILES READ:  user-produced input file                           *
#*      FILES WRITTEN:  jobfile.crayin, jobfile.craylog                 *
#*      HARDWARE INPUT:  None.                                          *
#*      HARDWARE OUTPUT:  None.                                         *
#*      CALLING MODULES:  None.                                         *
#*      CALLED MODULES:  None                                           *
#*                                                                      *
#*      AUTHOR:  Capt Todd Tasseff                                      *
#*      HISTORY:  1.0 original version (Capt Todd Tasseff)              *
#*                Note:  lines beginning with ## refer to               *
#*                       items necessary for implementing               *
#*                       the program at root level                      *
#*                Note:  lines beginning with ### refer to              *
#*                       features not yet implemented                   *
#*                                                                      *
#************************************************************************
## $SYSTEMDIR/ to be placed in front of sendcray.log, $jobfile.craylog, and
##   the inbound subdirectory
```

```
##set SYSTEMDIR = ~ttasseff/cray
set startime = `date`
set SOURCEHOST = ssc
set SOURCEID = sc
set PROCESSID = $$
@ JOBFLAG = 0
@ TIMEFLAG = 0
@ OUTFLAG = 0
###@ DESTFLAG = 0
set destination = $SOURCEHOST
set DEFJOBTYPE = fortran
@ MAXJOBTIME = 15
@ DEFJOBTIME = 5
set inputfile = 'not_given_and_so'
# Get command-line options and inputfilename
while ($#argv)
        switch ($argv[1])
                case -j:
                        @ JOBFLAG++
                        shift
                        set jobtype = $argv[1]
                        shift
                        breaksw
                case -t:
                        @ TIMEFLAG++
                        shift
                        set jobtime = $argv[1]
                        shift
                        breaksw
                case -o:
                        @ OUTFLAG++
                        shift
                        set outputfile = $argv[1]
                        shift
                        breaksw
###             case -d:
###                     @ DESTFLAG++
###                     shift
###                     set destination = $argv[1]
###                     shift
###                     breaksw
                case -:
                        echo 'sendcray:   -   must   be   followed   by   a   valid
option'
                        goto usagemsg
                        breaksw
                default:
                        set inputfile = $argv[1]
```

```
                        shift
                        if (`expr $inputfile : "-."`) then
                           echo 'sendcray:' $inputfile 'invalid option'
                           goto usagemsg
                        else if ($#argv != 0) then
                           echo 'sendcray: filename' $inputfile 'must' \
                              'follow options'
                           goto usagemsg
                        endif
                        breaksw
        endsw
end
# Check for non-existence of inputfile
if (! -e $inputfile) then
   echo 'sendcray: filename' $inputfile 'does not exist'
   goto usagemsg
endif
#
#### If DESTFLAG is set, use $destination; else set $destination to default
###if ($DESTFLAG != 0) then
###   # Check for existence of destination host
###   set destchk = `grep $destination hosts.sendcray | awk '{print $1}'`
###   if ($destchk == '') then
###      echo 'sendcray: host' $destination 'invalid; see hosts.sendcray'
###      goto usagemsg
###   else
###      set destination = $destchk
###   endif
###endif
#
# If JOBFLAG is set, use $jobtype; else set $jobtype to default
if ($JOBFLAG != 0) then
# Check for existence of job type
  set job = `grep $jobtype jobtypes.sendcray | awk '{print $1}'`
  if ($job == '') then
     echo 'sendcray: jobtype' $jobtype 'invalid; see jobtypes.sendcray'
     goto usagemsg
  else
     set jobtype = $job
  endif
else
  set jobtype = $DEFJOBTYPE
endif
#
# If TIMEFLAG is set, use $jobtime; else set $jobtime to default
if ($TIMEFLAG != 0) then
# Check to make sure max job time is not exceeded
  if ($jobtime > $MAXJOBTIME) then
```

```
      echo 'sendcray: jobtime' $jobtime 'exceeds max jobtime of' $MAXJOBTIME \
        'minutes'
      goto usagemsg
    endif
  else
    set jobtime = $DEFJOBTIME
  endif
  #
  # If OUTFLAG is not set, set $outputfile to default (else, continue)
  if ($OUTFLAG == 0) then
     set outputfile = $inputfile.crayout
  endif
  #****************
  # usagemsg prints a Usage message then exits
  goto endusagemsg
  usagemsg:
  echo 'Usage: sendcray [ -j job_type ] [ -t number_of_min. ]'
  echo '  [ -o output_filename ] input_filename'
  ###echo '  [ -o output_filename ] [ -d AFIT_destination_host ] input_filename'
  exit ()
  endusagemsg:
  #****************
  #
  # Create unique Cray job file identifier, using the source ID as a prefix and
  #   the current sendcray process ID as the suffix, and echo it as an online
  #   message
  set jobfile = $SOURCEID$PROCESSID
  echo 'sendcray: jobfile ID,' $jobfile', created:' \
    `date | awk '{print $4 " " $5}'`
  #
  # Check to see if local source sendcray log file exists; if it does not,
  #   create it and protect it
  if (! -e sendcray.log) then
     cp /dev/null sendcray.log ; chmod 0600 sendcray.log
     echo '## this is the local sendcray log file' > sendcray.log
     echo '## the format is:  date unique_jobfile_ID source destination jobtype'
  \
       'jobtime' >> sendcray.log
     echo '##   inputfile outputfile user inputfile_directory process_ID' \
       >> sendcray.log
  endif
  # Update sendcray.log with Cray job information
  echo `date` $jobfile $SOURCEHOST $destination $jobtype $jobtime $inputfile \
    $outputfile `whoami` `pwd` $PROCESSID >> sendcray.log
  #
  # Create Cray job log file (using job file name and .craylog suffix),
  #   give it protection, then enter Cray job information into job log file
  cp /dev/null $jobfile.craylog ; chmod 0644 $jobfile.craylog
```

```
echo '************** Cray job' $jobfile 'log file **************' \
  > $jobfile.craylog
echo $startime ': actual sendcray start time' >> $jobfile.craylog
echo  'source:    '   $SOURCEHOST    '        destination:    '  $destination  >>
$jobfile.craylog
echo 'jobtype: ' $jobtype '    jobtime: ' $jobtime >> $jobfile.craylog
echo 'inputfile: ' $inputfile >> $jobfile.craylog
echo 'outputfile: ' $outputfile >> $jobfile.craylog
echo 'user: ' `whoami` >> $jobfile.craylog
echo 'directory: ' `pwd` >> $jobfile.craylog
echo 'processid: ' $PROCESSID >> $jobfile.craylog
echo `date` ': Cray log file creation completed at' $SOURCEHOST \
  >> $jobfile.craylog
echo '************************************************************' \
  >> $jobfile.craylog
# Echo Cray job log file creation message as an online message to the user
echo 'sendcray: log file created'
#
#*****************************************************************
#*                                                               *
#*      DATE:  December 5, 1986                                  *
#*      VERSION:  1.0                                            *
#*                                                               *
#*      MODULE NUMBER:  A21                                      *
#*      NAME:  Route to SSC                                      *
#*      FUNCTION:  Transfers Cray job file to the SSC  inbound   *
#*                 directory, updates Cray job log file,         *
#*                 and transfers the log file to the SSC         *
#*                 inbound  directory.                           *
#*      INPUTS:  None.                                           *
#*      OUTPUTS:  Status messages (on-screen)                    *
#*      GLOBALS READ:  job file identifier                       *
#*      GLOBALS WRITTEN:  None.                                  *
#*      FILES READ:  Cray job input file                         *
#*      FILES WRITTEN:  jobfile.crayin, jobfile.craylog          *
#*      HARDWARE INPUT:  None.                                   *
#*      HARDWARE OUTPUT:  AFIT/ENG Ethernet (if ftp used)        *
#*      CALLING MODULES:  None.                                  *
#*      CALLED MODULES:  None.                                   *
#*                                                               *
#*      AUTHOR:  Capt Todd Tasseff                               *
#*      HISTORY:  1.0 original version (Capt Todd Tasseff)       *
#*                                                               *
#*****************************************************************
#
# Create Cray job file-space in  inbound  directory, protect it, then copy
#   Cray inputfile into it
cp /dev/null inbound/$jobfile.crayin ; chmod 0600 inbound $jobfile.crayin
```

System Program Listing - sendcray

```
cp $inputfile inbound/$jobfile.crayin
# Update Cray job log file
echo `date` ´: Cray job input and log files´ >> $jobfile.craylog
echo ´  being transferred to SSC´ >> $jobfile.craylog
# Move Cray job log file to  inbound  directory
mv $jobfile.craylog inbound/.
# Echo Cray job input and log file transfer message as online message to user
echo ´sendcray: input and log files transferred to SSC´
#
#       *** END OF FILE sendcray ***
```

System Program Listing - sendcray

```
#!/bin/csh -f
#*****************************************************************
#*                                                               *
#*      DATE:  December 5, 1986                                  *
#*      VERSION:  1.0                                            *
#*                                                               *
#*      TITLE:  AFIT - ASD Cray System                           *
#*      FILENAME:  sendcray                                      *
#*      OWNER:  ttasseff (Capt Todd Tasseff, GCS-86D)            *
#*      SOFTWARE SYSTEM: AFIT/ENG Sun - 4.2 BSD Unix Operating   *
#*                       System                                  *
#*      USE:  for Lt Col Carter, EENG Thesis Project             *
#*      CONTENTS:  Modules A1, Initiate Cray Job, and A21,       *
#*                   Route to SSC                                *
#*                                                               *
#*****************************************************************
#
#*****************************************************************
#*                                                               *
#*      DATE:  December 5, 1986                                  *
#*      VERSION:  1.0                                            *
#*                                                               *
#*      MODULE NUMBER:  A1                                       *
#*      NAME:  Initiate Cray Job                                 *
#*      FUNCTION:  Interprets the user command inputs, creates   *
#*                   a Cray job input file and log file.         *
#*      INPUTS:  From keyboard                                   *
#*      OUTPUTS:  Status messages (on-screen)                    *
#*      GLOBALS READ:  None.                                     *
#*      GLOBALS WRITTEN:  job file identifier                    *
#*      FILES READ:  user-produced input file                    *
#*      FILES WRITTEN:  jobfile.crayin, jobfile.craylog          *
#*      HARDWARE INPUT:  None.                                   *
#*      HARDWARE OUTPUT:  None.                                  *
#*      CALLING MODULES:  None.                                  *
#*      CALLED MODULES:  None                                    *
#*                                                               *
#*      AUTHOR:  Capt Todd Tasseff                               *
#*      HISTORY:  1.0 original version (Capt Todd Tasseff)       *
#*                Note:  lines beginning with ## refer to        *
#*                       items necessary for implementing        *
#*                       the program at root level               *
#*                Note:  lines beginning with ### refer to       *
#*                       features not yet implemented            *
#*                                                               *
#*****************************************************************
#
#  [illegible] be placed in front of sendcray.log, $jobfile.craylog, and
#  [illegible] subdirectory
```

```
##set SYSTEMDIR = ~ttasseff/cray
set startime = `date`
set SOURCEHOST = mercury
set SOURCEID = mc
set PROCESSID = $$
@ JOBFLAG = 0
@ TIMEFLAG = 0
@ OUTFLAG = 0
###@ DESTFLAG = 0
set destination = $SOURCEHOST
set DEFJOBTYPE = fortran
@ MAXJOBTIME = 15
@ DEFJOBTIME = 5
set inputfile = 'not_given_and_so'
# Get command-line options and inputfilename
while ($#argv)
        switch ($argv[1])
                case -j:
                        @ JOBFLAG++
                        shift
                        set jobtype = $argv[1]
                        shift
                        breaksw
                case -t:
                        @ TIMEFLAG++
                        shift
                        set jobtime = $argv[1]
                        shift
                        breaksw
                case -o:
                        @ OUTFLAG++
                        shift
                        set outputfile = $argv[1]
                        shift
                        breaksw
###             case -d:
###                     @ DESTFLAG++
###                     shift
###                     set destination = $argv[1]
###                     shift
###                     breaksw
                case -:
                        echo 'sendcray:   -   must   be   followed   by   a   valid
option'
                        goto usagemsg
                        breaksw
                default:
                        set inputfile = $argv[1]
```

```
                        shift
                        if (`expr $inputfile : "-."`) then
                           echo 'sendcray:' $inputfile 'invalid option'
                           goto usagemsg
                        else if ($#argv != 0) then
                           echo 'sendcray: filename' $inputfile 'must' \
                              'follow options'
                           goto usagemsg
                        endif
                        breaksw
        endsw
end
# Check for non-existence of inputfile
if (! -e $inputfile) then
   echo 'sendcray: filename' $inputfile 'does not exist'
   goto usagemsg
endif
#
#### If DESTFLAG is set, use $destination; else set $destination to default
###if ($DESTFLAG != 0) then
###   # Check for existence of destination host
###   set destchk = `grep $destination hosts.sendcray | awk '{print $1}'`
###   if ($destchk == '') then
###      echo 'sendcray: host' $destination 'invalid; see hosts.sendcray'
###      goto usagemsg
###   else
###      set destination = $destchk
###   endif
###endif
#
# If JOBFLAG is set, use $jobtype; else set $jobtype to default
if ($JOBFLAG != 0) then
# Check for existence of job type
  set job = `grep $jobtype jobtypes.sendcray | awk '{print $1}'`
  if ($job == '') then
     echo 'sendcray: jobtype' $jobtype 'invalid; see jobtypes.sendcray'
     goto usagemsg
  else
     set jobtype = $job
  endif
else
  set jobtype = $DEFJOBTYPE
endif
#
# If TIMEFLAG is set, use $jobtime; else set $jobtime to default
if ($TIMEFLAG != 0) then
# Check to make sure max job time is not exceeded
  if ($jobtime > $MAXJOBTIME) then
```

```
      echo 'sendcray: jobtime' $jobtime 'exceeds max jobtime of' $MAXJOBTIME \
        'minutes'
      goto usagemsg
    endif
  else
    set jobtime = $DEFJOBTIME
  endif
  #
  # If OUTFLAG is not set, set $outputfile to default (else, continue)
  if ($OUTFLAG == 0) then
     set outputfile = $inputfile.crayout
  endif
  #****************
  # usagemsg prints a Usage message then exits
  goto endusagemsg
  usagemsg:
  echo 'Usage: sendcray [ -j job_type ] [ -t number_of_min. ]'
  echo '   [ -o output_filename ] input_filename'
  ###echo '   [ -o output_filename ] [ -d AFIT_destination_host ] input_filename'
  exit ()
  endusagemsg:
  #****************
  #
  # Create unique Cray job file identifier, using the source ID as a prefix and
  #   the current sendcray process ID as the suffix, and echo it as an online
  #   message
  set jobfile = $SOURCEID$PROCESSID
  echo 'sendcray: jobfile ID,' $jobfile', created:' \
    `date | awk '{print $4 " " $5}'`
  #
  # Check to see if local source sendcray log file exists; if it does not,
  #   create it and protect it
  if (! -e sendcray.log) then
     cp /dev/null sendcray.log ; chmod 0600 sendcray.log
     echo '## this is the local sendcray log file' > sendcray.log
     echo '## the format is:  date unique_jobfile_ID source destination jobtype'
  \
       'jobtime' >> sendcray.log
     echo '##   inputfile outputfile user inputfile_directory process_ID' \
       >> sendcray.log
  endif
  # Update sendcray.log with Cray job information
  echo `date` $jobfile $SOURCEHOST $destination $jobtype $jobtime $inputfile \
    $outputfile `whoami` `pwd` $PROCESSID >> sendcray.log
  #
  # Create Cray job log file (using job file name and .craylog suffix),
  #   give it protection, then enter Cray job information into job log file
  cp /dev/null $jobfile.craylog ; chmod 0644 $jobfile.craylog
```

```
echo '************** Cray job' $jobfile 'log file *************' \
  > $jobfile.craylog
echo $startime ': actual sendcray start time' >> $jobfile.craylog
echo  'source:    '   $SOURCEHOST    '        destination:    '  $destination  >>
$jobfile.craylog
echo 'jobtype: ' $jobtype '    jobtime: ' $jobtime >> $jobfile.craylog
echo 'inputfile: ' $inputfile >> $jobfile.craylog
echo 'outputfile: ' $outputfile >> $jobfile.craylog
echo 'user: ' `whoami` >> $jobfile.craylog
echo 'directory: ' `pwd` >> $jobfile.craylog
echo 'processid: ' $PROCESSID >> $jobfile.craylog
echo `date` ': Cray log file creation completed at' $SOURCEHOST \
  >> $jobfile.craylog
echo '*********************************************************' \
  >> $jobfile.craylog
# Echo Cray job log file creation message as an online message to the user
echo 'sendcray: log file created'
#
#*****************************************************************
#*                                                               *
#*      DATE:  December 5, 1986                                  *
#*      VERSION:  1.0                                            *
#*                                                               *
#*      MODULE NUMBER:  A21                                      *
#*      NAME:  Route to SSC                                      *
#*      FUNCTION:  Transfers Cray job file to the SSC  inbound   *
#*                 directory, updates Cray job log file,         *
#*                 and transfers the log file to the SSC         *
#*                 inbound  directory.                           *
#*      INPUTS:  None.                                           *
#*      OUTPUTS:  Status messages (on-screen)                    *
#*      GLOBALS READ:  job file identifier                       *
#*      GLOBALS WRITTEN:  None.                                  *
#*      FILES READ:  Cray job input file                         *
#*      FILES WRITTEN:  jobfile.crayin, jobfile.craylog          *
#*      HARDWARE INPUT:  None.                                   *
#*      HARDWARE OUTPUT:  AFIT/ENG Ethernet (if ftp used)        *
#*      CALLING MODULES:  None.                                  *
#*      CALLED MODULES:  None.                                   *
#*                                                               *
#*      AUTHOR:  Capt Todd Tasseff                               *
#*      HISTORY:  1.0 original version (Capt Todd Tasseff)       *
#*                                                               *
#*****************************************************************
# Transfer Cray job input and log files to  outbound  subdirectory
cp $inputfile outbound/$jobfile.crayin
mv $jobfile.craylog outbound/.
# Change directory to  outbound  subdirectory
```

System Program Listing - sendcray

```
cd outbound
#
## Change the ownership of the input and log files to the username that
##   owns the  outbound subdirectory, then switch user to that username
##chown $SYSTEMUSER outbound/$jobfile.crayin outbound/$jobfile.craylog
##exec su $SYSTEMUSER -f -c << SUEND
#
ftpsend:
# Update Cray job log file
echo `date` ': Cray job output and log files being sent to SSC' \
  >> $jobfile.craylog
echo '  from' $SOURCEHOST >> $jobfile.craylog
cp /dev/null $jobfile.ftpchk; chmod 0600 $jobfile.ftpchk
ftp ssc << ENDFTP >& $jobfile.ftpchk
cd cray/inbound
send $jobfile.crayin
send $jobfile.craylog
quit
ENDFTP
set ftp_test = `wc -w $jobfile.ftpchk | awk '{print $1}'`
if ($ftp_test) then   # ftp-send did not work out
   cat $jobfile.ftpchk >> $jobfile.craylog
   echo `date`  ':  sendcray:  ftp  send  failed  at'  $SOURCEHOST:  >>
$jobfile.craylog
   cat $jobfile.ftpchk >> $jobfile.craylog
   # Echo Cray job files transfer-failure message as online message to user
   echo 'sendcray: ftp send failed to transfer input and log files to SSC'
   echo 'sendcray: try again later'
else   # ftp-send did work out
   # Echo Cray job files transfer-success message as online message to user
   echo 'sendcray: input and log files transferred to SSC'
   # Remove both input and log files
   rm $jobfile.crayin
   rm $jobfile.craylog
endif
rm $jobfile.ftpchk  # ftp-check file no longer needed
##SUEND
#
#       *** END OF FILE sendcray ***
```

System Program Listing - send_cyber_cray

```
#!/bin/csh -f
#*********************************************************************
#*                                                                   *
#*      DATE:  December 5, 1986                                      *
#*      VERSION:  1.0                                                *
#*                                                                   *
#*      TITLE:  AFIT - ASD Cray System                               *
#*      FILENAME:  send_cyber_cray                                   *
#*      OWNER:  ttasseff (Capt Todd Tasseff, GCS-86D)                *
#*      SOFTWARE SYSTEM: AFIT SSC - 4.2 BSD Unix Operating           *
#*                       System                                      *
#*      USE:  for Lt Col Carter, EENG Thesis Project                 *
#*      CONTENTS:  Module A22, Route to Cyber/Cray                   *
#*                                                                   *
#*********************************************************************
#
#*********************************************************************
#*                                                                   *
#*      DATE:  December 5, 1986                                      *
#*      VERSION:  1.0                                                *
#*                                                                   *
#*      MODULE NUMBER:  A22                                          *
#*      NAME:  Route to Cyber/Cray                                   *
#*      FUNCTION:  Updates Cray job log file, creates a Cray         *
#*                 job job-deck, (using Cyber and Cray job-          *
#*                 control commands), sends job-deck to              *
#*                 Cyber/Cray, and updates log file                  *
#*      INPUTS:  None.                                               *
#*      OUTPUTS:  None.                                              *
#*      GLOBALS READ:  job file identifier                           *
#*      GLOBALS WRITTEN:  None.                                      *
#*      FILES READ:  jobfile.crayin, jobfile.craylog,                *
#*                   crayjobtop, crayjobbot                          *
#*      FILES WRITTEN:  jobfile.crayin.deck, jobfile.craylog         *
#*      HARDWARE INPUT:  AFIT/ENG Ethernet (if ftp used)             *
#*      HARDWARE OUTPUT:  RJE link to ASD Cyber                      *
#*      CALLING MODULES:  None.                                      *
#*      CALLED MODULES:  None.                                       *
#*                                                                   *
#*      AUTHOR:  Capt Todd Tasseff                                   *
#*      HISTORY:  1.0 original version (Capt Todd Tasseff)           *
#*                                                                   *
#*********************************************************************
set SYSTEMDIR = ~ttasseff/cray
#
cd $SYSTEMDIR
# Check to see if the send_cyber_cray.log file exists; if not, create it
if (! -e send_cyber_cray.log) then
```

```
   cp /dev/null send_cyber_cray.log; chmod 0600 send_cyber_cray.log
endif
# Update send_cyber_cray.log file
echo `date` ': send_cyber_cray initiated' >> send_cyber_cray.log
# Check to make sure that no other  send_cyber_cray  shells are running...
if (`ps | grep send_cyber_cray | wc -l` > 2) then
   echo '  send_cyber_cray already running' >> send_cyber_cray.log
   exit()  # ...if so, update send_cyber_cray.log file and exit
endif
#
# Move to  inbound  subdirectory
cd inbound
# Check for existence of Cray job log files; if none, then exit foreach loop
foreach logfile (`ls | grep '\.craylog$'`)
   # Protect Cray job log file
   chmod 0644 $logfile
   # Get jobfile name (prefix-before-the-dot) of Cray job log file
   set jobfile = `echo $logfile | awk -f. '{print $1}'`
   # Check for existence of Cray job input file; if none, then place an
   #   error message in the Cray job log file, create a dummy Cray job output
   #   file, move both output and log files to  outbound  subdirectory, and
   #   go to end of the foreach loop; else, if it exists, give it protection
   if (! -e $jobfile.crayin) then
      echo `date` ': send_cyber_cray: Cray job log file ONLY received at SSC;'
\
        >> $jobfile.craylog
      echo '  input file lost; job aborted' >> $jobfile.craylog
      cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
      echo 'send_cyber_cray: input file lost; job aborted -- see log file' \
        > $jobfile.crayout
      mv $jobfile.crayout ../outbound/.; mv $jobfile.craylog ../outbound/.
      goto endforeach
   else
      chmod 0600 $jobfile.crayin
   endif
   # Update Cray job log file
   echo `date` ': Cray job input and log files received at SSC' \
     >> $jobfile.craylog
   #
   # Check to see if user is on the sendcray valid user list; if not, place an
   #   error message in the Cray job log file, create a dummy Cray job output
   #   file, move both output and log files to  outbound  subdirectory, and
   #   go to end of the foreach loop; else, get Cyber and Cray account info
   set user = `head -10 $jobfile.craylog | awk '/user:/ {print $2}'`
   set user_info = `grep ^$user" " ../users.sendcray`
   if ($#user_info) then
      set cyberuser = $user_info[2]
      set cyberpass = $user_info[3]
```

```
      set crayacct = $user_info[4]
      set crayacpass = $user_info[5]
      set crayuser = $user_info[6]
      set crayuspass = $user_info[7]
   else
      echo `date` ': send_cyber_cray: user NOT on valid user list at SSC;' \
        >> $jobfile.craylog
      echo '  job aborted' >> $jobfile.craylog
      cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
      echo 'send_cyber_cray: user NOT on valid user list; job aborted' \
        '-- see log file' > $jobfile.crayout
      mv $jobfile.crayout ../outbound/.; mv $jobfile.craylog ../outbound/.
      goto endforeach
   endif
   # Begin forming Cray job deck -- update Cray job log file
   echo `date` ': Cray job deck being formed at SSC' >> $jobfile.craylog
   # Get Cray job type and job time from the log file
   set jobtype = `head -10 $jobfile.craylog | awk '/jobtype:/ {print $2}'`
   set jobtime = `head -10 $jobfile.craylog | awk '/jobtime:/ {print $4}'`
   switch ($jobtype)
      case fortran:
                  set jobtype = CFT,L=0
      case spice:
                  set jobtype = SPICE
      default:
                  set jobtype = CFT,L=0
   endsw
   # Create and protect a temporary file; edit top portion of Cray job deck
   #   and replace dummy variable names with the given variable names,
   #   then store it in a temporary file; also, update Cray job log file
   #   with any editing info
   cp /dev/null ../$jobfile.crayjobtop; chmod 0600 ../$jobfile.crayjobtop
ed ../crayjobtop << ENDEDIT >>& $jobfile.craylog
1,\$s/JOBFILEXXX/$jobfile/g
1,\$s/CYBERUSER/$cyberuser/g
1,\$s/CYBERPASS/$cyberpass/g
1,\$s/JOBTIMEXXX/$jobtime/g
1,\$s/CRAYACCT/$crayacct/g
1,\$s/CRAYACPASS/$crayacpass/g
1,\$s/CRAYUSER/$crayuser/g
1,\$s/CRAYUSPASS/$crayuspass/g
1,\$s/JOBTYPEXXX/$jobtype/g
w ../$jobfile.crayjobtop
q
ENDEDIT
   # Create and protect the new Cray job deck; complete new Cray job deck by
   #   concatenating the top portion of the job deck, the Cray input file,
   #   and the bottom portion of the job deck together; remove old job files
```

```
   cp /dev/null $jobfile.crayin.deck; chmod 0600 $jobfile.crayin.deck
   cat ../$jobfile.crayjobtop $jobfile.crayin ../crayjobbot > \
     $jobfile.crayin.deck
   rm ../$jobfile.crayjobtop
   rm $jobfile.crayin
   #
   # Send Cray job deck to the Cyber/Cray; update the log file as usual, and
   #   also include any system message or error message; any Cray job output
   #   will go to the  outbound  subdirectory; also, remove the job deck after
   #   it is sent
   echo `date` ': Cray job deck send-to-cyber/cray started at SSC' \
     >> $jobfile.craylog
   cp /dev/null $jobfile.sendchk; chmod 0600 $jobfile.sendchk
   /usr/afit/send -output=../outbound/$jobfile.crayout $jobfile.crayin.deck \
     >& $jobfile.sendchk
   # Check for problems with sending to Cyber/Cray; if O.K., update Cray job
   #   log file and continue; if not O.K., update and place an error message
   #   into the log file,  create  a  dummy  Cray  job  output file, move both
output
   #   and log files to  outbound  subdirectory, and go to end of the foreach
   #   loop
   set send_test = \
     `cat $jobfile.sendchk | awk '$0  ~  /Job  number/  &&  $0  ~ /entered in
queue/'`
   if ($#send_test) then
      cat $jobfile.sendchk >> $jobfile.craylog  # send to Cyber O.K.
   else
      cat $jobfile.sendchk >> $jobfile.craylog  # send to Cyber not O.K.
      echo `date` ': send_cyber_cray: send to Cyber/Cray failed at SSC;' \
        >> $jobfile.craylog
      echo '  job aborted' >> $jobfile.craylog
      cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
      echo 'send_cyber_cray: send to Cyber/Cray failed; job aborted -- see' \
        'log file' > $jobfile.crayout
      mv $jobfile.crayout ../outbound/.; mv $jobfile.craylog ../outbound/.
      goto endforeach
   endif
   rm $jobfile.crayin.deck
   rm $jobfile.sendchk
   #
   # Update Cray job log file and move it to the  outbound  subdirectory
   echo `date` ': Cray job deck send-to-cyber/cray completed at SSC' \
     >> $jobfile.craylog
   mv $jobfile.craylog ../outbound/.
endforeach:
end
#
#       *** END OF FILE send_cyber_cray ***
```

System Program Listing - send_destination

```
#!/bin/csh -f
#**********************************************************************
#*                                                                    *
#*      DATE:  December 5, 1986                                       *
#*      VERSION:  1.0                                                 *
#*                                                                    *
#*      TITLE:  AFIT - ASD Cray System                                *
#*      FILENAME:  send_destination                                   *
#*      OWNER:  ttasseff (Capt Todd Tasseff, GCS-86D)                 *
#*      SOFTWARE SYSTEM: AFIT SSC - 4.2 BSD Unix Operating            *
#*                       System                                       *
#*      USE:  for Lt Col Carter, EENG Thesis Project                  *
#*      CONTENTS:  Module A4, Route Cray Job Output to                *
#*                   Destination                                      *
#*                                                                    *
#**********************************************************************
#
#**********************************************************************
#*                                                                    *
#*      DATE:  December 5, 1986                                       *
#*      VERSION:  1.0                                                 *
#*                                                                    *
#*      MODULE NUMBER:  A4                                            *
#*      NAME:  Route Cray Job Output to Destination                   *
#*      FUNCTION:  Checks for Cray job output, updates Cray job       *
#*                   log file, determines output destination,         *
#*                   and transfers output and log files to the        *
#*                   destination host.                                *
#*      INPUTS:  None.                                                *
#*      OUTPUTS:  Onscreen or mail done-message                       *
#*      GLOBALS READ:  job file identifier                            *
#*      GLOBALS WRITTEN:  None.                                       *
#*      FILES READ:  jobfile.craylog, jobfile.crayout                 *
#*      FILES WRITTEN:  jobfile.craylog, outputfile                   *
#*      HARDWARE INPUT:  RJE link from ASD Cyber                      *
#*      HARDWARE OUTPUT:  AFIT/ENG Ethernet (if ftp used)             *
#*      CALLING MODULES:  None.                                       *
#*      CALLED MODULES:  None.                                        *
#*                                                                    *
#*      AUTHOR:  Capt Todd Tasseff                                    *
#*      HISTORY:  1.0 original version (Capt Todd Tasseff)            *
#*                Note:  lines beginning with ## refer to             *
#*                       items necessary for implementing             *
#*                       the program at root level                    *
#*                Note:  lines beginning with ### refer to            *
#*                       features not yet implemented                 *
#*                                                                    *
#**********************************************************************
```

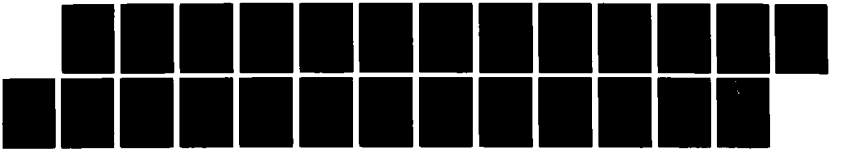
```
set SYSTEMUSER = ttasseff
set SYSTEMDIR = ~ttasseff/cray
#
cd $SYSTEMDIR
#
# Check to see if the send_destination.log file exists; if not, create it
if (! -e send_destination.log) then
   cp /dev/null send_destination.log; chmod 0600 send_destination.log
endif
# Update send_destination.log file
echo `date` ': send_destination initiated' >> send_destination.log
# Check to make sure that no other  send_destination  shells are running...
if (`ps | grep send_destination | wc -l` > 2) then
   echo '  send_destination already running' >> send_destination.log
   exit()  # ...if so, update send_destination.log file and exit
endif
#
set SYSTEMHOST = ssc
###set PASSWDFILE = /etc/passwd
# the following are based on an every-15-min. cron execution of this program
@ SEND_CHK = 4  # equal to one hour
@ OUTPUT_CHK = 8  # equal to two hours
#
# Go to  outbound  subdirectory
cd outbound
# Check for existence of Cray job log files prefixed by  sc  (for ssc); if
#   none, then exit foreach loop
foreach sc_logfile (`ls | grep 'sc.*\.craylog$'`)
  # Get jobfile name (prefix-before-the-dot) of Cray job log file
  set jobfile = `echo $sc_logfile | awk -f. '{print $1}'`
  #
  # Check for existence of Cray job output file; if not, increment Cray job
  #   counter and check for error limits based on the counter's value
  if (! -e $jobfile.crayout) then
     if (! -e $jobfile.craycounter) then  # Create counter file if not present
        cp /dev/null $jobfile.craycounter; chmod 0600 $jobfile.craycounter
        echo '0' > $jobfile.craycounter  # Set counter to zero
     endif
     # Get counter value from craycounter file and increment it by 1
     set counter = `awk '{print $1}' $jobfile.craycounter`
     @ counter++
     if ($counter == $SEND_CHK) then  # Check for send limit
        set send_test = `/usr/afit/send | grep $jobfile.crayin.deck`
        if ($#send_test) then  # Cray job input file/deck still in send queue
           echo `date` ': send_destination: send to Cyber/Cray timeout' \
             >> $jobfile.craylog  # Update Cray job log file
           echo '  exceeded; SSC not sending to Cyber; job aborted' \
             >> $jobfile.craylog
```

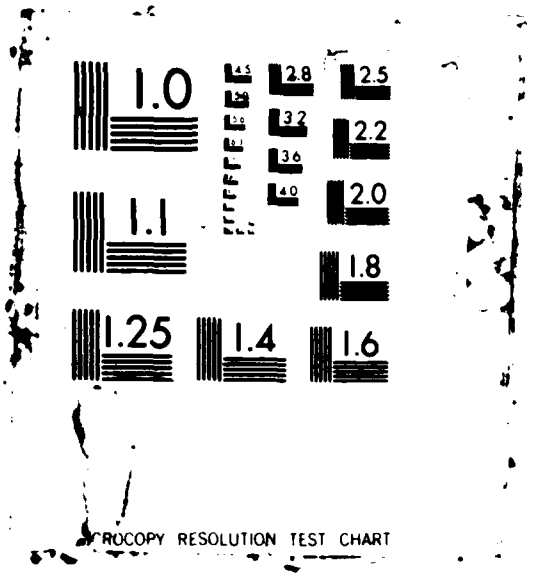
```
            # Create dummy output file
            cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
            echo 'send_destination: file not sent to Cyber Cray; job aborted'
              '-- see log file' > $jobfile.crayout
            rm $jobfile.craycounter  # Remove counter file
            goto endcrayoutchk  # Go to end of Cray job output file check
          endif
        else if ($counter >= $OUTPUT_CHK) then  # Check for output limit
          echo `date` ': send_destination: Cray job output timeout exceeded;' \
            >> $jobfile.craylog  # Update Cray job log file
          echo '   output   file   lost   at   Cyber/Cray;   job   aborted' >>
$jobfile.craylog
          # Create dummy output file
          cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
          echo 'send_destination: output file lost at Cyber/Cray; job aborted' \
            '-- see log file' > $jobfile.crayout
          rm $jobfile.craycounter  # Remove counter file
          goto endcrayoutchk  # Go to end of Cray job output file check
        else
          echo $counter > $jobfile.craycounter
          goto sc_endforeach  # Go to end of foreach loop
        endif
      else
        # If Cray job output file exists, protect it and update log file
        chmod 0600 $jobfile.crayout
        echo `date` ':   Cray   job   output   received   at'   $SYSTEMHOST  >>
$jobfile.craylog
        if (-e $jobfile.craycounter) then
          rm $jobfile.craycounter  # If counter file exists, remove it
        endif
      endif
    endcrayoutchk:
      #
      # Get user name, source, and destination for Cray job output
      set user = `head -10 $jobfile.craylog | awk '/user:/ {print $2}'`
      set source = \
        `head -10 $jobfile.craylog | awk '/source:/ {print $2}'`
      set destination = \
        `head -10 $jobfile.craylog | awk '/destination:/ {print $4}'`
      # Transfer Cray job output and log files to the user
      ###if ({ grep -s "^"$user":" $PASSWDFILE } == 0) then
      ###   # If user does not have an account on this system host...
      ###   echo `date` 'send_destination: user' $user 'not valid on' \
      ###     $SYSTEMHOST';' >> $jobfile.craylog
      ###   echo '  sending output to source host' >> $jobfile.craylog
      ###   set destination = $source
      ###   #  go to some ftp-send routine within this foreach loop...
      ###else   # If user does have an account on this system host...
```

```
        ## Get directory and output file names from Cray job log file
        set directory = \
          `head -10 $jobfile.craylog | awk '/directory:/ {print $2}'`
        set outputfile = \
          `head -10 $jobfile.craylog | awk '/outputfile:/ {print $2}'`
        # Transfer the output file to the user's directory
        mv $jobfile.crayout $directory/$outputfile
        # Update Cray job log file
        echo `date` ': Cray job output and log files sent to directory' \
          >> $jobfile.craylog
        echo ' ' $directory ' and user notified at' $SYSTEMHOST \
          >> $jobfile.craylog
        # Protect the log file then transfer it to the user's directory
        chmod 0600 $jobfile.craylog
        mv $jobfile.craylog $directory/$outputfile.craylog
        ## Change ownership of Cray job log and output files to user
        /etc/chown $user $directory/$outputfile $directory/$outputfile.craylog
        # Check to see if user is logged on...
        set onlinetest = `who | grep "^$user "`
        if ($#onlinetest > 1) then   # ...if so, write message
write $user << WRITEND
Message from sendcray: Cray output and log files for $jobfile have returned
WRITEND
        else   # ...if not, mail message
mail $user << MAILEND
Message from sendcray: Cray output and log files for $jobfile have returned
MAILEND
        endif
        # If any extraneous  out  files leftover from mail exist, remove them
        if (-e out) then
           rm out
        endif
  ###   # If source is not this system host, send mail to user at source
  ###   if ($source != $SYSTEMHOST) then
  ###mail $user << MAILEND
  ###Message from  sendcray:  Cray  output  and  log  files  for $jobfile have
returned
  ###MAILEND
  ###   endif
  ###endif
sc_endforeach:
end
#
#
# Check for existence of stray Cray job output files; if none, then exit
#   the foreach loop
foreach outfile (`ls | grep '\.crayout$'`)
  # Get jobfile name (prefix-before-the-dot) of Cray job output file
```

AD-A179 577 VIRTUAL COMMUNICATIONS TO ASD'S (AERONAUTICAL SYSTEMS DIVISION'S) CRAY CO..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. T W TASSEFF 2/2

UNCLASSIFIED DEC 86 AFIT/GCS/ENG/86D-10 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
ROCOPY RESOLUTION TEST CHART

```
  set jobfile = `echo $outfile | awk -f. '{print $1}'`
  #
  # Check for existence of the companion Cray job log file; if not, then
  #   update the sendcray.log and remove the stray output file
  if (! -e $jobfile.craylog) then
     echo `date` ': send_destination: stray Cray job output file' \
       >> ../sendcray.log
     echo ' ' $outfile 'removed from outbound subdirectory' >> ../sendcray.log
     rm $outfile
  endif
end
#
#
## Switch user to the username who owns this  outbound  subdirectory
exec su $SYSTEMUSER -f -c << SUEND
#
# Check for existence of other Cray job log files; if none, then exit this
#   foreach loop
foreach logfile (`ls | grep '\.craylog$'`)
  # Just in case an  sc  logfile sneaked in, go to end of this foreach loop
  set sc_log_test = `echo $logfile | grep '^sc'`
  if ($#sc_log_test) then
    goto endforeach
  endif
  # Get jobfile name (prefix-before-the-dot) of Cray job log file
  set jobfile = `echo $logfile | awk -f. '{print $1}'`
  #
  # Check for existence of Cray job output file; if not, increment Cray job
  #   counter and check for error limits based on the counter's value
  if (! -e $jobfile.crayout) then
     if (! -e $jobfile.craycounter) then  # Create counter file if not present
        cp /dev/null $jobfile.craycounter; chmod 0600 $jobfile.craycounter
        echo '0' > $jobfile.craycounter  # Set counter to zero
     endif
     # Get counter value from craycounter file and increment it by 1
     set counter = `awk '{print $1}' $jobfile.craycounter`
     @ counter++
     if ($counter == $SEND_CHK) then  # Check for send limit
        set send_test = `/usr/afit/send | grep $jobfile.crayin.deck`
        if ($#send_test) then  # Cray job input file/deck still in send queue
           echo `date` ': send_destination: send to Cyber/Cray timeout' \
             >> $jobfile.craylog  # Update Cray job log file
           echo '  exceeded; SSC not sending to Cyber; job aborted' \
             >> $jobfile.craylog
           # Create dummy output file
           cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
           echo 'send_destination: file not sent to Cyber/Cray; job aborted' \
             '-- see log file' > $jobfile.crayout
```

```
            rm $jobfile.craycounter  # Remove counter file
            goto endcrayoutchk  # Go to end of Cray job output file check
         endif
      else if ($counter >= $OUTPUT_CHK) then  # Check for output limit
         echo `date` ': send_destination: Cray job output timeout exceeded;' \
           >> $jobfile.craylog  # Update Cray job log file
         echo '   output   file   lost   at   Cyber/Cray;   job   aborted' >>
$jobfile.craylog
         # Create dummy output file
         cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
         echo 'send_destination: output file lost at Cyber/Cray; job aborted' \
           '-- see log file' > $jobfile.crayout
         rm $jobfile.craycounter  # Remove counter file
         goto endcrayoutchk  # Go to end of Cray job output file check
      else
         echo $counter > $jobfile.craycounter
         goto endforeach  # Go to end of foreach loop
      endif
   else
      # If Cray job output file exists, protect it and update log file
      chmod 0600 $jobfile.crayout
      echo `date` ':   Cray   job   output   received   at'   $SYSTEMHOST  >>
$jobfile.craylog
      if (-e $jobfile.craycounter) then
         rm $jobfile.craycounter  # If counter file exists, remove it
      endif
   endif
endcrayoutchk:
   #
   # Get user name, source, and destination for Cray job output
   set user = `head -10 $jobfile.craylog | awk '/user:/ {print $2}'`
   set source = \
     `head -10 $jobfile.craylog | awk '/source:/ {print $2}'`
   set destination = \
     `head -10 $jobfile.craylog | awk '/destination:/ {print $4}'`
   # Transfer Cray job output and log files to destination host
ftpsend:
   # Update Cray job log file
   echo `date` ': Cray job output and log files being ' >> $jobfile.craylog
   echo '  sent to' $destination 'from' $SYSTEMHOST >> $jobfile.craylog
   cp /dev/null $jobfile.ftpchk; chmod 0666 $jobfile.ftpchk
ftp $destination << ENDFTP >& $jobfile.ftpchk
cd cray/inbound
send $jobfile.crayout
send $jobfile.craylog
quit
ENDFTP
   set ftp_test = `wc -w $jobfile.ftpchk | awk '{print $1}'`
```

```
  if ($ftp_test) then    # ftp-send did not work out
     echo `date` ': send_destination: ftp send failed at' $SYSTEMHOST':' \
       >> $jobfile.craylog
     cat $jobfile.ftpchk >> $jobfile.craylog
     echo 'send_destination:' $SYSTEMHOST 'will retry ftp later' \
       >> $jobfile.craylog
  else    # ftp-send did work out
     rm $jobfile.crayout
     rm $jobfile.craylog
  endif
  ##rm $jobfile.ftpchk  # ftp-check file no longer needed
endforeach:
end
# End of switch user
SUEND
#
#***    END OF FILE send_destination    ***
```

System Program Listing - send_user

```
#!/bin/csh -f
#*********************************************************************
#*                                                                   *
#*      DATE:  December 5, 1986                                      *
#*      VERSION:  1.0                                                *
#*                                                                   *
#*      TITLE:  AFIT - ASD Cray System                               *
#*      FILENAME:  send_user                                         *
#*      OWNER:  ttasseff (Capt Todd Tasseff, GCS-86D)                *
#*      SOFTWARE SYSTEM: AFIT/ENG Sun - 4.2 BSD Unix                 *
#*                       Operating System                            *
#*      USE:  for Lt Col Carter, EENG Thesis Project                 *
#*      CONTENTS:  Module A4, Route Cray Job Output to               *
#*                   Destination                                     *
#*                                                                   *
#*********************************************************************
#
#*********************************************************************
#*                                                                   *
#*      DATE:  December 5, 1986                                      *
#*      VERSION:  1.0                                                *
#*                                                                   *
#*      MODULE NUMBER:  A4                                           *
#*      NAME:  Route Cray Job Output to Destination                  *
#*      FUNCTION:  Checks for Cray job output, updates Cray job      *
#*                   log file, determines user directory,            *
#*                   and transfers output and log files to the       *
#*                   user directory.                                 *
#*      INPUTS:  None.                                               *
#*      OUTPUTS:  Onscreen or mail done-message                      *
#*      GLOBALS READ:  job file identifier                           *
#*      GLOBALS WRITTEN:  None.                                      *
#*      FILES READ:  jobfile.craylog, jobfile.crayout                *
#*      FILES WRITTEN:  jobfile.craylog, outputfile                  *
#*      HARDWARE INPUT:  AFIT/ENG Ethernet (using ftp)               *
#*      HARDWARE OUTPUT:  None.                                      *
#*      CALLING MODULES:  None.                                      *
#*      CALLED MODULES:  None.                                       *
#*                                                                   *
#*      AUTHOR:  Capt Todd Tasseff                                   *
#*      HISTORY:  1.0 original version (Capt Todd Tasseff)           *
#*                Note:  lines beginning with ## refer to            *
#*                       items necessary for implementing            *
#*                       the program at root level                   *
#*                                                                   *
#*********************************************************************
set SYSTEMDIR = ~ttasseff/cray
#
```

```
cd $SYSTEMDIR
#
# Check to see if the send_user.log file exists; if not, create it
if (! -e send_user.log) then
   cp /dev/null send_user.log; chmod 0600 send_user.log
endif
# Update send_user.log file
echo `date` ': send_user initiated' >> send_user.log
# Check to make sure that no other  send_user  shells are running...
if (`ps | grep send_user | wc -l` > 2) then
   echo '  send_user already running' >> send_user.log
   exit()  # ...if so, update send_user.log file and exit
endif
#
set SYSTEMHOST = mercury
#
# Go to  outbound subdirectory
cd inbound
# Check for existence of Cray job log files; if none, then exit foreach loop
foreach logfile (`ls | grep '\.craylog$'`)
  # Protect Cray job log file
  chmod 0644 $logfile
  # Get jobfile name (prefix-before-the-dot) of Cray job log file
  set jobfile = `echo $logfile | awk -f. '{print $1}'`
  #
  # Check for existence of Cray job output file; if none, then place an error
  #   message in the Cray job log file, and create a dummy Cray job output
  #   file; else, if it exists, give it protection
  if (! -e $jobfile.crayout) then
     echo `date` ': send_user: Cray job log file ONLY received' \
       >> $jobfile.craylog
     echo '  at' $SYSTEMHOST'; output file lost; job aborted' \
       >> $jobfile.craylog
     # Create dummy output file
     cp /dev/null $jobfile.crayout; chmod 0600 $jobfile.crayout
     echo 'send_user: output file lost; job aborted -- see log file' \
       > $jobfile.crayout
  else
     # If Cray job output file exists, protect it and update log file
     chmod 0600 $jobfile.crayout
     echo `date` ': Cray job output and log files received at' $SYSTEMHOST \
       >> $jobfile.craylog
  endif
  #
  ## Get directory and outputfile name from the Cray job log file
  set directory = \
    `head -10 $jobfile.craylog | awk '/directory:/ {print $2}'`
  set outputfile = \
```

```
    `head -10 $jobfile.craylog | awk '/outputfile:/ {print $2}'`
  # Transfer the output file to the user's directory
  mv $jobfile.crayout $directory/$outputfile
  # Update Cray job log file
  echo `date` ': Cray job output and log files sent to directory' \
    >> $jobfile.craylog
  echo ' ' $directory ' and user notified at' $SYSTEMHOST >> $jobfile.craylog
  # Protect the log file then transfer it to the user's directory
  chmod 0600 $jobfile.craylog
  mv $jobfile.craylog $directory/$outputfile.craylog
  ## Change ownership of Cray job log and output files to user
  /etc/chown $user $directory/$outputfile $directory/$outputfile.craylog
  # Check to see if user is logged on...
  set onlinetest = `who | grep "^$user "`
  if ($#onlinetest > 1) then    # ...if so, write message
write $user << WRITEND
Message from sendcray: Cray output and log files for $jobfile have returned
WRITEND
  else    # ...if not, mail message
mail $user << MAILEND
Message from sendcray: Cray output and log files for $jobfile have returned
MAILEND
  endif
endforeach:
end
#
#       *** END OF FILE send_user ***
```

NAME

sendcray - send a job to ASD's Cray Computer via AFIT data communications resources

SYNOPSIS

```
sendcray [ -j job_type ] [ -t number_of_min. ]
         [ -o output_filename] input_filename
```

DESCRIPTION

Sendcray is the initiating command of the AFIT-ASD Cray System. In general, the AFIT-ASD Cray System operates over the currently available AFIT data communications facility, which is connected to ASD's data communications facility. Specifically, the System makes use of the existing RJE link between the AFIT SSC and ASD's CDC Cyber computer, which is in turn connected to the Cray. The System also communicates with AFIT/ENG's ELXSI computer and Sun workstations and the AFIT SSC via a dedicated, coaxial cable, local computer network link called Ethernet.

Input data files created on the ELXSI or the Sun workstations and destined for ASD's Cray computer will be automatically transmitted over the Ethernet link to a special System account on the SSC. Input data files created on the SSC are transferred internally to the System's account.

Once at the SSC, the ELXSI, Sun or SSC input data file destined for the Cray will have the appropriate Cyber and Cray job-control commands placed into the input data file, forming a job deck. The SSC will then take the Cray job deck and transmit it to the Cyber via the SSC-Cyber RJE link. Once at the Cyber, the job file will be transferred to the Cray, the job processed, and the output returned to the Cyber, which in turn returns the output to the AFIT SSC.

Once the SSC receives the Cray output from the Cyber, the SSC stores it or passes it on to the ELXSI or Sun workstations (using the Ethernet), depending on where the job originated. The output will then be placed into the user directory where the job originated.

The System will notify the user that the output has returned via an on-screen message at the computer where the job originated. If the user has logged-off the computer, the System will place a message in the user's computer "mailbox."

The System also has an audit capability which keeps track of the progress of a job as it passes from one computer to the next. A log file accompanies both the input file and corresponding output file through the System. The System then sends to the user the log file along with the output file. The log file contains date/time stamps of important events (e.g., file transfer from one computer to the next) and any pertinent System messages and error

messages sent by the AFIT and ASD host computers involved.

The user has the responsibility to first create an input file for the AFIT-ASD Cray System. This is accomplished by using the file editor of choice on any one of the AFIT host computers or workstations involved. Once the input file is created, the user may enter the sendcray command followed by a set of options, and ending with the input filename. If no options are given, then the default options are assumed.

Sendcray provides the user with the following options:

-j Cray job type. The job type can be either FORTRAN or SPICE. A FORTRAN job indicates that the input data file is a FORTRAN program, while a SPICE job indicates that the input data file is input data for a VLSI circuit simulation. The default Cray job type is FORTRAN.

-t Cray job time. The job time is the estimated time in minutes that the user expects the Cray job to run while at the Cray. The System uses this time value to place an upper limit on how long ASD's Cyber should wait for output to return from the Cray, the maximum time being 15 minutes. The default Cray job time is 5 minutes. Caution should be taken when using this option (see Bugs section below).

-o Output filename. The user can also specify the exact name of the Cray job's output filename. However, if no output filename is given, the default output filename will be the same as the input filename followed by a ".crayout" suffix. Accompanying the output file will be the log file for the Cray job. The name of the log file will be the same as the specified output filename followed by a ".craylog" suffix, with the default being the input filename followed by that same suffix.

INSTRUCTIONS

(1) Getting User Accounts and Passwords. The user must first obtain user accounts on the AFIT SSC and on ASD's Cyber and Cray computers by asking for and submitting an AFIT Form 35 to AFIT/SIOO personnel in Bldg 640, Room 133. The user will then be assigned an SSC username and password, a Cyber account (user) number with an initial interactive (login) password, and a Cray account number with an initial password. Along with the Cray account number and password are assigned an identical Cray user number and password. Once the user has these accounts and passwords, the user should change the passwords to protect these user accounts. The user must then create a batch password on the Cyber by logging-in to the Cyber and using the PASSWOR command to create a separate batch password.

The user must then submit the AFIT SSC username, Cyber user number and

batch password, Cray account number and password, and Cray user number and password to the AFIT-ASD Cray System administrator in AFIT/SIO in Bldg 125. The System administrator will load the SSC username and Cyber and Cray account numbers and passwords into the proper control file on the AFIT SSC so that the user will have access to the System. Note: a username on the AFIT SSC is not necessarily required for the System to operate, but is helpful in establishing a unique username among the AFIT host computers involved in the System (the AFIT SSC and AFIT/ENG's ELXSI and Suns). If the user wishes to obtain a username on AFIT/ENG's ELXSI and Suns, the username should be identical to the username assigned for the SSC.

(2) Creating an Input File. Next, the user must create an input file on the AFIT host computer of choice using the text editor of choice. Caution should be taken not to include any blank lines in any of the input files (see Bugs section below).

(a) If the input file is to be FORTRAN, the user should create an input file containing the FORTRAN source code. If any input data is used, it must follow the source code; however, each input data section must be preceded by a single line containing "/EOF" (no quotation marks used) which must begin in the first column of that line. A "/EOF" line must separate the source code and each input data section from each other (no "/EOF" lines required at the beginning or end of entire input file).

(b) If the input file is for use with SPICE, the user should create an input file containing VLSI circuit simulation data as specified and required by the SPICE program.

(3) Using sendcray. While logged-in to the AFIT host where the input file resides, enter the command sendcray followed by the desired options followed by the input filename, all on one line. After entering the command line, sendcray will send on-screen messages to the user, indicating what action is currently being done. One important item to note is the jobfile ID number that sendcray issues and displays in one of these messages. This jobfile ID is unique for each job and can be used to track the progress of each job throughout the System. By checking the contents of the System subdirectories, inbound and outbound, the user can find out how far along the job has progressed. All job files used by the System are prefixed by the jobfile ID number. The user can tell, by inspection, the status of the job by seeing if files prefixed by the jobfile ID exist, and can also get more detailed status information by reading the contents of the job log file. The job log filename is prefixed by the jobfile ID number and suffixed by ".craylog". When the job output returns to the user, the output file and log file are placed in the same AFIT host and directory (file-space) where the user originated the job, and an appropriate "done" message (with jobfile ID number included) is issued (on-screen if user is logged-in, to the "mailbox" if not).

DIRECTORIES

```
~cray  # main System account directory on each AFIT host
~cray/inbound
~cray/outbound
```

SEE ALSO

```
ftp             # file transfer (over a network) program
/usr/afit/send  # send jobs to ASD's Cyber
```

BUGS

At present, SPICE is not implemented on the Cray. The System will notify the user that this is the case if SPICE is chosen as a job type (using the -j option).

Caution should be taken in setting the job time (-t) option. If the job runs on the Cray longer than the job time specified, the Cray output will be lost to the System. However, the output will be left at the Cyber, so the user may retrieve it from there if desired.

Caution should also be taken not to include any blank lines when creating any of the input files. Blank lines within the input files may cause ASD's Cyber or Cray to abort the job, returning only an error listing as output.

Also, the alternate output destination option (-d) is not implemented yet. The user would have been able to specify which AFIT host computer or workstation would be the final destination of a Cray job's output. The choices would be the AFIT SSC, AFIT/ENG's ELXSI, or any one of AFIT/ENG's Sun workstations. The default output destination would still be the source computer or workstation (the machine that originated the Cray job).

AUTHOR

Todd W. Tasseff (Dec. 1986)

1.0 INTRODUCTION

(Note: see also Sendcray User Documentation for additional information)

1.1 Command Name

sendcray - send a job to ASD's Cray Computer via AFIT data communications resources

1.2 Command Line and Options

```
sendcray [ -j job_type ] [ -t number_of_min. ]
         [ -o output_filename] input_filename
```

1.3 Functional Description

1.3.1 System Operation. sendcray is the initiating command of the AFIT-ASD Cray System. In general, the AFIT-ASD Cray System operates over the currently available AFIT data communications facility, which is connected to ASD's data communications facility. Specifically, the System makes use of the existing RJE link between the AFIT SSC and ASD's CDC Cyber computer, which is in turn connected to the Cray. The System also communicates with AFIT/ENG's ELXSI computer and Sun workstations and the AFIT SSC via a dedicated, coaxial cable, local computer network link called Ethernet. All of the AFIT-ASD Cray System programs are written in 4.2 BSD UNIX C-shell commands.

Input data files created on the ELXSI or the Sun workstations and destined for ASD's Cray computer will be automatically transmitted over the Ethernet link to a special System account on the SSC. Input data files created on the SSC are transferred internally to the System's account.

Once at the SSC, the ELXSI, Sun or SSC input data file destined for the Cray will have the appropriate Cyber and Cray job-control commands placed into the input data file, forming a job deck. The SSC will then take the Cray job deck and transmit it to the Cyber via the SSC-Cyber RJE link. Once at the Cyber, the job file will be transferred to the Cray, the job processed, and the output returned to the Cyber, which in turn returns the output to the AFIT SSC.

Once the SSC receives the Cray output from the Cyber, the SSC stores it or passes it on to the ELXSI or Sun workstations (using the Ethernet), depending on where the job originated. The output will then be placed into the user directory where the job originated.

The System will notify the user that the output has returned via an on-screen message at the computer where the job originated. If the user has logged-off the computer, the System will place a message in the user's computer "mailbox."

The System also has an audit capability which keeps track of the progress of a job as it passes from one computer to the next. A log file accompanies both the input file and corresponding output file through the System. The System then sends to the user the log file along with the output file. The log file contains date/time stamps of important events (e.g., file transfer from one computer to the next) and any pertinent System messages and error messages sent by the AFIT and ASD host computers involved.

The user has the responsibility to first create an input file for the AFIT-ASD Cray System. This is accomplished by using the file editor of choice on any one of the AFIT host computers or workstations involved. Once the input file is created, the user may enter the sendcray command followed by a set of options, and ending with the input filename. If no options are given, then the default options are assumed.

1.3.2 System Options. sendcray provides the user with the following options:

-j Cray job type. The job type can be either FORTRAN or SPICE. A FORTRAN job indicates that the input data file is a FORTRAN program, while a SPICE job indicates that the input data file is input data for a VLSI circuit simulation. The default Cray job type is FORTRAN.

-t Cray job time. The job time is the estimated time in minutes that the user expects the Cray job to run while at the Cray. The System uses this time value to place an upper limit on how long ASD's Cyber should wait for output to return from the Cray, the maximum time being 15 minutes. The default Cray job time is 5 minutes. Caution should be taken when using this option (see Bugs section below).

-o Output filename. The user can also specify the exact name of the Cray job's output filename. However, if no output filename is given, the default output filename will be the same as the input filename followed by a ".crayout" suffix. Accompanying the output file will be the log file for the Cray job. The name of the log file will be the same as the specified output filename followed by a ".craylog" suffix, with the default being the input filename followed by that same suffix.

2.0 DIRECTORIES USED

~cray

This is the main System account directory on each AFIT host computer involved (AFIT SSC, AFIT/ENG's ELXSI and Suns). It contains all the programs and support files necessary to operate the System on each host. For the sake of simplicity, ~cray will represent the System account, or username, throughout this document.

~cray/inbound

The inbound subdirectory is used by the System to receive incoming job files from each of the AFIT hosts involved. On the AFIT SSC, it is used by the send_cyber_cray program; on non-SSC AFIT hosts, it is used by the send_user program.

~cray/outbound

The outbound subdirectory is used by the System to dispatch outgoing job files to each of the AFIT hosts involved. On the AFIT SSC, it is used by the send_destination program; on non-SSC AFIT hosts, it is used by the sendcray program.

3.0 FILES USED

~cray/sendcray.log

The sendcray.log file appears on all AFIT hosts involved and is created and used by the sendcray command to record the initiation of every sendcray job. Pertinent job information such as date/time, username, job type, and other items are recorded on a single entry in the sendcray.log. The sendcray.log file is also used to record any other System messages that the System needs to record.

~cray/.netrc

The .netrc file appears on all AFIT hosts involved and contains the usernames and passwords of the System accounts on the other AFIT host computers. Only those usernames and passwords of the other AFIT host that will have files sent to it from the one AFIT host are included. On the AFIT SSC, .netrc is used by the send_destination program; on non-SSC AFIT hosts, it is used by the sendcray program.

~cray/jobtypes.sendcray

The jobtypes.sendcray file appears on all AFIT hosts involved and contains only the job types allowed by the System, including alternate representations of the same job type (relevant combinations of upper and lower case spellings). Presently, the only two job type options are FORTRAN and SPICE. The jobtypes.sendcray file is used by the sendcray program on all of the AFIT hosts involved.

~cray/hosts.sendcray

The hosts.sendcray file appears on all AFIT hosts involved and contains only the AFIT hosts that have access to the System, including aliases (alternative names) for each host. The hosts.sendcray file has the potential of being used by the sendcray program on all the AFIT hosts involved (see Bugs section
below).

~cray/users.sendcray

The users.sendcray file appears on the AFIT SSC only and contains only those users that have access to the System. Each user has one entry in the file which contains the user's SSC (or ELXSI or Sun) username, Cyber user number and batch password, Cray account number and password, and Cray user number and password. The users.sendcray file is used by the send_cyber_cray program on the AFIT SSC.

~cray/crayjobtop
~cray/crayjobbot

The crayjobtop and crayjobbot files appear on the AFIT SSC only. Together, these two files comprise a "dummy" job deck which contains a number of "dummy" variable names which represent such items as job type, job time, Cyber and Cray account numbers and passwords, and jobfile (job work file) names. These "dummy" variable names are contained in crayjobtop and are replaced by the real things in order to begin forming a working job deck. The crayjobbot file contains only end-of-file trailer lines. The user's input file is placed between the completed crayjobtop file and the crayjobbot file in order to form a complete, working job deck that is sent to the Cyber and Cray for processing. The crayjobtop and crayjobbot files are used by the send_cyber_cray program on the AFIT SSC.

4.0 SYSTEM PROGRAMS - DESCRIPTION AND OPERATION

4.1 ~cray/sendcray. The sendcray program resides on all AFIT hosts involved and is the initial program for the System. Sendcray accepts the user's input options and Cray input filename (see User Documentation), checks for input

errors, creates a Cray job log file, and transfers the input and log files to the AFIT SSC. If sendcray is initiated at the SSC, the transfer takes place via a copy and a move command. If sendcray is initiated by any other AFIT host, the transfer takes place via the ftp command across the AFIT/ENG Ethernet to the SSC. Sendcray displays several messages to the user during its execution to include providing a unique job identifier name which is used as a prefix to the Cray input, log, and output files throughout the life of a job.

4.2 ~cray/send_cyber_cray. The send_cyber_cray program resides on the AFIT SSC only and picks up where sendcray leaves off. Send_cyber_cray receives the Cray job input and log files, packages the input file into a job deck, using information from the log file, then sends the job deck to ASD's Cyber where it is forwarded to the Cray. Output from the Cray returns to the Cyber and in turn returns to the AFIT SSC. Send_cyber_cray runs automatically via the cron facility on the SSC on a preset schedule (e.g., every 15 minutes on the quarter hour).

4.3 ~cray/send_destination. The send_destination program also resides on the AFIT SSC only. Send_destination receives the Cray job output file from the Cyber/Cray, determines the output destination from the Cray job log file, then transfers the output and log files to the proper destination. If the destination is the SSC, the transfer is accomplished via a move command directly to the user's directory (file-space). Then the user is notified, via onscreen message or mail, that the output has returned. If the destination is not the SSC, the transfer is accomplished via the ftp command directly to another System directory at the AFIT destination host. Send_destination also runs automatically via the cron facility on the SSC on the same preset schedule.

4.4 ~cray/send_user. The send_user program resides on all non-SSC AFIT hosts, and is similar to send_destination in that it also transfers the Cray job output and log files to the user's directory but does so after receiving the files from the SSC. Once the output and log files are at the user's directory, the user is notified via onscreen message or mail. Send_user also runs automatically via the cron facility on whatever host it resides, usually at a faster rate (e.g., every five minutes).

4.5 The Cray Job Log File. The Cray job log file accompanies both the Cray job input and output files throughout the life of a job. The log file contains important information about the Cray job: unique Cray job identifier, source and destination host, job type and time, input and output filenames, the user's name and current file directory, and the process ID during when sendcray was executing. The Cray job log file is also updated with a date/time stamp and a message for every significant event that occurs in each of the above programs, including any error messages (see Section 6 below).

5.0 SYSTEM SETUP

5.1 BASIC SETUP

The first step in setting up the AFIT-ASD Cray System is to create identical System usernames (accounts) and passwords on each AFIT host computer or workstation involved. The second step is to create two subdirectories named "inbound" and "outbound" which will be used for passing Cray job input, log, and output files between hosts. Note that all System directories and subdirectories must have read-permission so that the contents of the Cray job log file (mentioned in Section 4.5) can be inspected by the user during any stage in the life of a job (see Section 6).

The third step is to create a .netrc file containing the System usernames and passwords of the other AFIT hosts (for the SSC: all other hosts' usernames and passwords needed; for non-SSC hosts: username and password of SSC needed). The .netrc file is used by the ftp command within the System programs to automatically login to a remote host.

The fourth step is to place the System support files and programs mentioned above in the System accounts of their respective hosts. The System programs are suffixed by a ".src" (source) designator. Each ".src" file needs to be copied into a file with the same name as the program (minus the ".src" suffix) and edited to remove all comment lines (those that begin with a "#") in order to speed up program execution. The commands to create and update a ".log" file in the send_cyber_cray, send_destination, and send_user programs may also be removed.

Once the designated lines are removed from the programs, several variable names, located toward the beginning of each module within the programs, need to be set. Each of the System programs have one or both of the SYSTEMUSER or SYSTEMDIR variables and must be set to the the System username and directory (full path) respectively. In the sendcray program, the SOURCEHOST variable needs to be set to the name of the host on which that particular sendcray resides, based on the entries found in the hosts.sendcray file; and the SOURCEID variable needs to be set to some unique two-letter host ID (sc for SSC; el for ELXSI; mc for Sun-Mercury; etc.). In the send_destination and send_user programs, the SYSTEMHOST variable needs to be set to the host on which the program resides.

The fifth step is to establish an entry in the UNIX crontab file of every host for each System program requiring automatic execution via cron (send_cyber_cray and send_destination on the SSC; send_user on all non-SSC hosts). The cron execution time for these programs is at the discretion of the System administrator, but should be in the range of every 5 to 15 minutes.

Examples of the System support files mentioned above can be found at the

end of this document.

5.2 ESTABLISHING USERS ON THE SYSTEM

Users desiring access to the AFIT-ASD Cray System must furnish the System administrator with a [SSC] username, an ASD Cyber user number and password, an ASD Cray account and password and Cray user number and password. These user accounts and passwords must then be entered, in the order mentioned above, on a single line in the users.sendcray file.

6.0 SYSTEM DIAGNOSTICS

The primary diagnostic feature or features of the AFIT-ASD Cray System are the use of log files to record significant System events or error messages. The sendcray.log file records information pertaining to a Cray job each time a job is initiated via the sendcray program; the sendcray.log file also records a message in the event a stray output file returns from the Cyber/Cray (after a System timeout occurred and the Cray job log file was returned to the user) and is removed. Each of the other System programs have the option of creating and updating their own ".log" files containing a date/time stamp and message for each time the program is initiated.

Probably the most effective diagnostic tool is the Cray job log file which accompanies the job input and output files throughout the life of a job in the System. Besides a date/time stamp and System message for significant events (e.g., transfer of files from one host to the next), the Cray job log file contains any error messages incurred during the life of the job. These error messages indicate problems with: ftp transfers between hosts; the send command while attempting to submit job decks to the Cyber/Cray; and delayed or lost files within the system.

The user can track the progress of a job by matching the unique Cray job identifier, provided in an onscreen message by the sendcray program, with the prefix of a file ending with ".craylog" in any of the System's "inbound" and "outbound" subdirectories. This ".craylog" file is the Cray job log file, and has read-permission so that the user or System administrator can inspect its contents and determine the status of the job.

7.0 BUGS

At present, SPICE is not implemented on the Cray. The System will notify the user that this is the case if SPICE is chosen as a job type (using the -j option).

Caution should be taken in setting the job time (-t) option. If the job runs on the Cray longer than the job time specified, the Cray output will be

lost to the System. However, the output will be left at the Cyber, so the user may retrieve it from there if desired.

Caution should also be taken not to include any blank lines when creating any of the input files. Blank lines within the input files may cause ASD's Cyber or Cray to abort the job, returning only an error listing as output.

Also, the alternate output destination option (-d) is not implemented yet. The user would have been able to specify which AFIT host computer or workstation would be the final destination of a Cray job's output. The choices would be the AFIT SSC, AFIT/ENG's ELXSI, or any one of AFIT/ENG's Sun workstations. The default output destination would still be the source computer or workstation (the machine that originated the Cray job).

8.0 AUTHOR

Todd W. Tasseff (Dec. 1986)

EXAMPLES OF SYSTEM SUPPORT FILES

```
::::::::::::::
sendcray.log
::::::::::::::
## this is the local sendcray log file
## the format is:  date unique_jobfile_ID source destination jobtype jobtime
##   inputfile outputfile user inputfile_directory process_ID
Thu Nov 6 19:04:27 EST 1986 sc16349 ssc ssc fortran 5 craytestjob
craytestjob.crayout ttasseff /en0/gcs86d/ttasseff/cray 16349


::::::::::::::
example of .netrc file
::::::::::::::
On the SSC...
machine zelxb, login ttasseff, password xxxxxxx
machine vsun2-1, login ttasseff, password xxxxxxx

On the Sun-Mercury...
machine ssc, login ttasseff, password xxxxxxx


::::::::::::::
jobtypes.sendcray
::::::::::::::
## list of valid Cray job types for the sendcray command
## the format is:  jobtype followed by list of aliases (all on same line)
fortran FORTRAN Fortran
spice SPICE Spice


::::::::::::::
hosts.sendcray
::::::::::::::
## list of valid AFIT hosts for the sendcray command; the format is:
##   primary hostname followed by list of aliases (all on same line)
zssc ssc SSC
zelxb bsd afitbsd
dsun2-1 dsun2 apollo
vsun2-1 vsun2 mercury
psun2-1 psun2 venus
isun2-1 isun2 zeus
```

EXAMPLES OF SYSTEM SUPPORT FILES (cont.)

```
::::::::::::::
example of users.sendcray file
::::::::::::::
## this is the sendcray valid user list; the format is  username  followed by
##   cyberuser cyberbatchpasswd crayacct crayacctpasswd crayuser
 crayuserpasswd
##   (all on same line)
ttasseff T888888 XXXXXX T999999 ZZZZZZZ T999999 ZZZZZZZ


::::::::::::::
crayjobtop
::::::::::::::
CRAYJOB,P1,STCSB.
USER,CYBERUSER,CYBERPASS.
CHARGE,*.
COPYEI,,JOBFILEXXX.
REWIND,JOBFILEXXX.
CSUB,JOBFILEXXX,MF=CRX,US=CYBERUSER,PW=CYBERPASS.
SET,R1=0.
SET,R2=0.
SET,R3=JOBTIMEXXX.
WHILE,R1=0.AND.R2.LT.R3,LOOP.
ATTACH,JOBFILEXXX=JOBFILEXXX/NA.
IF,FILE(JOBFILEXXX,AS),CHECK.
REWIND,JOBFILEXXX.
COPYEI,JOBFILEXXX.
PURGE,JOBFILEXXX.
SET,R1=1.
ELSE,CHECK.
ROLLOUT,60.
SET,R2=R2+1.
ENDIF,CHECK.
ENDW,LOOP.
/*EOR
JOB,JN=JOBFILEXXX,US=CRAYACCT.
ACCOUNT,AC=CRAYACCT,APW=CRAYACPASS,US=CRAYUSER,UPW=CRAYUSPASS.
JOBTYPEXXX.
LDR.
DISPOSE,DN=$OUT,SDN=JOBFILEXXX,MF=CB,DC=ST,DF=CB,DEFER,^
TEXT='DEFINE,JOBFILEXXX.CTASK,ALL.'.
/EOF
```

EXAMPLES OF SYSTEM SUPPORT FILES (cont.)

```
::::::::::::::
crayjobbot
::::::::::::::
/EOF
/*EOI


::::::::::::::
example of crontab entry
::::::::::::::
0,15,30,45 8-23 * * 1-6 /en0/gcs86d/ttasseff/cray/send_cyber_cray
0,15,30,45 8-23 * * 1-6 /en0/gcs86d/ttasseff/cray/send_destination
```

VITA

Todd William Tasseff was born on November 4, 1959 in Orrville, Ohio. He graduated from Fairless High School in 1977, and attended the United States Air Force Academy where he received the Degree of Bachelor of Science, majoring in Computer Science, in June 1981. Upon graduation, he received a regular commission as a second lieutenant in the USAF. He was then assigned to the Air Force Institute of Technology at Wright-Patterson AFB and served as the data communications manager for the AFIT Information Systems Directorate from July 1981 to May 1985. In June 1985, he entered AFIT as a Master of Science Degree student, studying Computer Systems.

Permanent Address: 123 West Fourth Street

Navarre, Ohio 44662

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/86D-10	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Todd W. Tasseff, B.S., Captain, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____ 14. DATE OF REPORT (Year, Month, Day) 1986 December 15. PAGE COUNT 117	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Computer Communications, Computer Programs, Executive Routines, Computer Applications

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

Title: VIRTUAL COMMUNICATIONS TO ASD'S CRAY COMPUTER VIA AFIT DATA COMMUNICATIONS RESOURCES

Thesis Chairman: Harold W. Carter, Lt Col, USAF

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER 7 April 87
Dean for Research and Professional Development
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Harold W. Carter, Lt Col, USAF	22b. TELEPHONE (Include Area Code) 513-255-6913 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19.

The AFIT-ASD Cray System, a virtual communications capability from the AFIT School of Engineering to ASD's Cray computer, was studied, created, and evaluated. The SADT design, detailed design, and System program code are included in the study. A user enters the System's user command sendcray which initiates a Cray job from a remote or central AFIT host computer, and transfers the user's input file to the central AFIT host computer. The input file is then sent first to the Cyber, which is a front-end to the Cray, and then to the Cray itself where the input is processed. Output from the Cray is sent to the Cyber and then back to the AFIT central computer. Finally, the output, with an accompanying job log file, is transferred to the user at the originating AFIT host computer.

The System uses a combination of 4.2 BSD UNIX, Cyber NOS, and Cray operating system commands, and makes use of an AFIT Ethernet network and a VAX-UNIX/Cyber HASP/modem link. The System operates over a series of UNIX C-shell programs, most of which are executed automatically. The System provides the user with simple, error-free, and automatic access to the Cray, with the potential of improved turnaround time for compute-intensive jobs at AFIT.

END

5-87

DTIC