ADA025956

BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

(14) BBN Report No. 3315

(11) June 1976

(12) 18 p.

(6) DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES.

(9) Quarterly Progress Report No. 6

1 February 1976 - 30 April 1976

(10) R. J. Schantz
R. E. Thomas

DDC
RECEIVED
JUN 24 1976
D

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the United States Government.

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2901, Contract No. N00014-75-C-0773 (15)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

060 100

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 3315	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES		5. TYPE OF REPORT & PERIOD COVERED Quarterly Progress 2/1/76 - 4/30/76 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. Schantz, R. Thomas		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0773
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Massachusetts 02138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE June 1976 13. NUMBER OF PAGES 15
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2935.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

distributed computation
National Software Works
distributed operating system
TENEX operating system

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

This report describes BBN efforts in the design of the National Software Works system and BBN efforts to integrate TENEX into the National Software Works system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

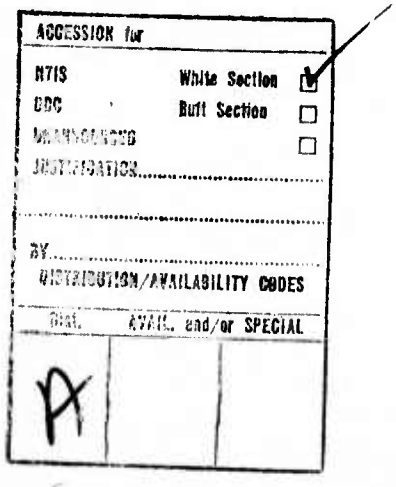
Unclassified

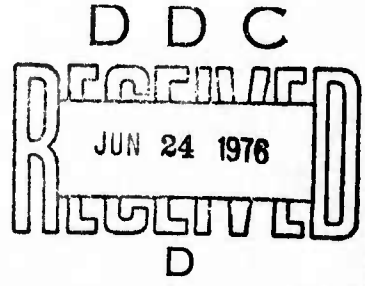
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

Preface

Previous reports for this contract have reported on research and development activities in message technology as well as distributed computation. As of January 1, 1976 the message technology work is being performed under a separate contract (MDA-903-76-C0212) with a different reporting cycle. This report describes our work in the distributed computation area whose current focus is the National Software Works Project.





I. Overview

This quarter, as in previous reporting periods, our participation in the National Software Works project (NSW) was divided into three major categories. These are system design, system implementation, and contractor/sponsor meetings to coordinate design and implementation goals and strategies. This quarter our implementation efforts focussed on a TENEX implementation of the NSW interprocess communication facility which we had designed in the previous quarter (BBN Report No. 3237). Our design efforts focussed on the NSW component known as the Foreman, and culminated in a formal Foreman design specification document (BBN Report No. 3266) which has been distributed to all participants in the NSW project.

There were a series of meetings between BBN staff members and NSW participants from Massachusetts Computer Associates to discuss Foreman/Works Manager issues, and to develop a coherent plan for establishing the NSW tool environment. These meetings provided a much needed consensus on many design issues.

In addition to these design meetings, we attended a project review meeting held at the ARPA offices in Washington, March 4th and 5th. At this meeting, the contractors clarified for attending Air Force and Navy representatives the current state and short-term future of the evolving NSW implementation. We also met with tool builders from the Information Sciences Institute (ISI) to discuss bringing the PRIM microprogramming facility into the NSW domain as a

tool. In conjunction with that meeting, we met with the System Design Laboratory (SDL) design team at the Naval Electronics Laboratory Center. This group will be one of the first user groups in the NSW, and the meeting attempted to clarify the role of the NSW in their SDL effort. A follow-up meeting with NELC personnel was held at our Cambridge office, to help guide the Navy in its use of the NSW.

The following sections provide more detail on our MSG implementation and Foreman design work.

II. NSW Interprocess Communication Facility (MSG)

During the last quarter we completed the initial TENEX implementation of MSG, the interprocess communication facility for the NSW. The MSG facility is described in overview in our last progress report, (BBN Report #3256), and a more complete description can be found in BBN Report #3237, "MSG: The Interprocess Communication Facility for the National Software Works."

The initial MSG implementation supports message exchange between processes residing on the same host or on different hosts. Both MSG modes of process addressing (specific addressing and generic addressing) are supported. In addition, the MSG alarm (interprocess interrupt) mechanism is included in the initial implementation. The implementation is currently being used to check out new versions of the NSW Front End, Works Manager, and Foreman modules.

Facilities not support in the initial TENEX implementation of MSG but which will be supported in later implementation include:

* Direct Connections.

  Direct connections provide an efficient means for a pair of processes to engage in a long-term conversation. Communication between users (at Front Ends) and interactive tools will be supported by TELNET direct connections. In addition, inter-host file transfers will be accomplished using direct connections between File Package processes. Implementation of the direct connection facility is underway and expected to be complete by the end of the current quarter.

* Message Sequencing.

  MSG does not normally guarantee that messages sent from one process to another will be delivered in the same order in which they were sent. However, MSG supports a sequencing option which a process may invoke when sending a message. This option guarantees that the message in question will be sequenced with respect to all other "sequenced" messages transmitted from the sending to the receiving process.

* Debugging Aids.

  The initial MSG implementation supports several debuggers (DDT, IDDT, BDDT) which process (i.e., WM, FE, Foreman, etc.) implementers can use to debug their processes. However, it does not at present provide a convenient way for implementers debugging interactions between NSW processes (e.g., FE and WM) to determine the status of those processes with respect to MSG itself. Implementation of facilities which will enable a process implementer to determine the status of his process with respect to MSG (e.g., status of sends and/or receives still outstanding, status of pending connections, etc.) is underway and expected to be completed during the current quarter.

* Process Introduction.

  MSG will be used to support communication between tool processes as well as between NSW system modules. In order to do this, a facility must be added to MSG which will enable a tool Foreman to make the tool known to MSG as a legitimate MSG process with access to the MSG communication primitives. The functional characteristics of process introduction are described in the Foreman specification document (BBN Report #3266).

In the remainder of this section we give a brief overview of the TENEX MSG implementation. The TENEX MSG implementation is entirely "user" code (as opposed to "system" code which would represent additions to the TENEX Operating System).

MSG on a TENEX is supported by multiple jobs which share a common data base. There are two kinds of MSG jobs:

* Central MSG job.

  There is a single central MSG job. It is responsible for initializing the MSG system and interacting with MSGs on other ARPANET hosts on behalf of local processes.

* Process managing MSG job.

  There may be one or more process managing jobs. A process managing job directly controls MSG processes (e.g., WM, FE, or Foreman processes). It responds to primitive calls (e.g., SendSpecificMessage, OpenConnection, etc.) executed by processes it controls and interacts with the central MSG job and other process managing MSG jobs as necessary to implement the primitive calls.

To explain the operation of the TENEX implementation, we describe below the interactions that occur when a WM process on TENEX Host A sends a (specifically addressed) message to a Foreman process on TENEX Host B.

Process managing MSG jobs use the JSYS trap mechanism to control processes. When the WM process executes the JSYS reserved for MSG primitives its (process managing) MSG job gains control and makes an entry which represents the message transaction in an MSG data base shared by all the (local) MSG jobs. It also observes that the destination process is not on the local host and uses the new TENEX SIGNAL facility (see BBN Report #3089) to notify the central MSG job that the message must be sent to host B. The central MSG job responds to the inter-job signal by interacting with the central MSG job at host B. If ARPANET (host/host) protocol connections already exist between the MSG jobs on hosts A and B, they are used for this interaction; otherwise, the central MSG job at host A must establish connections with the job at host B via the standard

ARPANET initial connection protocol (ICP) using an ICP contact socket reserved for MSG use. After the MSG-to-MSG connections are established the central MSG jobs interact via the MSG-to-MSG protocol (specified in BBN Report #3237) to move the message from host A to host B. As a result of this interaction, the central MSG job at host B makes an entry which represents the message transaction in its (local) MSG data base, and the central MSG job at host A uses the TENEX SIGNAL facility to notify the process managing MSG job for the WM process that the message was accepted at host B. The process managing MSG job then signals the completion of the message transmission primitive to the WM process. When a receive primitive is executed by the Foreman process at host B, the process managing MSG job which controls the Foreman process gains control, finds the message to be delivered in the local MSG data base and delivers it to the Foreman process, thereby completing the message transmission.

III. The Foreman Component of the NSW

This quarter we completed the document detailing the function and initial specification of the NSW component known as the Foreman (see "The FOREMAN: Providing the Program Execution Environment for the National Software Works," BBN Report No. 3266). This report was distributed to other members of the NSW community so that implementations could begin for the initial set of Tool Bearing Hosts (TBHs). The Foreman implementation for the TENEX family of computer systems is currently underway in this phase of the NSW development plan.

The Foreman is one component of the host software required of each TBH. It is mainly concerned with integrating tools into the NSW framework, and assuring the controlled execution of a tool in the NSW context. Every tool instance runs under the control of a Foreman. Components which do not run under a Foreman are considered part of the underlying NSW system structure itself. Below, we describe some of the salient features of the NSW Foreman; a more detailed description is to be found in the design document.

There are five aspects of the Foreman component which are of concern. They are:

* providing for tool startup/control/termination
* providing the NSW runtime environment for tools written to function in the NSW
* (optionally) providing for encapsulation of tools not written to function in the NSW by defining mappings from existing local operating system functions to NSW system functions

* providing for batch type tools
* providing mechanisms for debugging tools and recovering from errors and malfunctions

Additionally, each tool must be prevented from interfering with other tools and other (non-NSW) processes running on the host operating system. Toward this end, the Foreman implements a protection domain surrounding the tool, and a temporary workspace for file manipulation during a tool session. The Foreman processes must assure tool separation and maintain boundaries between workspaces.

Prior to initiating the tool, the Foreman selects a workspace in which to run the tool. It then initializes this workspace. This usually involves clearing the workspace directory of any remaining files. The set of Foremen processes on a TBH are responsible for managing the set of workspaces the TBH has for NSW tool support. The organization and utilization of the workspaces are left completely to the Foreman. However, the Works Manager (WM) must be informed as to which workspace a tool has been assigned, so that it can initiate proper file movement into and out of the tool workspace. The WM maintains lists of the TBH workspaces which are running tools, and these play an important role in helping a Foreman recover from system crashes without losing user files left in the workspace.

In a complete Foreman implementation, a stopped tool can be started in a number of ways. We introduce the notion of entry

vector to unify the "start tool" concepts. The WM recognizes several standard entry points for its tools. These are an initial entry point (cold start), a standard re-entry point (warm start), a termination entry point, and a continue from where stopped entry point. (Other system wide entry points will be defined as needed. It may also be possible for tools to implement private entry points invocable via the WM command language.) The meaning of all of the standard entry points is obvious, except perhaps for termination. The intent of the termination entry point is to allow a user to force control to be passed to a final tool cleanup routine, which may also involve saving some of the work of the session. Tools will normally implement commands to do this without WM intervention. However, circumstances may exist (e.g. runaway tool, depletion of resources) where it is useful to force an orderly termination while circumventing the normal tool function dispatching code.

The Foreman provides tools with a primitive operation for indicating that tool execution is complete. The HALTME primitive is the means by which a tool voluntarily relinquishes control for the final time. The Foreman may yet have to save files for the tool before actually removing the tool instance from the NSW domain. The tool can indicate the type of Foreman file processing it expects. After all file processing operations by the Foreman are complete, the Foreman notifies the WM of the tool completion and includes an accounting data list describing the cost of running the tool for the session and other resource utilization measures maintained by the host operating system.

Each Foreman must implement an externally invocable function (e.g. requested by a WM process) for probing the status of the tool currently being executed. We have taken the approach of allowing many types of probes. In response to a probe, a Foreman is expected to gather the requisite values and send them to the invoking process. Two probe types are initially identified here, with others to be added as their need arises. One initial probe queries the current state of the tool as a program in execution. The second initially defined probe queries the current state of the tool resource utilization for the session.

The NSW tool environment differs in a few key areas from the environment provided by the host operating system. The NSW has its own means for inter-component communication and for dynamically creating NSW entities, and maintains its own file system. These facilities are in addition to any similar facilities which the host operating system may already provide, and may be used simultaneously if there is no conflict with providing NSW services. The Foreman and other NSW components provide access to the NSW facilities through enhancements to the set of primitive operations available to tools. There are primitive operations for dealing with each of the areas mentioned:

* NSW file system
* NSW process communication
* NSW process creation

A tool running under the NSW system is provided with primitive operations to independently manipulate items in two distinct file spaces. One file space is the sharable NSW global file space managed by the Works Manager and maintained independently of any tools that manipulate the files. The other space is the non-sharable, temporary workspace (local file space) for the copies of the files in use by a tool during the current tool session. A file entered in the central NSW catalog must have a unique global name, and hence is able to be referenced (though perhaps not accessed) by any tool. A file which exists in the workspace for a tool can be referenced only by the tool operating in that workspace, and the mere existence of such a file may be unknown to other tools and even to the WM. The Foreman is responsible for maintaining the local workspace part of the NSW file system, and also provides the tool interface to the global NSW file space.

The NSW user accesses the NSW system through a Front End process. For those tools that require direct user involvement, the Foreman and the Front End must cooperate to provide channels for the communication. The NSW will provide tools with the ability to utilize MSG for both message type communication and direct connections with the FE. The FE could interpret and package user input and transport the pertinent data to the tool in a network MSG message. The tool to FE communication could be handled in an analogous fashion. Another approach to tool/FE communication is through the use of direct network connections. This would typically take the form of an ARPANET TELNET connection pair from the FE

directly to the tool. The decision as to which type of communication facility a tool uses is left entirely to the tool builder. The extent and type of user interaction which the tool supports, as well as the possibility of additional burden on the FE system must be weighed in selecting a mode for tool communication. A tool will be able to selectively use messages, or sets of connections, or both, depending upon the tool circumstances. The Foreman will be responsible for regulating the use of message communication to the FE and in addition, it will implement a CONNECT-TO-FE primitive operation for its tool.

As in the file system operations, the WM and the Foreman in combination will be responsible for the dynamic creation and communication aspects of the tool abstract machine, with a large assist from MSG. A tool will be provided with a primitive operation which can be invoked via the Foreman for creating a new instance of another tool. The tool will also be provided with primitives for locating service facilities which are implemented as tools, but which do not dynamically become part of the initiating tool's job, as is the case with tool-to-tool creation. With proper verification, the tool can then engage in message and/or connection oriented exchanges with its inferior tools and with the service tools. At this time, tool creation primitives are only in the discussion stages. These ideas must be examined in further detail before they can be realized in any NSW implementation.

Encapsulation: An Option in Building a Foreman

A Foreman implementation may optionally provide procedures which can be invoked automatically to allow existing local host programs to be executed as NSW tools. The operation of these procedures is known as tool encapsulation. In general terms, NSW encapsulation implies the automatic trapping and translation of local host operating system calls into calls meaningful in the NSW system. Any trapping and translation is done within the Foreman process. Using an encapsulation technique, programs which are written exclusively for the local host operating system execution environment can be made to execute as NSW tools with little or no modification. This is possible only because of the similarity, in many aspects, of the NSW system to a conventional single host operating system. Using both local host facilities and facilities supported by other NSW components (e.g. WM), the Foreman "implements" the local host primitives in a new context.

The initial TENEX approach to integrating tools into the NSW was through an encapsulation technique. This approach has proven very successful, and we, therefore, feel that each Foreman implementor should consider a similar facility. However, we must emphasize that encapsulation has limitations. There will always be local host programs which cannot be NSW encapsulated. This is because the NSW system IS different from the local host system, and substituted components can be made to appear similar only to a certain degree. Tools which utilize obscure features, or features

peculiar to a particular operating system are sure to be difficult or impossible to encapsulate correctly. Very often this will mean that certain features of a tool are not available when the tool is run in an encapsulated environment. If this is not satisfactory, or if other problems prevent the tool from being encapsulated (e.g., the local host does not have system facilities for building an encapsulator) then the tool program must be modified to call Foreman NSW primitives directly if it is to function as an NSW tool. Let us also emphasize that for a tool to be most effective in the NSW domain it should be coded using the NSW facilities directly.

We feel that for some TBHs encapsulation can have a high payoff in establishing a large class of programs as NSW tools, and should be seriously considered. It is often undesirable to recode existing programs, and it is in this area that encapsulation has its maximum effect.