

DESIGN CONSIDERATIONS FOR A NETWORK
CONTROL LANGUAGE (NCL)

by

WAYNE BARRETT CHAPIN

B.S., Kansas State University, 1969

A MASTER'S REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1977

Approved:


V. E. Wallentine

LD
2668
R4
1977
C52
C-2
Document

100

TABLE OF CONTENTS

Section	Page
1.0 Introduction	1
1.1 Motivation	1
1.2 Overview	2
2.0 Definitions	4
3.0 Review of Literature	7
3.1 Recent Approaches	7
3.2 An All-In-One OSCL	9
3.3 Batch Considerations	10
3.4 Measures of Completeness	12
3.5 On Dependent and Independent Aspects	12
3.6 Uniform Control Language Dilemma	15
3.7 Recommended Language Attributes	15
3.8 General Implementation Language Considerations	19
3.9 Motivation for a User Profile	19
4.0 OSCL Complexity for Stand-Alone vs. Network Systems	22
4.1 Extended Machine View	22
4.2 Application Point of View	24
4.3 Independent and Dependent Systems	24
5.0 Computer Communication Network vs. Computer Network	26
5.1 Computer Communications Network	26
5.2 Computer Network	28
6.0 Model of System Evolution	31
6.1 One User: One Machine System	31
6.2 N Users: One Machine System	32

6.3	N Users: M Machine System	32
6.4	One User: M Machine System	33
7.0	Application Oriented Usability and Simplicity	34
7.1	Application Evolution and Direction	34
7.2	An Application Oriented View	35
7.3	Usability Considerations-Convenient and Pragmatic	36
7.4	Simplicity Considerations-Maximum Usability	37
7.5	The Identity Crisis	38
8.0	The ION (Identity Oriented Network) Alternative	39
9.0	Perspectives on the Set of Basic Operations	41
9.1	Module Interface Perspective	41
9.2	Dependent and Independent Interface Views	45
9.3	General Interface Requirements	45
9.4	The User Profile	46
9.5	Basic Operations	47
9.6	First Person Third Person Control	49
9.7	Network Resource Control	50
9.8	Extended Machine Interface	50
9.9	Basic Operations Selection	50
9.10	Classification of Basic Operations	51
9.11	Basic Operation Details	52
9.12	Extended Machine Feedback	57
10.0	Conclusions and Recommendations	58
10.1	Conclusions	58
10.2	Recommendations	59
11.0	Appendices	63
11.1	Sample User Mapping to Basic Operations	63

1.0 Introduction

There is an increasing desire to share capabilities by interconnecting geographically distributed hardware, software and human resources. Often, economy and efficiency dictate the use of machines with differing architectures, operating systems and job control languages. The combination of heterogeneous computers to form networks faces a networking dilemma. It is economically desirable to link computers and provide new capabilities only if usage complexity does not result. The user of such systems must become familiar with multiple control languages. The system is viewed as a network of distinct computing facilities. All user levels are affected. Operations control, system support personnel, as well as applications oriented users are affected by the fusion of new and different combinations of machines, operating systems and their changing control languages.

[ENSL.P75]

1.1 Motivation

A multiplicity of control languages especially adapted to fit various user application as well as control and system support environments is needed. User identity, in our opinion an under-utilized attribute for all current implementations, can and should be utilized for the benefit of all users. The operating system control languages (OSCL) of most systems fail to gain acceptability in netted and non-netted computer systems because they fail to provide a user tailored view of the OSCL. They fail to provide application independence from a system support view. Used properly, user identity can provide for separateness of application languages.

The importance of the application independence issue is evident in two questions posed in the summary of the Proceedings of the IFIP Working Conference on Computer Language.

1. "Who are the users that should be considered with respect to the OSCL?"
2. "What functions do these users require for their support and how might these functions be described?" [ENSL.P76]

The answer to the first question requires a user or organization oriented response. That is, those who can afford a 'unique' language and an appropriate translator for their specialized application area will be considered. Question 1 is principally an economic and political question unless a means for providing application independence is identified.

The second question is a machine-oriented question. This question is addressed in section 9.0 covering the minimum set of basic operations. It is noted that whether addressing user views from a networking standpoint or a stand-alone time sharing system view point, the control language issues are similar.

1.2 Overview

In order to determine what is desirable and necessary for development of a tailored set of languages the following analysis is provided and considerations are made. Section 2 contains definitions for terms which are frequently used in this paper. Section 3 provides a critical review of recent literature covering current and planned efforts for the development of control languages. Literature covering both stand-alone OSCL and complex network OSCL is reviewed. Section 4 develops the notion that an extended machine view of computer networks and stand-alone systems can be viewed as functionally dependent rather than as machine dependent entities. Section 5 draws a distinction between network systems to develop a notion of what is the preferred user view of networks. Section 6 provides a model with which

to view the evolution of conventional and computer networks. Section 7 develops a notion of anticipated development needs in application oriented systems. Section 8 presents an additional alternative to the RAN, VAN, MON alternatives of Kimbleton and Schneider. [KIMB.J75] The additional alternative is the identity oriented network (ION). Section 9 develops a perspective within which a set of basic operations are explicated. This set is what is minimally required for implementation of a set of user oriented control languages. Section 10 provides recommendations and conclusions.

The reader interested in skimming the contents of this paper may read sections 2, 3.1, 3.2, 8, 9 and 10.

2.0 Definitions

Some terms are frequently used in this paper and are defined as follows.

OSCL . . . Operating System Control Language

This is a very general term used to represent sets of both the network operating system and stand-alone conventional operating system supported control languages.

JCL . . . Job Control Language

This term refers to the batch oriented set of OSCL commands.

RTE . . . Run Time Environment

This term refers to a set of control languages capability service utilities. These are usually loaded with the object modules of the user program to provide an interface to a set of operating system supported capabilities.

DBMS . . . Data Base Management System

The DBMS is often implemented in an RTE which interfaces to a set of operating system file and record management supported capabilities.

C/R . . . Command/Response

This term is introduced and explicated by White in [WHIT.J76]. It is an application independent protocol used to support run time environments on networks.

OS . . . Operating System

This term refers to the system code which is provided on stand-alone processors for their resource management.

NOS . . . Network Operating System

This term refers to the system code which is necessary for support of network resource management and their interface to the participating computer in a network of computer systems.

HCP . . . Host Control Process

This term refers to the process which more specifically manages the individual computer systems' participation in a computer network.

SU . . . Service Utility

This term refers to the set of software which provide services for user application of a stand-alone or netted computer facility. Such resources as Fortran computers or file maintenance packages are examples.

MS . . . Message System

This term identifies the portion of the communications facilities which most closely support communications with processes on other machines. It handles complete transactions or messages visible at the user programming level.

PLC . . . Physical Line Control

This software is responsible for communication of message packets across the machines nodes of a computer network. It is principally responsible for line controls and flow control.

NCF . . . Network Control Facility

This set of modules includes those modules which provide user control of the network resources. The set includes the extensible contractable translator, the user profiles, the network machine, the network resource controller and the extended machine interface.

NFD . . . The network file description provides global access information to a set of network files. It is accessible to the network user via a pointer in the user's profile.

NPD . . . The network program directory provides global access and parametric specifications to a set of network programs. It is accessible to the network user via a pointer in the user's profile.

3.0 Review of Literature

The literature reviewed was found in two principal research areas: stand-alone processors and networking. Conventional or stand-alone computer system's control languages have been the object of extensive recent analysis and empirical study. [UNG.C75] The considerations and views found in the literature motivate a new approach to the OSCL problem.

Two recent approaches to providing unique machine independent and application independent solutions to the OSCL problems are described in subsection 3.1. A discussion of the search for an all-in-one OSCL is provided in subsection 3.2. JCL definitions and statement of JCL purposes and requirements are presented in subsection 3.3. Three measures of revised-OSCL requirements for completeness are presented in subsection 3.4. Dependent and independent aspects of the OSCL problem are discussed in subsection 3.5. The uniform control language dilemma is discussed in subsection 3.6. The generally recommended language attributes are addressed in subsection 3.7. General implementation language considerations are presented in subsection 3.8. The motivation for a user profile is presented in subsection 3.9.

3.1 Recent Approaches

Recent efforts in the development of control languages for conventional and networking computer systems have taken either one of two approaches.

3.1.1 Universal Translator Approach

A front end "translator" is developed which is designed to provide all users with a unified, machine independent, universal control language. The following diagram depicts this approach. [DAKI.R73] . [NEWM.I73]

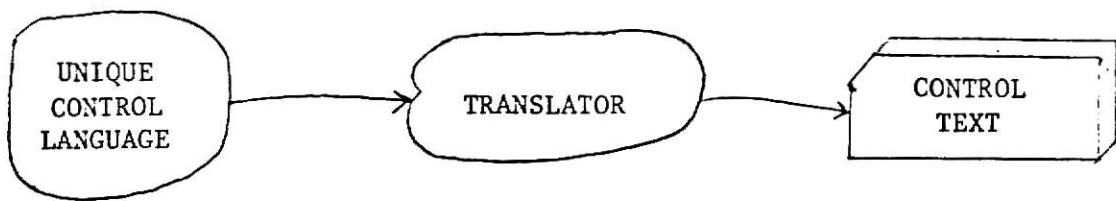


FIGURE 3.1 Translator Approach

3.1.2 Basic Operations Approach

A job control program interface or envelope, RTE, is developed which provides unique interface mapping for the use of a set of system supported "basic operations" or SVC level functions. [BARR.D74] The following diagram depicts this approach.

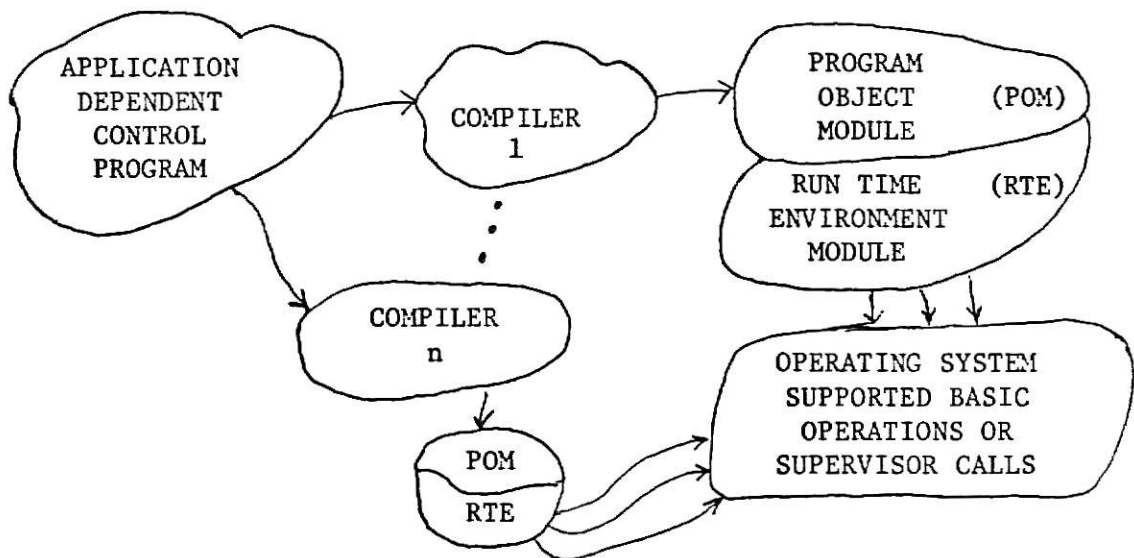


FIGURE 3.2 Basic Operations Approach

The principle difference between the two approaches depends on their suitability for resolving issues of portability, adaptability and simplicity.

The translator approach offers a relatively high degree of machine independence and portability. The basic operations approach requires mapping mechanisms for each applications language on each machine on which it is installed. This is less portable than would be one unique mapping for each

different operating system. However, if application orientation is a measure of control language simplicity the basic operations approach is found to address the simplicity issue better than the translator approach. This is so because it supports the user in his own application environment. For portability and machine independence the translator approach is recommended. For usage simplicity or application independence the basic operation approach is preferred.

How important is simplicity or the application oriented issue? Future projections indicate that by 1985 the end user of computer services will have no exposure to an academic computer science course. In general, it is predicted that the level of user and programmer sophistication will be at a very low level. [ENSL.P75]

With increasing numbers of unsophisticated computer users employing the computer in their own application areas, the need is clearly established for control languages or control mechanisms which are highly adaptive and which can be tailored to fit user application environments. A premise of this paper is that development of application oriented languages is the key to simplicity.

3.2 An All-In-One OSCL

The literature was examined for ideas about what would constitute an ideal control language. It was envisioned that a combination of recommendations existed which would lead to discovery of that all-in-one general purpose control language.

The literature is quite rich with ideas about what constitutes a good control language. It is recommended that the same rules for judgement which apply to the selection of a programming or "software development" language

be applied as criteria for evaluating an operating system control language, (OSCL). [BARR.D74] Many favor network command language standardization. [CROC.S76] The unresolved question seems to be: At what level of user sophistication?

3.3 Batch Considerations

Much of the recent effort has focused on improving the batch or job control language, (JCL), situation. We are motivated to understand definitions, purposes, requirements and implementation views of batch or job control languages.

A very general definition for a JCL is given by Brinch Hansen: "...enables users to identify themselves and describe the requirements of computational jobs: the types and amounts of resources needed, and the names of programs and data files used." [BRIN.P73]

The purposes of a JCL in conventional systems are noted by Barron.

1. to control the sequence of a number of related operations
2. to call programs (compilers, utilities) into action, providing values for various items of parametric information
3. to provide an environment for these programs by relating symbolic names to actual files or devices
4. to set up and manipulate files, directories, etc. [BARR.D74]

Kimbleton and Mondell describe four similar purposes as generic capabilities as follows.

1. identify the precedence and priority conditions required for job step execution
2. making files within a user directory known to the step which usually has its own expectations regarding the naming of the files

3. insertion of files generated by program into the appropriate directory
4. controlling the assignment of files to devices and, for a given file, controlling the layout of the file on the device.

[KIMB.S76]

The following are generally regarded as the minimally expected requirements of operating systems which support a batch subsystem environment:

1. "Job step assignment and control
2. step execution and monitoring
3. JCL generation
4. interprocess communication and support." [KIMB.S76]

Job step assignment and control is necessary if conditional execution of part of a job sequence is to be supported. Step execution and monitoring is necessary so that accounting of completion conditions can be provided to the user. JCL generation is necessary for macro expansion of job control text. Inter-process communication is necessary to allow for data dependent conditions to be reported between job steps.

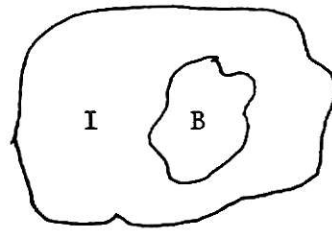
Control Language Implementation views are presented here. "JCL should be viewed as a programming language for a processor with an unusual repertoire of basic operations." [BARR.D74]

In the Brinch Hansen PASCAL reports we find the assertion that all which is necessary for a job control language is "...simply the ability to call other programs and pass parameters to them." [BRIN.P75]

The view presented in the above statements is dependent on a translator (either centralized or distributed among the called command programs) and an interpreter mechanism which minimally locates the appropriate program object code or procedure and initiates its execution. In most sophisticated systems

there is much tailoring of resource views so that symbolic manipulation of data files, for example, can be managed (created or destroyed) without detailed knowledge of unsightly physical device attributes.

It is desirable for the commands used in the batch environment to be a proper subset of the interactive command set. This minimizes the task of learning to use commands and contributes to usage simplicity.



I . . . interactive command set

B . . . batch command set

FIGURE 3.3 Batch Subset

3.4 Measures of Completeness

If an adequate means for resource control is to be identified, a measure of completeness must be established to evaluate the replacement OSCL. The following criteria have been found in the literature.

1. It must reproduce the capabilities of the discipline it replaces.
2. It must remove known deficiencies.
3. It must provide additional capabilities as required by more complex systems. [WHIT.J76]

3.5 On Dependent and Independent Aspects

Machine independent solutions appear to be the most popular ones in the literature. What system entities can be viewed as machine dependent or requiring standards?

3.5.1 Machine Independence

The users view of resources such as files should be independent of the machines on which the control language is used. [WHIT.J76] The language should be implementable or usable on small or large machines. [WALD.D72]

3.5.2 Machine Dependent

A common command/response, C/R, discipline is needed. The dialogue between server and user processes (i.e. RTE) should be standardized. [WHIT.J76] Note that this is a highly machine dependent area. Much of the literature proposes that a fixed or uniform command language be established. It is not clear that this is a machine dependent area requiring standardization. The key is separation of machine independent from machine dependent software. In reference [ENSL.P75] it is estimated that the cost of errors due to JCL or OSCL goofs amount to 1.45 billion dollars each year and therefore warrants the consideration for uniform control languages. [ENSL.P75] An often unquoted tenet of standard or uniform OSCL advocates that the user will desire to know increasingly more control language features. [DAKI.R73]. [PARS.I75] On the contrary it is highly unlikely that a user in the applications environment will need to know more than a few commands which can be prepared especially for him by a systems specialist. Generally environments such as educational institutions find it desirable to encourage the standardization of a control language so as to simplify student use of the computer. This demand for a unique control language is a parochial issue of educational institutions. A DBMS requires a different set of control than is generally required in an academic environment. The problem causing 1.45 billion dollars losses per year would more likely be resolved by the development of highly tailored applications oriented control languages.

3.5.3 Application Independence

Few articles mention the need for application independence. The common command/response discipline espoused by White is an example of an applications independent protocol. It is made so via encapsulation of the application process via a run-time-environment which intercedes for the user in mapping his operating system (OS) or network operating system (NOS) control requests.

Machine independence is not simplified via the RTE C/R approach for controlled resource sharing. This is due to the likelihood that as many machine dependent RTE programs must be developed as there are application languages on a network machine. This drawback is principally due to the lack of a 'table driven' translator capable of mapping 'n' application dependent command requests into a common unique and standardized C/R discipline for issuance in the network.

3.5.4 OSCL Translator Independence

The 'level of accuracy' required by an OSCL affects the portability or machine independence of the OSCL. As Kray et al point out:

"Commands of the existing operating systems often force the user to give more details than is necessary for the problem and the intentions of the user. By tolerating a lower level of accuracy the user can increase the number of equivalence classes and thus the probability that his JCL program can be translated into the envisaged target JCL." [KRAY.H75]

The need is to accomodate languages with excesses of required detail such as IBM's JCL. This requires a high degree of adaptability. In order to provide this, a multi-level adapting mechanism is required. By providing for user and application dependent syntax and interpreter specification, it

is expected that any level of detail can be accommodated. This has motivated a user profile whose objectives and purposes are covered in section 9.4.

3.5.5 User Dependence or Application Dependence

It is often envisioned that the correct command language or ultimate basic system functions can be resolved by study of current operating system JCL's with an eye for what is most used and what in the languages could safely be omitted. As Hertwick so aptly commented in discussions on portability of JCL programs:

"This is like looking at a FORTRAN program doing some very complicated integer arithmetic and only to find out it is doing string handling. It is easier to get insight by asking people what they are trying to do."

3.6 Uniform Control Language Dilemma

The desirable features in such a language are quite extensive. Perhaps such a high level language does not exist at the user level! That is, it would seem a contradiction for such a powerful language to support both ease of use for the simple job and yet provide the flexibility and power required for large complex jobs while also addressing efficiency considerations. As per Cheatham and Wickham:

"A system designed to cope with large complex jobs takes so much of the resources of the machine to discover that a job is small and simple that no job ever is (small and simple)."

3.7 Recommended Language Attributes

This provides motivation for a table driven translator which is adaptable to the user requirements and dependent on a unique set of profiles to direct its syntactic and semantic actions.

Newman points out that "no widely accepted 'high level' command language has yet emerged. Yet, "...in programming, a number of high level languages have emerged which meet the needs of many users and removed the obligation to learn a machine code." [NEWM.I73] So, do we possibly need a number of high level control languages? If so, what are some of the high level language attributes whose considerations is recommended?

3.7.1 Generally Recommended Language Attributes

An implicit tenet held in much of the literature is that the user should be provided with a programming language view of job control. This follows from the notion that the user can be expected to have been endowed with programming experience and a background in computing. [BARR.D74] . [PARS.I75] . [WHIT.J76] This notion is not borne out by predictions about what computer expertise future users will possess. It is also assumed that the user will need to become increasingly familiar with the control language, learn more and more of the features of the language so as to be able to extend or develop new control capabilities with the language. Further, it is recommended that the grammar should be logical, consistent or stable, and made up of as few arbitrary rules as possible. [NEWM.I73]

3.7.2 Specifically Recommended Language Attributes

More specific language attributes which are listed as highly desirable in the literature include: variable storage, inter and intra job communication, and high level conditionals such as CASE or WHENEVER operators.

3.7.2.1 Variable Store

Variable storage accessibility is desirable so that the status of various job steps or events can be saved. These variables can then be used to

represent the job state at any point in the sequence of events. This allows for decisions to be made dynamically to adjust the sequence of job steps or the job environment, (i.e. re-establish a disk file from back-up storage or alternate memory medias). Globals, typed variables and constants as well as variable scope are indicated as desirable. [BARR.D72]

3.7.2.2 Conditionals and Transfer of Control

Conditionals and transfer controls are necessary so that different job states can be distinguished and appropriate actions or operations can be involved. Some recommend the ability to have loops. [BARR.D, JACK.I72] A backward transfer is required for this capability. Brinch Hanson and others strongly recommend against the possibilities of loops in a control language, OSCL. [BRIN.P] . [DAVI.D73] If the language is considered as a developmental language then perhaps loops are advised, however, as a high level 'end-user' visible control language, not a developmental language, greater complexity is introduced in the language and thus a possibility for expanding costs due to errors in control specifications.

3.7.2.3 Communications and Synchronization

Synchronization is of concern for conventional multi-tasking systems as well as for network systems. The communications and synchronization issues are viewed here in two cases.

CASE 1 - Intra Job Communications is desirable so that changes of state can be reported between ordered or synchronized job operations, steps, phases or tasks. One method in conventional systems for providing this is via a "pseudo-register" which can be manipulated within a user program or a job control level procedure. [CDC.K73] Another method discussed in the literature calls for supporting parameters through which variable 'type' arguments

can be passed or results returned. [WHIT.J76] . [BARR.D72] Some examples found in the literature include 'the follows document' of Barron and allowances for 'multiple encoded parameters' recommended by White. The 'follows document' allows for text to directly follow the command for which it is to be received as input. The 'multiple encoded parameter' allows for parameters to be treated as variables which define a variety of encoded types. Potentially desirable types include allowance for key and file parameter types.

It may be desirable to support parallel processing within a job or interactive session. [BARR.D72] This requires synchronization via event variables and semaphores which are unique for any job or session occurrence. It is not considered likely by the author that 'end-users' need or should be aware of the details of event synchronization but that they would need to understand certain rules when invoking a set of concurrent or simultaneous events. The extent of transparency to parallel processing that can be provided to end users is an open question.

CASE 2 - Inter job communications and shared event variables and semaphores are necessary to support mutually exclusive access to shared data bases or other information susceptible to time dependent errors. Very little OSCL literature appears to address this "shared resource control" type of OSCL. Generally demands for this type of capability have been met by turn key systems in which specialized control directives are provided the end user via transaction oriented systems such as the CDC-TOOS system. In cases where the entire system or mission of the system is not wholly directed toward support of transaction oriented support then multi user job or session subsystems such as CDC TRANEX of the KRONOS system are available. In the latter type of subsystem the user submits a Cobol program to be controlled by

the subsystem executive and provides the translator interpreter mechanism to support a highly application adapted set of control directives. Unfortunately the control development language chosen, COBOL, does not check for potential problems in which mutually exclusive access is needed. Data Access synchronization is left totally as a problem for the development staff to resolve without any automation aids such as are provided by Concurrent PASCAL. [CDC.K75] . [BRIN.P75]

3.8 General Implementation Language Considerations

"Instruction sets of many contemporary computers were designed with hardware realization as the foremost constraint. Little attention has been given to the types of operations the computers will perform. Micro programming remains largely an alternative technique for manufacturer implementation of basic machine language instruction sets. [TOML.G76] The highly user oriented OSCL can be made efficient using micro code to interpret a high level implementation language such as PASCAL.

3.9 Motivation for a User Profile

The preferred view of any resource is dependent on the users application environment and level of sophistication with computers. As Barron has noted:

"The JCL statements are not clearly distinguished from the facilities they control, and this is the cause of much of the trouble." [BARR.D74]

From Barron's environmental point of view his objections are to visibility of the physical and logical level of resources. It is likely that another user level or application environment would object to the use of logical units and even files as resources.

It is desirable to consider a means for tailoring the user OSCL to provide a user preferred application environment. A user profile which contains both the syntax and specialized interpreter code could provide the desired OSCL.

3.9.1 Resource Objections

The reasons a user objects to a specified view of resources minimally include: identity or resource name convention; accessability; structure type. Objections to some arbitrary view of resources depend both on closeness or user sophistication with the real resources of a system and upon the user environment or application area. Resource view objections are described as follows.

3.9.1.1 Resource Identification

If a named resource is not part of the users psychological set and distinct from the purpose of its use, the user is likely to be confused by its name. [LEDG.H75] Most systems provide the capability to relate between logical and symbolic identifiers and some allow for dynamic pre-specification of symbolic names (i.e. use short local names rather than long permanent names). [CDC.K73]

3.9.1.2 Resource Accessability

The resource's addressability may change from time to time or it may be desirable to change a resource's origin or residence without changing the software used to access it. Most conventional systems therefore provide means for requesting a specific resource whose address space is allowed to be moved. The possibility of resource migration in computer networks presents a need for resource adaptation.

3.9.1.3 Resource Structure

If the resource imposes awkward structure to the user, it tends to complicate the use of the resource. In some applications, such as scientific use of a database, the user needs a random or keyed access view of a resource rather than a sequential resource.

3.9.1.4 Resource Type

If the resource does not conform or fit well with a user application process due to internal code differences (EBCDIC vs. ASCII), the resource will impose a conversion effort on the user and force new, probably unwanted, details about the system upon the user.

3.9.2 Profile User Adaptation

Individuals want a simple to use set of resources. For the above objections to be covered by a control language facility, there must be a user tailored point of reference. This point of reference must be such that it allows a user to identify a set of supported capabilities which are unique to the user's application environment. This suggests the need for a user profile.

4.0 OSCL Complexity for Stand-Alone vs. Network Systems

An extended machine view is described for stand-alone systems. This view is shown to cover computer networks. A conclusion is drawn that what is desirable on a stand-alone system is also desired for network systems. Thus most literature which deals with the OSCL requirements can and should be applied to understand what is desirable for networks.

4.1 Extended Machine View

The degree of usage complexity effecting a user tends to diminish as the level of abstraction moves away from the physical resource level. We often see the resources of a general purpose computer system modeled in the literature via a series of outward extending concentric circles as depicted below. Each succeeding level or outer circle represents system resources which are one level farther extended or removed from the real or physical view of resources. It follows that this should be called an "Extended Machine." We note that it is a commonly held opinion in the OSCL literature that a set of capabilities exist which would allow the computer machine to participate in a machine independent manner. [BARR.D74] . [WHIT.J] . [PARS.I] . [DAKI.R73] . [NEWM.I73] . [MADN.S74]

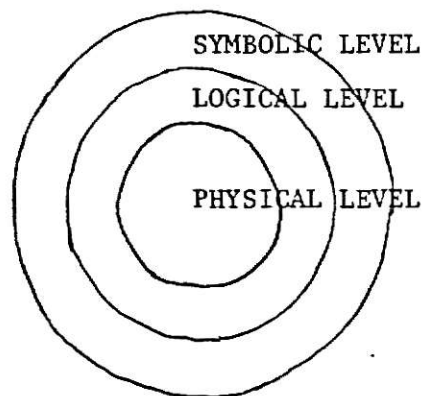


FIGURE 4.1a Extended Machine

A common view of distributed networks portrays levels of software processes with respect to communication protocol between machines. The individual members of a computer network are of two types. These are differentiated by their dependence or independence with respect to other member machines. The following diagrams place distributed networks in perspective with emphasis on an application oriented view.

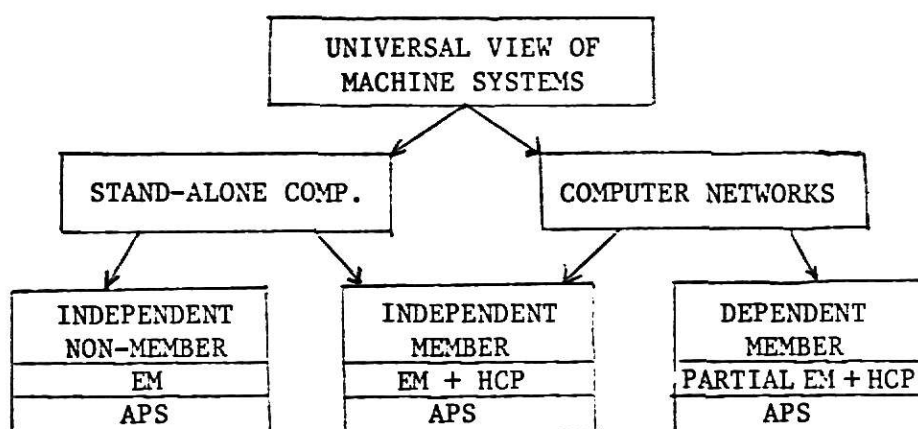


FIGURE 4.1b Universal View

The universal view of machine systems includes both stand-alone computers and computer networks. Stand-alone computer systems can participate in a computer network if they have an adequate EM (Extended Machine) plus the HCP (Host Control Program). Dependent members participation in a computer network can take place when the set of basic operations is shared between a collection of dependent or independent computer systems. The APS (Application Software) of dependent or independent computers can be shared when the basic operations of the extended machine are made available to participating network members. The HCP makes this set or partial set of basic operations available to the network.

4.2 Application Point of View

An application oriented point of view is presented here. Design decisions for both dependent and independent members of a network system are based on a set of assumptions which directly relate to the end-user's utilization of the network. These assumptions follow.

4.2.1 User Specialization

Since users are becoming more specialized, their usage can be classified according to application areas for determining: the appropriate system interface or command set; the extent and sophistication level of appropriate resource views; the level of necessary protection; the level of user closeness to the physical hardware.

4.2.2 System Specialization

Systems are specialized and can be classified according to area of specialization: scientific orientation; business orientation; industry orientation (i.e. process control, real time).

4.3 Independent and Dependent Systems

We can represent dependent and independent systems as part of a layered and hierarchical composite system (see figure 4.3). As a special case, suppose that a network system needs to look like a collection of separate computing systems. Such a system can be viewed as a set or collection of functions or computing capabilities rather than a machine dependent system. This concept of virtual machines has, of course, been demonstrated on existing conventional systems. Thus, an additional level of abstraction can be added to the network machine and this exceptional case is covered. We therefore conclude that whatever may be considered optional or best for a stand-alone system in terms of high-level usability must also be optimal for network machines.

The combined mini computer systems provide $A_1 \dots A_n$ application coverage for dependent systems.

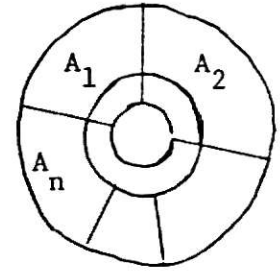


FIGURE 4.3a

The application complete computer also provides $A_1 \dots A_n$ application coverage for independent systems.

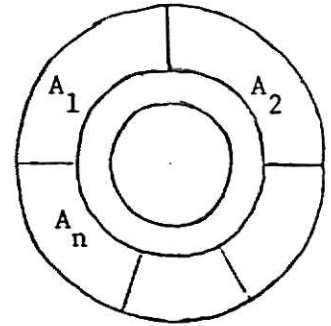
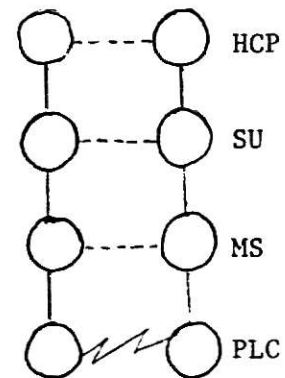


FIGURE 4.3b

With dependent network components, the communication interface is often presented with respect to logical component communications levels of:

- HCP . . . host control process
- SU . . . service utilities
- MS . . . message system
- PLC . . . physical line control



as per the diagram to the right.

FIGURE 4.3c

FIGURE 4.3 Independent and Dependent Systems

5.0 Computer Communication Network vs. Computer Network

In order to understand the complexity of using distributed network resources a dichotomy is examined. The dichotomy divides all networks into two types which are "computer communication networks" and "computer networks." [ELOV.H74] A set of factors, which is used to contrast and compare the two possible networks, is examined. The factors used to compare the two net-working types are:

- a. resource access automation
level and extent of machine extension
- b. learning requirement
extent and complexity of learning to use, ease of use
- c. level of integration and extent of individual machine visability
- d. closeness of default fit with the user application
- e. complexity of interface with basic operations of the netted
extended machines

The principle intent here is to distinguish between the desirable and the undesirable class of networks from a users point of view. However, note that there is an implied assumption that a standard and uniform view of control language is desirable. No distinction is made between user levels, unique user needs and application dependence. The importance of minimal machine dependence is an obvious need.

5.1 Computer Communications Network

With this type of network the user must be familiar with each OSCL of the serving host. Usability is complicated by this and other factors. OSCL consistency is not accomodated because individuals or groups of users are not easily identified with a set of capabilities.

The importance of this lack of association of users with a set of capabilities is exemplified as follows. Suppose the user prefers the features of a software utility and is highly dependent on its use. For sake of an example, further suppose that the software utility is an old program library management utility which is being replaced by a new version. All the users' software text is in a compacted form which is structured by the old program library maintenance package. All text files are accessible only via the old update package; the new package does not support the old package libraries. There should be a way to detect all users who are using the old utility so decisions can be made and either assist the use in converting to the new utility or continue support and offering the old utility. One might argue that the new utility should be upwards compatible with the old utility. However, changes may have been made to the new utility which conflict with this possibility. The new utility may provide features which affect the organization and output of the program library.

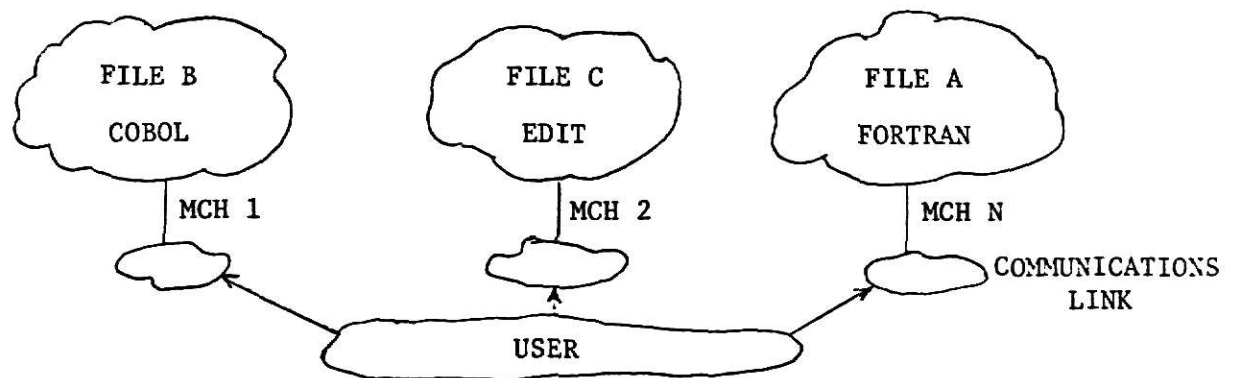


FIGURE 5.1 Computer Communications Network

a. Resource Access Automation

The level of machine extensions is likely to vary from machine system to machine system. The user must be extremely familiar with each machine system that is to be used.

b. Learning Requirement

The learning requirement is likely to be high for heterogeneous computer communication networks. The user will tend not to use available resources if the user's capabilities are machine dependent across the network.

c. Integration Level

Without machine integration the user must view the network as a collection of distinct computing systems.

d. Default Fit

In an attempt to help users and simplify the use of the OSCL, a set of OSCL defaults are developed. These are generally selected and specified by system maintenance personnel attempting to fit a general case user. The problem lies in the assumption that a single generalized user type exists. Few have not rued the day that a linking default ruined the two hour job near completion. Defaults tend to reflect the creators application needs for software maintenance.

e. Interface Complexity

No standard OSCL interface mechanism is defined to cover all systems. The interface is likely to be unique for each machine. Conversion of text files is potentially required for each machine.

5.2 Computer Network

The objective of this empirical study is to develop the concept of an integrated processing facility which provides the user with a 'computer network' view rather than that of a 'computer communications network' view.

With this type of network the user needs only be familiar with one set of capabilities which are or can be made to be application dependent. Usability is simplified because the user can select the desired view of resources to fit a specific applications environment.

It is possible and desirable for a set of capabilities to be closely tied and associated with a specific user or groups of users. In the above example, the users which are dependent on a specific capability are knowable without extensive audit processing to determine level of a capabilities usage.

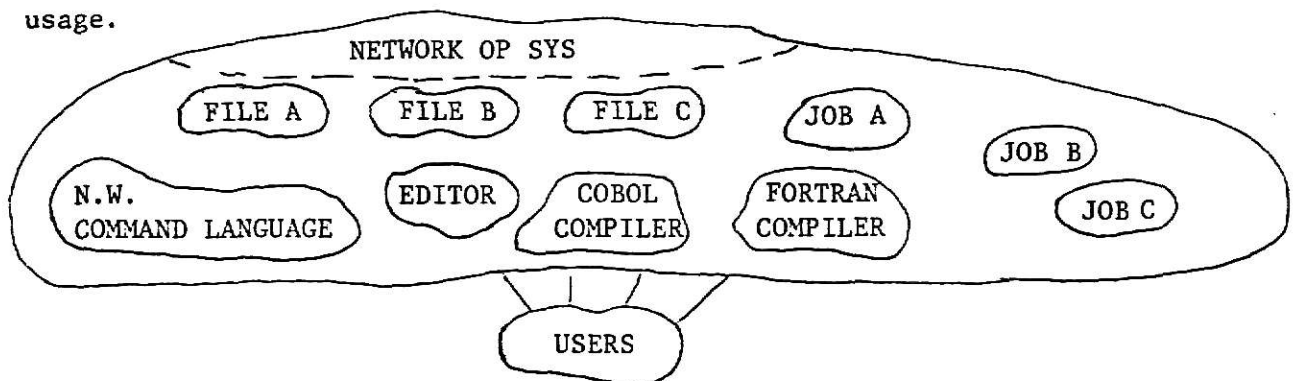


FIGURE 5.2 Computer Network

a. Resource Access Automation

Systems which are added to this network must be at or above a prescribed extended machine level to participate in the network.

b. Learning Requirement

The user needs and is restricted to a set of established capabilities (i.e. FORTRAN compiler, text editor, program library utility, etc.). The user takes part in the selection of and naming of a set of utilities. Because of the end users participation in selecting capabilities and default, there should be minimal learning time required.

c. Level of Integration

All independent systems possess a minimal set of capabilities which comprise the universal capability set. System integration is minimally at that level or above.

d. Default Fit

Because defaults are prescribed by the user there should be a minimum lack of fit throughout the network.

e. Interface Complexity

A standard for interchange of information is necessary so that all participating nodes of the network can share commands and data. An extended machine interface is required for each machine to perform any necessary mapping from network standards to EM standards.

6.0 Model of System Evolution

In order to describe the evolutionary path over which computing systems have been moving, the relationship of the user to the machine is characterized as a set of mappings of the user view onto the machine environment.

6.1 One User: One Machine System

The first systems considered are uniprocessors which operated for each user in a fairly straight forward and simple manner. The user was a sophisticated analyst or at least a well trained or experienced operator who was intimately familiar with the system and its 'quirks' and defaults. Initially, such systems were operated via switches and buttons on the machine console. Then control languages were developed to provide greater interface flexibility and simplicity via teletype or CRT consoles. [KRAY.H75] The problem oriented or applications languages were developed along a problem-oriented semantics level. The application language user pushed for their standardization. The operations and systems support user was also a form of applications oriented user, however, the scope and needs of his language were not well understood. Additionally, there was a certain amount of job security in knowing how to customize the operating system for various application usages. These systems afforded a one on one relationship between users and the operating system. When additional features were needed, they were usually developed by the user himself as a utility and added to a growing set of utilities which eventually were amalgamated and scrutinized by vendors and users to form a vendor or user standard for support.

6.2 N Users: One Machine System

These systems are comparable to RAN or time sharing systems. Multi-tasking systems were developed and provided opportunities for concurrent sharing of resources. The end user became more of a specialist. Use of the computer became ancillary although probably essential to the user's mission. The computer operator and operations staff became specialized and principally oversaw the security and direct maintenance routine of the op system. A new group - the control group has formed and administered to scheduling of the op systems usage. The sophisticated system oriented user was specialized in the support and development of new system features. Each group had need for a unique set of machine functional capabilities.

Fortunately the formal and informal interfaces are established easily for exchange of information needed to use the various system capabilities.

6.3 N Users: M Machine System Hosts

These systems are comparable to VAN systems or computer communication networks. Such systems are faced with the networking dilemma. It is increasingly difficult to share available resources as a function of heterogeneity and number of netted machines.

Computer services have been extended to include new applications areas. Computer usage has become highly specialized. Computer users background or experience with computers is minimal. Economy and efficiency considerations have produced specialized or turn-key systems. System support and resultant reliability are often at issue. Software components provided by vendors are tending toward standardization in order to address demand for reliable and supported software.

Computer support personnel are specialized. Computer operations staff, system support and control personnel are minimally involved in direct and informal communications with the end user. Application specialists provide the principle interface with the end user.

6.4 One User: M Machine Systems

This permutation represents systems such as the MON or ION variety; (see section 8: the ION section). The systems have become highly specialized to provide the end-user with a highly tailored view of the otherwise complex system. Each user may have his own private view of the computing complex. The novice can be expected to use the system for very specialized applications with very little time required for tutoring. See figure below.

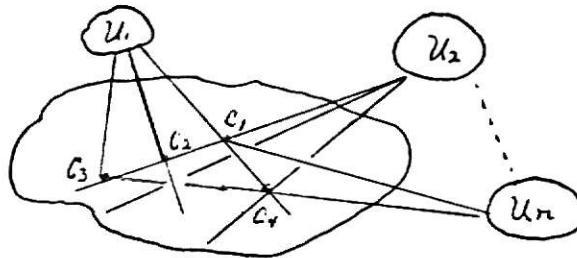


FIGURE 6.4

User U_i represents an individual user occurrence. Capability C_1 represents a set of capabilities which are available to the user U_i . Note some or all of C_1 are shared. Each user potentially has a unique view of these capability sets ($C_1 - C_m$).

7.0 Application Oriented Usability and Simplicity

The applications oriented user does not generally desire to know more than is fundamental and necessary to accomplish a set of goals. Those who deny this are to be found among those who support others use of the computer system and their applications. Those who wish to remain in a specific application area will more likely ask how and never ask 'why' with a desire to actually know.

7.1 Application Evolution and Direction

The trend in computing is moving towards increased specialization. Application oriented users are tending to be less sophisticated computer users as new computer applications are added to the total capability set.

Enslow reported the following four interesting statistical predictions for the year 1985.

- "a. Over 85% of all data processing will be done in event driven, data base oriented systems;
- b. At most 0.5% of all data processing will be done in academic and research laboratories;
- c. None of the 'end users', and no more than 25% of the applications programmers, will have been exposed to any academic computer science courses;
- d. And, at most 2% of the total programmer population will have computer science degrees." [ENSL.P75]

Jardine reports, "...the trend is to remove from the application program specific knowledge of the operating system environment under which it is to be executed." [JARD.D75]

The specification of control languages has been slow principally because very little is understood about them. It is difficult to judge whether a standard would be restrictive and tend to stifle future work in the area.

7.2 An Application Oriented View

Application orientation would allow the computer system to be controlled by the "end-user" in a more convenient manner. What is the most convenient manner for an "end-user?" There are many who take a machine oriented view of their OSCL. So, it is commonly seen as desirable to provide all users with a uniform control language. An application oriented view or a goal directed view asserts the following tenets.

- a. The "end-user" is not driven by a desire to understand computer control software any farther than is absolutely necessary. It is preferable to solicit consulting staff for those occasions when the application environment has grown computer-wise and needs more application oriented control ware.
- b. A control programming development language can be identified to support any currently visible application oriented control language.
- c. The most convenient control language is one which has close psychological fit with the user application environment.
- d. It is practicable to provide each user or user group with a control language which can be translated into a standard syntax for flow through the network to an extended machine interface.
- e. User identify is unique throughout the network.

7.3 Usability Considerations - Convient and Pragmatic

The extent or level of OSCL is determined on the basis of need and psychological closeness of the OSCL to the "end-user" applications environment. Recent efforts to simplify and adapt the OSCL to be more usable have produced defaults. However, the selection of defaults may not fit the users needs. Defaults tend to reflect the creators application needs. The extent of usability is a function of the following considerations.

- a. The language must be easy to learn. This is influenced by the closeness of the language to the "end-user" environment or psychological set.
- b. The language should be extendable.
- c. It should provide for user and system protection and security.
- d. There should be a minimum of rules.
- e. It should be consistent.
- f. It should be usable in both the batch and interactive environment.
- g. It should allow for a minimum of command data entry which is tailored to fit the users application environment.

Usability combines with simplicity when considered from the "end-user" viewpoint. The user with unique application oriented commands is joined with other users in the following description. Users and specialized user groups are represented by focal points depicted below as U_1 , U_2 , U_n . Eminating from each user and translator interface to the network are nine edges. These edges represent subsets of goal oriented capabilities. Each user has one or more goals which must be satisfied in some minimal capability set. Two capabilities are being shared by U_1 and U_2 . Three capabilities are shared between U_2 and U_n . One capability is shared between U_1 and U_n .

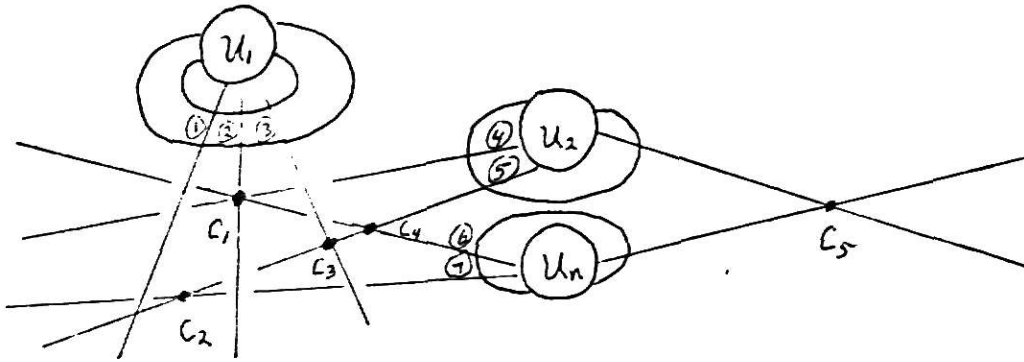


FIGURE 7.3 Capability Sharing

The sharing of system capabilities is similar to that of sharing use of a program library in stand-alone computer systems. With centralized, machine oriented capabilities the user must assure that such objects as file identification, subroutine names and any other named objects do not conflict with existing named objects.

7.4 Simplicity Consideration - Maximum Usability

In its most usable form the capability set which fits the user's environment is also in simplest form in terms of the user environment. If simplicity is also viewed in terms of a minimum capability set the following set diagram is used to evaluate the capability set for closeness of fit with minimum capabilities.

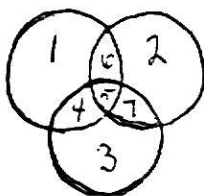


FIGURE 7.4

1. Current estimate of capability set need.
2. Future capabilities to be added to the system.
3. Unused capabilities.
4. Unused capabilities originally perceived as needed.
5. Unused future capabilities perceived mistakenly as needed.
6. Used future capabilities
7. Unused future capabilities which were perceived as needed.

The smaller that subsets 4 and 5 are, the closer the capability set is to being simple with respect to minimization of the capability sets. Minimization of the user's capability set also contributes to protection and security. This can be accomplished if the user capability set is scrutinously examined for potentially harmful user access to unneeded commands which allow the user to accidentally or intentionally misuse the capability set. The scrutiny could be automated.

7.5 The Identity Crisis

In reviewing the literature it is noted that user identity is considered only from an accounting point of view. It would seem desirable to use the user ID for keeping track of available files and unique user or application oriented control language syntax. One known exception where user identity is used for purposes other than billing or accounting is the KRONOS time sharing system. In this system the user's number serves as a hash key to locate user files. The system also uses this identification to provide for the enforcement of user restrictions. User identity could also be used to prepare the system for maximizing access to user store via moving the user stored data to a more optimal area for accessibility on secondary storage medias. This report proposes that user identity could be used to provide a more complete user directed form of tailoring for the end-user. A set of user defaults and special application oriented constructs could be made available to the end-user via more complete use of the user identity through use of a user profile.

8.0 The ION (Identity Oriented Network) Alternative

In order that this work may be placed in perspective, another model of networks is examined. A reasonably good framework for visualizing the evolution of resource sharing networks is provided by Kimbleton and Schneider. Three "basic network sharing alternatives" are identified as the RAN (Remote Access Network), the VAN (Value Added Network) and the MON (Mission Oriented Network). [KIMB.S76]

The RAN alternative is essentially available according to references in the article. This type is exemplified by time sharing systems such as Cybernet [CDC.K73] and Tymnet [BEER.M71]. The VAN alternative is described as a technological hurdle for which most or many solutions are on the horizon. The principle distinction made between the VAN and MON approaches are due to differences of degree in technological commitment needed to address what are considered basically organizational and political issues.

The principle issues addressed in the VAN approach are those of topology, protocols, user support, and network management. These issues are basic to the problems of interconnecting host machines as 'computer communications networks.' The VAN alternative is seen here as a basic and minimized solution to the network machine communications problem. It minimally addresses the network-user communications problem. The MON alternative embraces the issues of both organizational and technological concern. This alternative view represents the organization which controls the network as a "single administrative organization." Another 'alternative' view will be presented in which multiple individual organizations are in control of the network. The principle issues addressed in the MON approach are those which are generally visible at the user level. They include organizational issues such

as economy and finance, load leveling, information sharing, protection centralization, flow of funds, autonomy and technical usage control issues, such as the network operating system, control language, resource sharing, etc.

The possibility for another alternative is envisioned as one which addresses itself or adapts itself to a set or collection of organizations with the same consideration for organizational and technical concerns ascribed the MON alternative. This alternative shall be called an ION for Identity Oriented Network. In this network the emphasis is placed on providing individualized views of resources and resource capabilities based on user identity and associated attributes such as point of origin and potential immutable forms which may be made available, such as finger print, ID card, etc.

9.0 Perspectives on the Set of Basic Operations

The following subsections are included in this section. The various modules of the system are placed in perspective in subsection 9.1. The minimization of system dependent modules is discussed in subsection 9.2. General interface requirements are provided in subsection 9.3. User profile objectives and purposes are given in subsection 9.4. Basic operations considerations for portability or machine independence are given in subsection 9.5. First person third person command distinctions are discussed in subsection 9.6. The purpose of the network resource controller is discussed in subsection 9.7. The purpose of the extended machine interface is discussed in subsection 9.8. Factors considered in the selection of basic operations are discussed in subsection 9.9. Basic operations are classified in subsection 9.10. The function and syntax of the set of basic operations are presented in subsection 9.11. The feedback and reporting requirements for the set of basic operations are provided in subsection 9.12.

9.1 Module Interface Perspective

In order to place the details of this paper in perspective, the various interfaces between software modules are outlined and discussed. Specific network standard protocols are described by providing language rules for the network machine (NM) with:

1. the network resource control (NRC) component,
2. the extended machine interface (EMI) component.

The following abbreviations are used to present software modules and the protocol requirements or mapping operations.

TC ... terminal control

u ... end-user

T ... extendable contractable translator

P_s ... profile syntax rules section


NM ... network machine


P_i ... profile interpreter section

NRC ... network resource control

EMI ... extended machine interface

EM ... extended machine

 ... two way interface

 ... one way interface

9.1.1 Extended View

Two models or user views are provided to place the chosen module elements of NOS in perspective. An extended machine view depicts the hierarchical structure of the network operating system. Note that this view depicts the message system (MS) as a common communications vehicle for all interfaces between translators and network machine. The message system does not provide communications between NRC and EMI modules on different machines inside or outside of any given cluster.

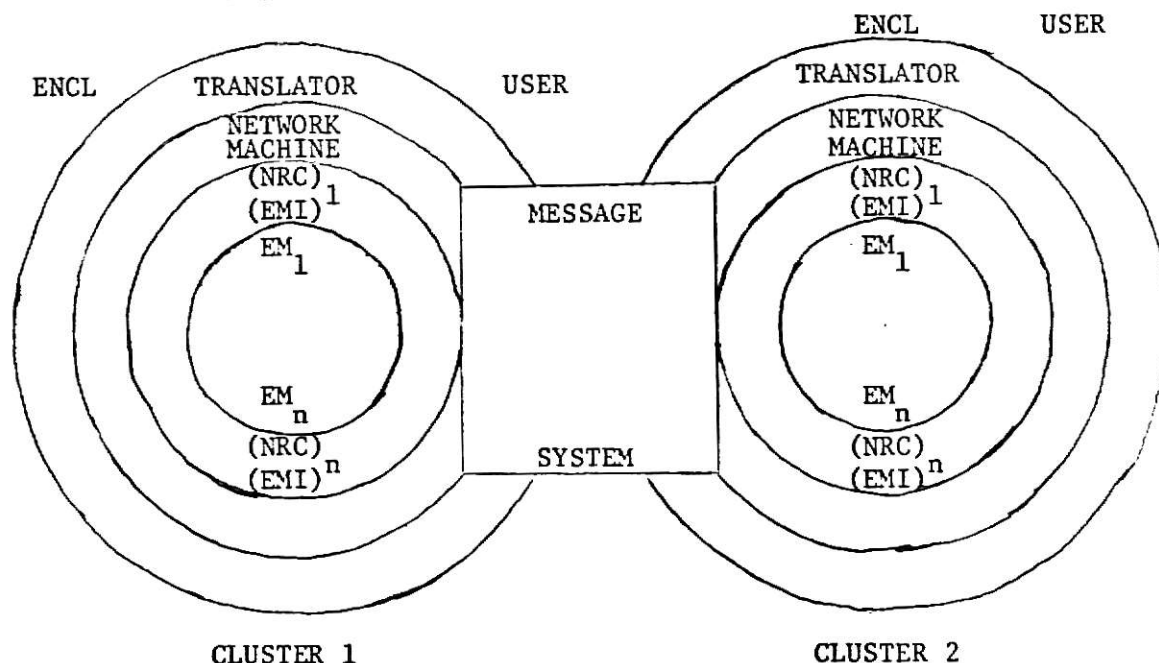


FIGURE 9.1a Extended Machine Model of NOS

9.1.2 Linear View

A linear view or model depicts flow of control and data interface between modules of the network operating systems.

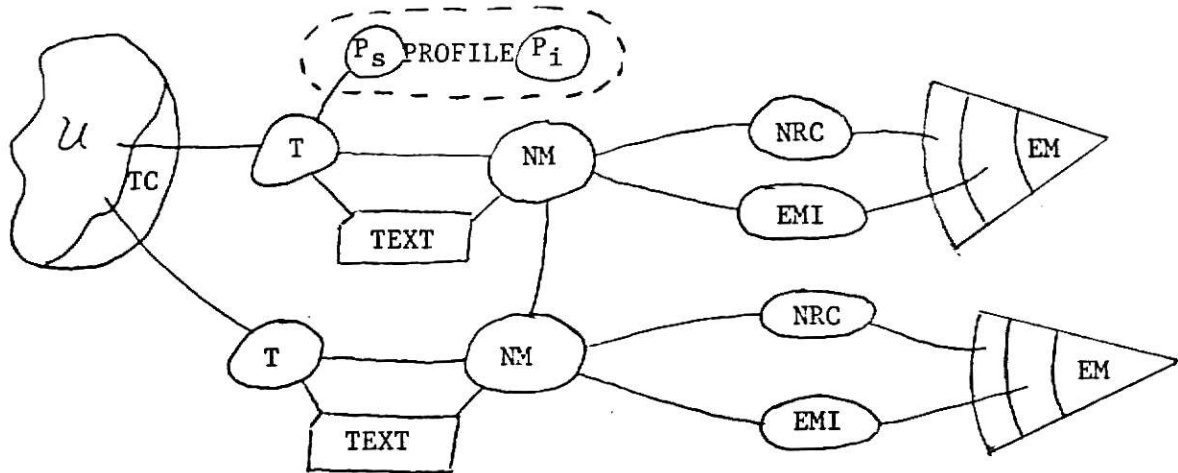


FIGURE 9.1b Linear Model

The user interfaces with the ENCL translator via some terminal controller (TC) which maps physical inputs to a network standard interface. The translator passes the user command input into an intermediate text code driven by the compile time profile section, also called the syntax portion (P_s). The network machine receives the intermediate text which is headed by a resource request block, (RRB). The RRB is sent to the NRC module. Resources are pre-allocated for the user by NRC. The NM sends RRB's to other NM's if needed to reserve resources not under its local control. The NM establishes the appropriate profile interpreter, also called run time profile section, and starts the interpretation of command text made available by the translator. When basic operation command requests are encountered while interpreting the command text, the commands are sent to the extended machine interface (EMI) for execution. If command requests are for another machine, then NM sends this request to the machines EMI via the controlling NM on the appropriate

machine. The NRC module is responsible for the reservation of resources which are owned by the set of extended machines participating in the network cluster. The EMI module is responsible for mapping network standard basic operation requests to the appropriate SVC or set of SVC's supported by the local extended machine.


9.1.3 Overview of Component Interfaces

1. $u \rightleftarrows T$... communication interface required for 'end-user' service with the extensible contractable control language translator
2. $T \leftarrow P_s$... access convention for variable specification of the control language syntax, capabilities and access rights (user variable)
3. $T \rightleftarrows NM$... network standard for communication between the translator and network machine
4. $MN \rightleftarrows P_i$... standard network protocol between the network machine and the profile command interpreter.
The sum of all profile interpreters equals the total network command capability.
- *5. $NM \rightleftarrows NRC$... standardized protocol for network devoted resource allocation and deallocation
- *6. $NM \rightleftarrows EMI$... standard network protocol for use of the extended machine
7. $EMI \rightleftarrows EM$... machine dependent protocol for use of extended-machine-supported basic operations

9.2 Dependent and Independent Interface Views

The above view of interfaces is recommended in order to minimize and segregate the machine dependent and user, application dependent parts. Interface items 1 and 2 above provide user dependent interfaces, i.e. table driven translation, for maximum user adaptability and application dependent capability views. Interface item 7 above provides highly machine dependent coverage. Interface items 3 through 6 provide for the machine independent and application independent network standards of interface.

9.3 General Interface Requirements

1. Communications between the translator and end-user must be bidirectional,  , for interactive usage and unidirectional for batch usage. It is recommended that the translator be transparent to mode of usage. This is possible if the I/O interface is distinct for the two usage modes. That is, there is a virtual terminal interface for Batch usage and a real terminal for interactive usage.

2. Interface between the Profile syntax section, P_s , is unidirectional and only provides for translator access to the user profile syntax section. So, the translator must have access capabilities to all user identified profiles and make policy decisions on behalf of the user profile access.

3. Interface between the translator and network machine is bidirectional. It must be possible for the translator to communicate command text to the network machine and also possible for the network machine to communicate feedback through to the end-user. Otherwise, there need be a mechanism for switching access to the user terminal for mutually exclusive access to the end-user's terminal or virtual terminal.

4. Bidirectional interface is required for the network machine and user profile interpreter. This is necessary to support conditional execution of the command text which is data dependent.

5. Bidirectional interface is necessary between the network machine and the network resource controls to provide feedback as to the success or failure of the machine dependent command for resource allocation.

6. Bidirectional interface is necessary between the network machine and the extended machine interface to provide feedback as to the success or failure of basic operation's execution.

7. A machine dependent form of bidirectional interface is necessary to provide feedback of success or failure of basic operations.

9.4 The User Profile

The general objectives and purposes supporting the need for consideration of a user profile are discussed here.

Profile Objectives

1. variable syntax definition: to provide user oriented specialization of the system environment which provides a framework for creation of simple to use commands which are tailored to fit the user environment
2. load leveling efficient resource allocation: to provide user related information to the system which can allow more efficient resource allocation development
3. simplify operations management: to provide system use control
4. protection and security: to provide protection and security to both the user and the system

Profile Purposes

The specific purposes addressed in the profile design are:

1. tailored language: to define a translator level which consists of a set of language syntax rules and a base level profile interpreter for determination of the semantics of the tailored control language
2. user default definition: to define implicit parameters for the semantic routines
3. define user capabilities: total directed capability requirements are explicitly defined through user profile. Both the semantic and syntax rules are part of the profile. Protection and security are improved due to the requirement for a master system profile to increase the user's capability set.
4. define resource requirements and limitations: only the operations staff (also a specialized applications oriented group) can increase the capability set

9.5 Basic Operations

This subsection places control language basic operation in perspective for portability considerations. Basic operations must be at a sufficiently high enough level to allow for them to be used in systems which otherwise are in conflict with respect to the order of lower level operations such as with file linkage, file lookup and logical unit assignment.

9.5.1 Basic Operation Considerations

The basic operations must be at a sufficiently high level to allow a set of network standard basic operations to be machine-system independent.

Batch Machine-System Independence

There are essentially two types of operating systems with respect to batch control. These two types of systems differ in the manner in which jobs are processed. The two types are referred to as task oriented and job oriented systems.

1. task oriented: These systems require that each task be initialized and set-up (i.e. loaded) before file resources are linked and established for the specific task execution. This type is typical of mini systems.
2. job oriented: This system does not require that the task be established prior to file resource linkage with the system. Files are established prior to or during task establishment by a loader process. Call by name parameterization is the normal mode for file passage in such systems.

The following are examples.

-A task oriented dialogue is as follows:

Job ident	for accounting use
Task 1, ident ₁	task space reserved and units assigned
while	
linked to task 1	
do	
specify symbolic assignment with units	obtain and assign files to task ₁
Task ₁ execution	
data ₁ items for task ₁	
.	
.	
.	
Task _n , ident _n	
Same _n assignment with another set of files	last task establishment with file-set assignment
Task _n execution	
data _n items for task _n	
End of job	

-A job oriented dialogue is as follows:

```

Job, ident
┌ specify files to
│ be obtained for job
│ use prior to task assignment
│
│ [Task1 (with files to be assigned as parameters
│   .
│   .
│   .
│   may obtain other files as necessary
│   .
│   .
│   .
│ [Taskn (with files to be assigned as parameters)
│   Data items for all tasks as separate record sets
└ End of job

```

9.6 First Person Third Person Control

In order for the network machine to control the sequence of events in a job or interactive session, a set of operations must be provided by the extended machine. These are referred to as "third person" operations. This set of commands are issued in behalf of the task process. They essentially provide start, stop, pause and continue control to a third person component, the network machine. These are necessary if the user is to be provided with execution control. Also, they provide for scheduling of events for implementing aging and priority based scheduling so that fair scheduling algorithms can be developed.

A set of commands must also exist to support "first person" control. This set of commands are issued by the task process itself in its own behalf. They provide control over files or data sets. Examples are: file creation, file destruction, file motion, file lookup.

9.7 Network Resource Control

This module provides for the preallocation and deallocation of network resources local to a given node of the network. PREALLOCATE and DEALLOCATE are exclusively addressed and provided by NRC.

9.8 Extended Machine Interface

The EMI provides an interface mechanism for mapping between the basic operations to the network machine and those operations which are generally supported by local machine operating systems.

9.8.1 Purpose of the EMI

The principle purpose of the EMI and NRC is to provide a machine independent and standard view of those basic operations which are required by the network machine. Since EMI and NRC provide the mapping mechanism for the transformation of network standard commands to extended machine commands, they are machine dependent modules.

9.9 Basic Operations Selection

The need to use existing extended machine capabilities motivates the specification of this set of basic operations.

The following factors were considered in selecting and specifying a set of network standard basic operations.

1. The minimum set must be supported by a computer system before it can fully participate in the network.
2. The operations must not impose an order of specification at the network command processor level which is not compatible with existing system software. The following examples are provided.

- With Interdata systems, file assignment follows task/program loading for each occasion of task load operation.

-With CDC-CYBER systems, file assignment preceeds program loading and is an implicit loader function once files have been assigned within a job or session.

3. Efficiency considerations dictate the inclusion of frequently used operations. It is noted that all basic operations can be accomodated via one basic operation. The RUN command could allow the invocation of all other basic operations. However, the RUN command must assume that the program module which it initiates is not resident or part of primary memory. An exception to this assumption requires RUN command processing to differentiate between resident and non-resident modules or makes the RUN command implementation dependent on an extended machine supported directory.

4. The need for more elaborate file management or record management functions is not addressed in this set of basic operations. Data base management implementations may require other than sequential files. Additional file structures such as random (word addressable or direct) access may be required. This may be accomodated by the addition of open and close operations.

5. The need for adding additional basic operations should be considered in the development of the extended machine interface.

9.10 Classification of Basic Operations

The basic operations recommended for support of the network command processor may be classified according to closeness to the primitive levels of the host operating system or "extended machine" support software. Operations listed below are explicated in the section "Basic Operations." The three classes of basic operations are as follows.

Class 1: This set of operations are those which support process management, such as start, stop, pause and resume. They are commonly available in

the standard set of supervisor level calls provided by vendor system software. These operations can be used by a supervisory process to control the execution of a subordinate process.

Class 2: This set of operations supports file management. The operations which are generally available are: (Create, Delete, Modify and Lookup). These are commonly available via supervisor components. For ease of implementation, it may be desirable to map the network command syntax into that of the host-extended-machines command processor. This would introduce one level of software above the supervisor call interface level. Also, the EXEC commands are submitted directly without mapping to the command processor level.

Class 3: This set of operations requires a composite of Class 1 and Class 2 extended machine software support. These operations are: (RUN, pre-allocate and de-allocate). The RUN command represents a composite of object loading, file assignment and execution. The pre-allocate and de-allocate operations may require interface to existing dead lock prevention software or the inclusion of special software in the NRC module to manage resources dedicated specifically for shared network usage.

9.11 Basic Operation Details

For an example of the mapping of user application commands to appropriate basic operations see Appendix A.

The following are Class 1 extended machine operations.

STOP (discontinue task process permanently)

This operation causes the task against which it is issued to be terminated.

Syntax example:

STOP (<ntid>)

The <ntid> is a network task identifier. For more detail refer to the "RUN" operation.

PAUSE (halt task execution temporarily)

This operation causes the task against which it is issued to be suspended until a RESUME operation is performed.

Syntax example:

PAUSE (<ntid>)

For a description of <ntid> refer to the "RUN" operation.

RESUME (continue task execution)

This operation causes the task against which it is issued to be removed from a suspended state and made ready for execution.

Syntax example:

RESUME (<ntid>)

For a description of <ntid> refer to the "RUN" operation.

The following are Class 2 extended machine operations.

EXEC (machine dependent operations)

This operation allows the user who has familiarity with the Host extended machine to employ that set of operations. The machine dependent code which follows this command is sent directly to the extended machine command processor.

Syntax example:

EXEC <machine.dependent.code> <eol>

The <machine.dependent.code> may be any valid command on the host machine on which this command is issued.

CREATE (define and allocate file)

The operation establishes a file control block, binds a symbolic

file identifier to the file and performs any initial operations required to define or create a permanent file storage area on secondary memory.

Syntax example:

CREATE (<f.name>, <password>)

The <f.name> parameter is a unique file name at the machine on which it resides. It is made unique by appending the user identifier to the file name. Access to the files is accomplished in the profile through use of a network file directory (NFD) pointer. The NFD provides a unique machine source address which defines the source of the named file.

DELETE (destroy file)

This operation returns all primary and secondary storage associated with the named file.

Syntax example:

DELETE (<f.name>, <password>)

For a description of <f.name> and <password> refer to the above "CREATE" operation.

MODIFY (change fdb attributes)

This operation allows file definition attributes to be changed and protection mechanism to be specified (i.e. file name, account identification, password, hash identifier).

Syntax example:

MODIFY (<f.name>, <password>, ky = <new.syms>⁺)

where: ky is some keyword such as:

NF, for new file name;

AI, for new account id;

PW, for new password;

HI, for new hash identifier.

<new.sym> is a character string

For a description of <f.name> and <password>, refer to the above "CREATE" operation.

LOOKUP (find and report fdb attributes)

This operation provides a means for file definition attributes to be examined by an authorized user.

Syntax example:

LOOKUP (<f.name>, password)

For a description of <f.name> and <password>, refer to the above "CREATE" operation.

The following are Class 3 extended machine operations.

RUN (establish, load, assign, execute)

It must be possible to direct the system to load an absolutized or runnable object module into primary memory, bind files to logical units referenced by the program module being loaded and start the program into execution.

Because systems perform a variety of functions with their load operation and may not support explicit binding of files to units with assignment operations, the high level RUN operation is used to represent this composite function.

Syntax example:

RUN (<ntid>, <net.task>)

The <ntid> parameter is the network task identifier and is unique within the network. This parameter consists of three parts (CMT); C for cluster identifier, M for machine identifier and T for task identifier.

The <net.task> parameter represents a position dependent string of files and or key words which are task dependent.

The network program directory (NPD), accessed via the user's profile, establishes the order and type of parameters required.

RESERVE (pre-allocate resources for specified job)

This operation is required to allow for deadlock prevention in the network. Implementation of this operation is dependent on the resources associated with the host extended machine. The implementation is also dependent on the sophistication of the extended machine. Some operating systems provide for resource allocation and deadlock prevention while less developed systems do not. Thus, the Network Resource Control module software will need to adapt for those systems which do not provide resource allocation commands.

The set of resources which may be pre-allocated should not be permanently fixed or restricted. However, a minimal set of resources should be accomodated. These are:

- central memory requirement (512 bytes);
- maximum # of sub jobs to run concurrently;
- file storage requirements;
- file directory logical units.

Syntax example:

RESERVE ([SJ~~mmmm~~,11]⁺, FSffff)

where: ~~mmmm~~ represents central memory requirements in
512 byte units;
11 represents logical file units needed;
ffff represents file storage requirements.

RELEASE (return resources for a specified job)

This operation is the counter part to the reserve operation. It must therefore be capable of returning all resources types which were reserved.

Syntax example:

RELEASE ([SJ<ntid>]⁺, FSffff)

where: <ntid> is described under the RUN operation;

ffff represents file storage being returned.

9.12 Extended Machine Feedback

E.M. Feedback/Reporting

RUN: After all task terminations, a return code (RC) and completion message is sent to the NCP which issued the RUN request.

EXEC: The above without a network standard return code.

STOP, PAUSE, RESUME: A coded status indicating the completion condition is returned to the NCP which issued the request. A message may or may not be present.

CREATE, DELETE, MODIFY, LOOKUP: A coded status plus a message report are always returned.

RESERVE: A reject message is sent if resources are not currently available. A network task identifier is returned if resources are available for each task requested.

RELEASE: (a) A normal completion message is issued to the network machine if all resources could be released (i.e. check made to assure limits not exceeded).

(b) An alarm message is issued if resources are in excess.

The operator is notified.

10.0 Conclusions and Recommendations

The following conclusions and recommendations are provided.

10.1 Conclusions

This paper has principally addressed the two questions cited in the introduction. The basic modules of NCP and the necessary basic operations have been identified to support a variety of user application environments. It is the author's opinion that the correct resolution of question two will make question one a more or less mute question. That is, the multiplicity of supported application environments tends to eliminate political and economic concern for which users are to be considered. They can all be considered.

The need for application and machine independent NCL views is identified. The literature review of section 3 identifies two current approaches. These approaches were found to be individually lacking due to either lack of application independence or machine independence in their approach to the OSCL problem.

Several individual factors have been discussed. The importance of a simple OSCL is identified along with what constitutes OSCL simplicity from the 'end-user' standpoint. The lack of existence of an 'all-in-one' OSCL is shown. Definitions, objectives and requirements for an OSCL from a batch use standpoint are identified. The need for an OSCL which serves both batch and interactive use is identified. Measure of solution completeness are described. Various OSCL considerations are explicated. The need for a user set of individualized capabilities motivates the need for a table-driven translator whose table is the user profile.

Literature providing stand-alone computing considerations are shown to be relevant with respect to networking systems. The idea of user and system

specialization is discussed and shown to fit the networking model of extended machines. A user preferred view of networks is discussed. A model of system evolution is presented which places networks in perspective as a many-on-many relationship of users to machines. The trend in application specialization is examined. It is shown that user language simplicity is a key issue in the development of an OSCL.

The motivation of this effort is placed in perspective with recent work which categorizes a set of network alternatives. The ION approach is presented as an additional alternative to network systems.

Finally, the modules which constitute the NCF are placed in perspective and discussed. The set of recommended basic operations are presented.

10.2 Recommendations

The set of basic operations are possibly weak in one application area. That is, the set does not provide for record management commands explicitly. This type of command can be added and invoked via the RUN command. It is an open question whether this is appropriate or not. It is therefore recommended that further study be done to determine whether or not a record management command is needed. It could be added to the CREATE command or a unique command could be provided.

In providing record management control the 'first person', third person' considerations should be made. Is it possible that the user would need the ability in a program to dynamically redefine, as 'first person', the expected record format from within the user's program? Or, is it sufficient to provide a 'third person' type of command capability via the CREATE command?

It is the author's opinion that the need for an alternative, or ION, view of networking has been established. Further work with an actual implementation is the next step toward solving the networking dilemma.

BIBLIOGRAPHY

1. BAIR.G75
Baird, George N. (U.S. Navy)
"Fredette's Operating System Interface Language", in Command Languages, edited by Unger, C., published by North-Holland Publishing Co., 1975, pp. 267-279.
2. BARR.D72
Barron, D. W. and Jackson, I. R., "The Evolution of Job Control Languages", Software Practice and Experience, Vol. 2, pp. 143-164.
3. BARR.D74
Barron, D. W., "Job Control Languages and Job Control Programs", The Computer Journal, Ag '74, pp. 282-288.
4. BEER.M71
Beere, M. and Sullivan, N., "Tymnet-A Serendipitous Evolution", Second ACM Symposium on Problems in the Optimization of Data Communications Systems-Proceedings, October 1971.
5. BRIN.P73
Brinch Hansen, Per, "An Overview of Operating Systems", Chapter 1, Operating System Principles, Printice-Hall, p. 2, 1973.
6. BRIN.P75
Brinch Hansen, Per, "The Solo Operating System-A Concurrent Pascal Program", The Solo Operating System, private report, 1975.
7. CDC.K73
"KRONOS 2.1 Reference Manual", Publication No. 60407000, Control Data Corporation, 1973.
8. CROC.S72
Crocker, S. D., et al., "Function-Oriented Protocols for the ARPA Computer Network", Spring Joint Computer Conference, 1972, pp. 271-280.
9. DAKI.R73
Dakin, R. J., "A General Control Language: Language Structure and Translation", The Computer Journal, Vol. 18, No. 4, pp. 325-332.
10. DAVI.D73
Davies, D. W. and Barber, D. L. A., Chapter 11, "Protocols, Terminals and Network Monitoring", Communication Networks for Computers, John Wiley and Sons, New York, 1973.
11. ELOV.H74
Elovitz, Honey S. and Heitmeyer, Constance L., "What is a Computer Network?", National Telecommunications Conference, 1974 Record, pp. 1007-1014.

12. ENSL.P75
 Enslow, Philip H., "Summary of the IFIP Working Conference on Operating System Command Languages", Command Languages, North-Holland Publishing Company., 1975, pp. 389-395.
13. GAGL.U75
 Gagliardi, U. O., "Trends in Computing System Architecture", Proceedings of the IEEE, Vol. 63, No. 6, June 1975.
14. JARD.D75
 Jardine, D. A., "The Structure of Operating System Control Languages", Command Languages, North-Holland Publishing Company, 1975, pp. 27-42.
15. JENS.J75
 Jensen, Jorn and Lanesen, Soren, "Programming Language Extensions Which Render Job Control Languages Superfluous", Command Languages, North-Holland Publishing Company, 1975, pp. 137-152.
16. KELL.C68
 Kelley, C. R., Manual and Automatic Control, John Wiley and Sons, Inc., New York.London.Sydney, 1968.
17. KIMB.S75
 Kimbleton, Stephen R. and Schneider, G. Michael, "Computer Communication Networks: Approaches, Objectives, and Performance Considerations", Computing Surveys, Vol. 7, No. 3, September 1975.
18. KIMB.S76
 Kimbleton, Stephen R. and Mandell, Richard L., "A Perspective on Network Operating Systems", AFIPS Conference Proceedings, Vol. 45, 1976, pp. 551-559.
19. KRAY.H75
 Krayl, H., Unger, C., Weller, T., "Portability of JCL Programs", Command Languages, North-Holland Publishing Company, 1975, pp. 293-304.
20. LEDG.H75
 Ledgard, Henry F., Programming Proverbs for FORTRAN Programmers, Hayden Book Company, Inc., 1975, 130 pages, pp. 95.
21. MADN.S74
 Madnick, Stuart E. and Donovan, John J., Operating Systems, McGraw-Hill Book Company, 1974, pp. 17-19.
22. NEED.R74
 Needham, R. M. and Wilkes, M. V., "Domains of Protection and the Management of Processes", The Computer Journal, Vol. 17, No. 2, May 1974.

23. NEWM.I73
Newman, I. A., "The Unique Command Language-Portable Job Control".
24. PARS.I75
Parsons, I. T., "A High-Level Job Control Language", Software Practice and Experience, John Wiley and Sons, Ltd., Vol. 5, 1975, pp. 69-82.
25. TOML.G76
Tomlinson, G. R. and Ashok, K. A., "Developing Application Oriented Computer Architectures on General Purpose Micro-Programmable Machines", National Computer Conference, 1976, pp. 715-722.
26. UNGE.C75
Unger, C., Command Languages, North-Holland Publishing Company, 1975.
27. WHIT.J76
White, James E., "A High-Level Framework for Network-Based Resource Sharing", National Computer Conference, 1976, pp. 561-570.
28. WULF.W74
Wulf, William A., "Issues in Higher-Level Machine-Oriented Languages", Machine Oriented Higher-Level Languages, North-Holland Publishing Company, 1974, pp. 7-12.

APPENDIX 11.1

Sample User Mapping to Basic Operations

A program text is to be compiled on Machine 1, M1, with the source text on Machine 2, M2. The resultant object module is executed on M1.

<u>User Commands</u>	<u>Basic Operations</u>
COMPILE, I = PROGA, B = LGO	01 - CREATE (LGO.uid*) 02 - CREATE (PROGA.uid) 03 - RUN (COPY, M2, M1, PROGA.uid) 04 - RUN (FTN, PROGA.uid, LGO.uid, OUTPUT.uid)
EXECUTE, LGO	RUN (LGO.uid)

- The translation identifies COMPILE as a Machine 1 operation via the profile network program directory, NPD.
- The translator identifies LGO as a temporary file because it is not in the profile network file directory NFD for this user. Thus, it produces line 01.
- The translator identifies PROGA through the NFD as resident on Machine 2, M2. It produces line 02 and 03 to make the file resident for compilation on Machine 1.
- The translator produces line 04 to cause the FTN compiler identifier from the NPD to be executed with the appropriate input and output files. Note the OUTPUT.uid file is a unique user default 'output' file for listing test output.
- The translator identifies EXECUTE as a command to run the binary object module, LGO.uid.

*General Comment: uid represents the user identity which is required to uniquely define files on a Host machine.

DESIGN CONSIDERATIONS FOR A NETWORK
CONTROL LANGUAGE (NCL)

by

WAYNE BARRETT CHAPIN

B.S., Kansas State University, 1969

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1977

ABSTRACT

This report presents considerations for the development of network control languages. The report is motivated by the need for a simplified means of resource sharing on heterogeneous computer networks, the networking dilemma. The user profile is motivated by the need for application dependent control languages provided through use of a table-driven translator. A minimal basic set of operations is motivated by the need for a machine independent implementation of control languages.

A review of the literature reveals the need for more than a unique, all-in-one, control language when usage simplicity is considered. Current approaches are found to be wanting either due to lack of machine independence or application dependence. The application of user identity is found to be limited to use in accounting for computer services. Application of user identity can and should be extended to aid in providing a set of tailored user capabilities. This extended application of user identity motivates a networking alternative called the identity oriented network, (ION).

The importance of user identity for tailoring control languages to meet individual user needs is stressed. The objectives and purposes of the user identified profile are presented. A minimal set of basic operations is prescribed. Conclusions and recommendations are provided.