Network access technology—A perspective*

by SHIRLEY WARD WATKINS and STEPHEN R. KIMBLETON
National Bureau of Standards
Washington, DC

INTRODUCTION

The utility of computer networking is widely accepted, as is the complexity of accessing network resources and services. This complexity must be reduced to support more effective user access to and utilization of these resources as discussed in References 1 through 3. Accomplishing this reduction across heterogeneous systems is a primary objective of network access support. Specific capabilities which have currently been shown to be feasible include: "macros" which permit the user to invoke via a single directive a complex of individual system and/or communications subnetwork commands which collectively accomplish a specific and common function such as "login," common command languages for file manipulation and network job execution across a heterogeneous collection of systems, "soft" user interfaces providing command completion, spelling correction, and, potentially, correction of minor syntactic errors; uniform interfaces to similar services existing on diverse systems such as bibliographic retrieval services, and the emergent area of "intelligent assistance."

Currently, there are several ongoing projects providing varying degrees and types of network access support for the interactive user—the population of interest in this paper. Appreciating the functions provided by these efforts and determining additional functions necessary to maximize access support requires establishment of a basic perspective on network access. The global objective of this paper is to provide such a perspective. This will be accomplished through: (i) identifying related research efforts; (ii) identifying problems inhibiting network access support development for the appropriate user categories; and (iii) structuring and discussing the major components of such support. One of the key components of this structure, expert assistance, shields the user from the requirement of learning yet another language—that of the access system. Expert assistance eliminates the requirement that the user become an 'expert' on the access system in order to obtain simplified access to the target system. The sixth section of this paper provides insight into how this goal may be achieved.

Rand Intelligent Terminal Agent (RITA)

The Rand Intelligent Terminal Agent (RITA) has initially been implemented on a shared minicomputer. This implementation is based on production systems (sets of condition-action rules) to encode complex sets of heuristics for handling interactions with users and with external systems. RITA performs complex time-dependent tasks over extended periods of time with minimal intermediate user input. This is accomplished through user agents (processes operating in behalf of the user). Since the productions systems can modify themselves, RITA agents can learn in the sense of dynamically modifying their behavior.

Two considerations reflected in the design of RITA are that the agents maintain knowledge bases containing heuristic assertions, data reflecting system behavior, and user preferences, and that these knowledge bases must be modifiable by the user. The adaptive behavior of RITA agents represents a sophisticated approach to the area of access assistance.

NBS Network Access Machine (NAM)

The Network Access Machine (NAM) at the National Bureau of Standards is implemented on a PDP 11/45 running the UNIX operating system and represents another shared minicomputer based approach to aiding the network user. Such support is provided through three primary mechanisms: macros which support expansion of simple user-entered commands into the command sequences executable on specific networks and host computers connected to the network; a response analyzer allowing alternative responses (typically the expected response plus error conditions which could occur) during the expansion of a macro; and control mechanisms (case statements, if-then-else, ... ). Collectively, these mechanisms constitute a command language level programming language.

The NAM design is based on the concept of presenting one uniform set of user commands which are executable

---

* This work is a contribution of the National Bureau of Standards and is not subject to copyright. Partial funding for the preparation of this paper was provided by the U.S. Air Force Rome Air Development Center (RADC) under Contract No. F 30602-77-F-0068. Certain commercial products are identified in this paper in order to adequately specify the procedures being described. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the material identified is necessarily the best available for the purpose.
across network boundaries and across heterogeneous host systems. In operation, user commands are first expanded into the command sequence appropriate to the system being accessed, responses are analyzed to determine if the interaction is proceeding as expected, anticipated errors are handled directly, and unanticipated errors are presented to the user for handling. These capabilities have proven sufficiently powerful to support implementation of a prototype common command language for file manipulation and network job execution. In addition, a uniform interface to a collection of bibliographic data retrieval systems has also been demonstrated.

Service specific access support

The use of computer networks has stimulated interest in providing a homogeneous (virtual) interface to multiple heterogeneous resources. One such effort has been pursued by Marcus at MIT in the information services community. Access to heterogeneous bibliographic retrieval systems is provided in such a way as to make them appear similar. There are three distinguishing features of this implementation: emphasizing a particular application, providing a uniform interface to heterogeneous systems such that all systems appear uniform to the user, and stressing access support to the naive user.

The initial interface, called CONIT (Connector for Networked Information Transfer), is based on the MULTICS system at MIT. Support includes a master index and thesaurus to store the vocabulary of the separate data bases along with index term interrelationships. In addition, the user is provided with a common bibliographic data structure in which data elements are structured and interrelated among different data bases. Once the desired retrieval system is selected, the user enters common commands which are then translated into the specific commands required on the selected retrieval system. The approach has been demonstrated using the ARPANET for access to the National Library of Medicine MEDLINE service and the MIT Intrex retrieval system.

PROBLEMS IN AUTOMATING SERVICE ACCESS

The objective of network usage is access to computer resources or services. Service, here, refers to programs or combinations of programs provided by a computer system: for example, assemblers, compilers, data base management, report generation, and text processing.

Since users access services, the fundamental problem of network access technology is simplification of service access. Moreover, in view of the trend toward interrelated services such as that provided by the National Software Works program production service, the problems of interfacing users to services may require consideration of both the interface to a service provided on an individual system as well as to a collection of services provided by several systems. Consequently, Common Command Languages (CCLs) across heterogeneous systems are desirable and have been shown to be feasible.

Simplification of service access requires identification of service user categories and analysis of the factors impeding access simplification.

User categories

Users of a computer system can be divided into three major categories: endusers, applications programmers who construct services for endusers, and systems programmers who provide the environment in which applications programmers function. Note that a given individual may function in all three categories; the importance of the trichotomy is its implicit definition of user objectives.

The global objective of the enduser is access to services. In particular, the nuances of the capabilities provided by a computer communication system, the general characteristics of the system, the locations actually providing the services and the availability of other services not of interest to the user are of secondary interest. The user would prefer that the computer communication system be transparent so that one sees services rather than systems and is only required to provide that information necessary to utilize the service. Further, the user should be relatively unconcerned with the precise structure of system command languages, problems caused by spelling errors and other requirements to conform to the precise syntactic requirements of the system.

Applications programmers construct services. Provided the performance impact is acceptable, high level access support of computer and communication components is very useful. Such support would reduce the need for explicit concern with the vagaries of Job Control Languages, permit common naming conventions across systems, ease access to systems, and support command language level programming.

Systems programmers will often find the software support provided applications programmers of use. However, the detailed, highly system specific knowledge required for the construction of 'systems' software often precludes adoption of a high level view. Although network access support mechanisms can be used by systems programmers, performance constraints will probably limit their applicability. This observation is consistent with the evidence provided by programming language development which demonstrates that for some functions one can only program at the assembly language level.

Looking at the objectives of each user category, it appears that endusers constitute the major user group for network access support. Applications programmers are likely to find such support to be of some help, and systems programmers seem unlikely to be directly aided in the construction or modification of 'system level' functions. Even for this restricted audience, developing the required access support proves difficult.
Access support difficulties

Access support would be comparatively simple if all users had the same problems with the same services. Unfortunately, user perceptions of the difficulty of service access are influenced by:

- user skill
- rate of change
- usage intensity
- access complexity

User skill

Developing effective access support would be difficult even if user skills were uniform across services. However, skills tend to be service dependent and, for each user, span the spectrum from “novice” to “expert.” We hypothesize that the desired access support level is inversely correlated with the user’s skill and, as a result, there is no single “best” access support methodology.

Rate of change

Rate of change measures the rapidity with which the user changes the collection of accessed services and, thereby, the rate at which the user must learn about the new (for the user) services being accessed. Although we shall not attempt to characterize the precise metric or the difference between ‘fast’ and ‘slow’ rates, it is evident that the higher the rate, the greater the learning burden of the user. Thus, we hypothesize that the utility of network access support, as perceived by the user, is positively correlated with the rate of change of service access.

Usage intensity

Usage intensity measures the frequency of use of a given service. We hypothesize that from the viewpoint of an individual user, for services of equivalent complexity (measured in terms of some unspecified metric), the utility of network access support is positively correlated with the (average) time between invocations of the service. This seems intuitively reasonable since the longer the interval between service accesses, the greater the likelihood that required items of information will be forgotten or confused.

Access complexity

Service complexity and capability are often positively correlated. We hypothesize that the desirability of network access support correlates positively with service complexity and, thereby, capability. This seems reasonable since more complex services typically require more complex interactions for their effective utilization.

NETWORK ACCESS OBJECTIVES

Identification of a “best” access support mechanism is a reasonable initial objective for network access investigations. This objective proves unrealistic due to the four factors identified above which, individually, cannot be optimized due to user differences. Heafner found the same difficulty in dealing with user differences for a subset of access support functions, namely command languages. He concluded that there is no “best” command language.

Since developing a “best” approach is infeasible, we consider an alternative—the determination of access functions which are most amenable to automation. Our alternative is based on the identification of: (i) service components appropriate for network access; (ii) support requirements appropriate to these components; and (iii) an integrated set of functions corresponding to these support requirements.

Service access taxonomy

Access to a service proceeds in four stages: acquisition, initialization, utilization and termination. Acquisition encompasses those commands required to “run” the service together with those required to gain access to the host, if required. Similarly, termination encompasses both the process of “exiting” the service and logging off the host, if appropriate. (It would be inappropriate, or at least inefficient, to log out if the next service to be accessed is also located on the same host.) Initialization covers the setting of parameters prior to actual utilization of the service to accomplish the user’s objective.

Given this taxonomy of service access, it follows that the objectives of network access support can be characterized in terms of the capabilities to be provided for each of these stages.

An assertion

We assert that service utilization is not amenable to centralized network level support. Such support can be provided for the other three stages.

Justification of this assertion is provided by both organizational and technological considerations. Organizational justification follows through consideration of the groups which could provide network access support and through analogy with the functions of a computing center counselor. Network access support of service utilization requires explicit knowledge of the service. Since one would not expect any individual or group of individuals to be knowledgeable in all network accessible services, it follows that no single group can support service utilization. Instead, such support should be provided either by user groups for the
various services, by the service developers (as was done for MYCIN, a diagnostic support service experiment for physicians,29 and is being done for the ACCAT Command and Control testbed,21,22 or through other means as is being done in the area of providing homogeneous virtual interfaces to heterogeneous bibliographic retrieval services.8,10,15 In this context, it should be noted that the RITA effort, described earlier is concerned with the development of "smart" user daemons to assist in the utilization of services. (Note that although RITA and MYCIN utilize a similar technology (production systems), RITA support is external to the service while MYCIN provides support as part of the service.)

The technological difficulty in supporting service utilization reflects its implicit requirement of knowledge of both the syntax and semantics of the service. As such, excluding trivial cases, it is effectively a specific instance of the artificial intelligence problem of knowledge representation which, in the general case, is unsolved and still a subject of study.

The following discussion establishes the feasibility of supporting service acquisition, initialization and termination at the network level. Although the difficulties of providing such support for service utilization have been articulated, some discussion of this topic is included for completeness.

Access support objectives

Network access support should simplify access to new (to the user) services and should minimize the need for entering relatively unchanged, previously supplied information when accessing a previously accessed service. We now discuss some of the major requirements implicit in supporting these two objectives.

Accessing new services

Access support of new services requires: (i) identifying the existence of the appropriate service; (ii) providing information about the service; and (iii) simplifying the problems of initial access. Within this paper we shall not consider item (i); however, some consideration has been given to this issue as is reflected in the existence of the ARPANET Resource Handbook20 and the objectives of the REX system.24

Users require two kinds of information to access a service: global descriptive information such as is commonly found in manuals, and specific problem solving oriented information to assist them in more efficient usage of the service. Although either online or hardcopy manuals are appropriate for providing global information, a dynamic mechanism providing highly specific information is required for problem solving. Such information would be based on the individual user and system. Network level support of dynamic assistance seems feasible for the acquisition, initialization and termination stages of service access. Further, it could be provided by the service constructor to support more effective service utilization.

Dynamic service access assistance requires profiles for the acquisition, initialization and termination stages of service access. Such profiles are system dependent and user dependent because access to services is dependent upon the idiosyncracies of the system providing the service and the idiosyncracies of the user accessing the service. Their implementation requires resolution of two issues: (i) determination of the information to be provided, and (ii) development of a mechanism for inserting, storing, modifying and retrieving profiles. For the three stages of interest, handling the service dependency of the profile seems relatively straightforward. However, handling user dependencies can become quite complex if self-tailoring mechanisms are provided.19

Accessing previously accessed services

Given that a service has been accessed before, its subsequent access can be simplified through minimizing reentry of previously supplied information via: (i) user and system profiles automating service initialization, and (ii) expert assistance to eliminate the need for explicit user concern with service acquisition, initialization, and termination. Although this reduces the need for entering previously supplied information, it will probably not eliminate it. Consequently, a 'soft user interface' is desirable to accommodate more flexible input than is presently acceptable.

The preceding can be summarized in our assertion that the key network access support components are: (i) profiles for users and services to simplify and automate service acquisition, initialization, utilization and termination; (ii) soft user interfaces to eliminate the need for concern with syntactic 'nits'; (iii) expert assistance to reduce the need for entering repetitive information during the acquisition, initialization and termination stages of accessing previously accessed services; and (iv) dynamic tutorial assistance to provide specific, pinpointed problem solving information online.

The next four sections summarize existing information on the four support components. It should be noted that the soft user interface, expert assistance, and dynamic tutorial assistance require a knowledge base which is provided in profiles. The order in which these components are discussed corresponds to a logical progression providing increasing support to the user (see Figure 1). However, the boundaries between these support functions are not precise and often overlap.

Profiles

The word "profile" implies an outline or representation of some entity. In access technology a profile is used to represent or characterize a user or service. Several groups have investigated the area of profiles and the information to be maintained therein.10,19 Since this is an evolving subject area, a succinct list of profile contents is lacking. However, a minimal list of required entries can be suggested.
The information in a profile parameterizes both the system and the user for all stages of service access: acquisition, initialization, utilization, and termination. Thus, the profile provides the means to free the user from remembering and executing the mundane, trivial portions of service access. As a specific example, consider the user of multiple systems—each system requiring different connection protocols, passwords, identifications, etc. The profile can be used to store the value of these variables for each system and user, thus relieving the user from remembering these details.

The information stored in profiles is of two different types: static and dynamic. Static refers to entries that are basically unchanged or change very infrequently. The profile entries for acquisition, initialization, and termination contain static entries. Dynamic refers to information which is typified by change. Profile entries associated with service utilization are dynamic; for example, although one may use an information retrieval system daily, it would be expected that key words would not remain the same from session to session. Another example of dynamic profile entries is a function of the access system design rather than the target system. If an objective of the access system is to characterize the user (e.g., experience level) or the use of a system (e.g., certain sequences of commands are habitually repeated), these characterizations are dynamic in nature.

**Static profile entries**

We know that certain static entries are required for both the user and system profiles. It would appear that such entries are fairly well defined given the access system, target systems, and the communications environment.

**User profiles**

In the user profile the following entries are required to support the acquisition stage: passwords, user identifications, and account numbers of the user for all known target systems. For the initialization stage, the descriptors required for the target system are any default actions to perform following login to a target system (e.g., any commands which describe special characteristics of a user's terminal) or any default options to issue upon initialization of given services on a target system. If the target system is configured as part of a computer network, the requirement may exist to transfer files from another system to the target system as part of the initialization procedure. Any commands which should be issued prior to finalizing connection with a target system are within the scope of termination. Thus, the user may wish to have certain classes of files deleted prior to logout. If the target system is configured as part of a computer network, the user may wish for all files (or a subset of files) to be transferred to another system in that network prior to disconnection.

**System profiles**

In the system profile, there are two subcategories of profile entries associated with the acquisition stage: those relative to target systems and those relative to the communications subnetwork (if employed). For the target system, the login sequence is required; for the communications subnetwork, the connection sequence to specify the target system is required along with any required communications control commands.

The profile entries for the initialization stage reflect any requirements of the system prior to utilization: Examples are the syntax required to invoke services, and any identification procedures required prior to accessing given services. Profile entries for the termination stage are in two subcategories: the target system and the communications subnetwork. For the target system, the syntax for logout is required, and for the communications subnetwork, commands for closing a network connection are required.

**Dynamic profile entries**

While much is known about requirements for static entries in the user and system profiles, determination of appropriate dynamic entries is still an open subject for research. Nevertheless, two major generators of such entries can be identified. Entries corresponding to service utilization are likely to fall into the dynamic category. Further, characterizations of users and modes of access support utilization maintained by the access system require regular updating and, thereby, are likely to be dynamic. We conjecture that dynamic profile entries will be highly dependent upon specific services being accessed and the design of the access system.

**SOFT USER INTERFACE**

The soft user interface refers to those tasks which humanize the system or network interface to a service for the user. This section structures the functions to be provided by a soft user interface and discusses an approach developed by the Stanford Research Institute.

The functional capabilities provided by a soft user interface can be divided into three major categories: (i) assistance in entering individual commands; (ii) assistance in entering a collection of commands; and (iii) service identification as discussed above. Moreover, each of these categories can be subdivided into two major parts according to whether the actual command syntax is assumed to be fixed. If this syntax is variable, the problem of providing such an interface is closely related to the general problem of restricted natural language interfaces.

Assuming the reader knows the appropriate command,
assistance in its entry can be provided through command completion to minimize the amount of information which must be entered, spelling correction, help features to remind the user of the command syntax, and user-oriented diagnostics to facilitate identification of what went wrong.

Assistance in reentering collections of commands can be achieved through allowing the user to catalog command sequences generated in previous interactions with the system. This minimizes the amount of information which must be reentered. An approach to providing such a mechanism is described in the following section.

The preceding has described means for facilitating the entry of commands having rigid syntactic specifications. We now discuss how these restrictions might be reduced through a more sophisticated interface.

Restricted natural language interfaces represent a logical means for providing syntactically and semantically flexible commands. In this context, there is a spectrum of features which can be provided. Thus, the interface may query the user concerning goals and objectives in an attempt to unravel a nebulous user request. Moreover, the interface may use a profile system to facilitate self-tailoring in terms of verbosity and user idiosyncrasies. As this query facility is expanded it rapidly approaches the capabilities included in dynamic tutorial assistance discussed below.

As an example of the potential power of a restricted natural language interface, we briefly summarize some of the capabilities of a system called LIFER (Language Interface Facility with Ellipsis and Recursion), which has been implemented at the Stanford Research Institute. LIFER provides an automatic facility for handling elliptical (incomplete) inputs, a spelling corrector, a grammar editor, and language extension through the use of paraphrase. LIFER is implemented in INTERLISP and consists of two major components. One component is a set of interactive language specification functions which are used to define an application language which is a superset of a natural language. The other component is a parser which interprets natural language inputs to translate them into appropriate interactions with the application software.

Production rules are at the core of the LIFER language specification functions. The production rules are advertised to be easily modified and interactively tested. The intermixing of calls to LIFER for language definition, extension or modification with calls to the parser for utilizing the developing language system serves as an aid for language development. The processing of elliptical inputs is another aid to the language builder. This elliptical handling is provided automatically by LIFER and frees the system builder from incorporating such constructions in the application language.

For spelling correction LIFER makes use of the spelling corrector in INTERLISP. In the event that LIFER does not understand an input it prints "user-oriented" error messages to indicate what it does understand and suggests correctional steps.

EXPERT ASSISTANCE

Using an access system often requires that the users learn a new command language—the language of the access system. Generally, system developers provide a menu of existing aids; however, in order for the users to tailor those aids to individual needs, or to create new aids, the users must deal with the language of the access system. This indeed is an ironic consequence since it is the goal of access assistance to lift users above the idiosyncrasies of individual systems. Expert assistance refers to the capability of automatic generation of access system commands to achieve the desired user actions.

Figure 2—Levels of expert assistance

There are five increasing levels of sophistication for expert assistance; however, the underlying commonality is that the user informs the expert assistance system of the points at which observation of the user's actions is to begin and end. Figure 2 represents the hierarchy of functions which can be provided in an expert assistance system.

The minimum level of access assistance is provided through a capability to simply record interactions which actually occur between a system and user, and produce the access system command sequences required to replay the recorded interactions. This level of expert assistance is not flexible because it does not handle variations in system response or minor (parameter setting) changes in the user commands. Greater sophistication is desired.

The second level of expert assistance permits identifying certain fields of the user command or system response as variable. Such identification can either be explicitly provided by the user or by the expert assistance system based on its knowledge about commands and services. The identified fields can then be incorporated as parameters in the single command entered by the user to invoke the parameterized sequence of commands entered earlier. As an example, the expert assistance system might recognize the invocation of an editor; given that the syntax for the editor is known, the location of the expected file name would also be known and could be flagged as a parameter.

The major drawback of the second level is the frequent occurrence of unexpected responses common to a networking environment. The remedy is to incorporate handling of
unexpected responses in the assistance system—the third level. Two approaches to their handling can be identified:
(i) a priori identification of all possible responses together with incorporation of appropriate actions; and (ii) notifying the user when an unexpected response occurs and permitting the user to specify the next action to be taken.

At the fourth level of sophistication, the expert assistance system draws upon its own knowledge base of services, and uses this information to enhance the generation of command sequences. For example, the expert assistance system may automatically generate appropriate responses to handle system level messages which occur during an interaction and should be ignored rather than generating an error. This level differs from the third level because the expert assistance system dynamically expands its knowledge base rather than having given situations identified at design time and programmed into the system.

At the fifth level of assistance, the user would be able to ask the assistance system to create the specific commands by indicating a general class of activity and a target system.

Currently, expert assistance is viewed as an option available to the user. There is no intent in the overall design to have the expert assistance system suggest to the user that certain activities are being repeated and could be incorporated into a command to the access system; nor is there an intent for the expert assistance system to suggest to the user that there are more efficient ways to achieve the goals being pursued. These capabilities are beyond the current scope of expert assistance. However, there is a sixth possible level, optimization, which may prove desirable as a future goal of such assistance. (Such support mechanisms have been considered for provision as part of a military message processing system.)

An implementation approach

The NBS Expert Assistance System (EAS) is a logical extension of the existing Network Access Machine. The implementation currently under way will accommodate field flagging (by either the user or the EAS if the information is contained in its database) and adaptive accommodation to unexpected system responses.

To accomplish these objectives, the EAS initially records all characters exchanged between a user and the system as demarcated by the user's overt initiation and termination of the recording session. A translator is then invoked to transform these character strings into proper NAM macros which may be called and expanded at any time. Variable fields are initially accommodated through flagging via a control character. Subsequently, as the NAM knowledge base is implemented, automatic recognition and flagging of variable fields (parameters) will be provided. As identified in Section 6.1, there are two approaches to handling unexpected responses. In view of the evident impossibility of foreseeing all possible system responses, the approach being adopted at NBS is the second (user intervention) coupled with preprogramming of some 'expected' unexpected responses. Thus, the EAS will have a minimal learning capability sufficient to permit incorporation of unexpected responses when they occur and, thereafter, to automatically handle such responses.

Usage of an expert assistance system by expert users constitutes an interesting, potential application. Users who are "expert" in the use of specific services on specific systems use the expert assistance system to generate the appropriate commands in the language of the access system to automate access to services for endusers. These "expert" users may be from a variety of areas—information retrieval, data base management, text processing—and are enabled by the assistance system to sit down and quickly create a library of commands to support predictable requirements of end-users. The "expert" users are shielded from learning the command language of the assistance system; they simply sit at terminals and enter the commands specific to their services. Further, the endusers are shielded from learning the command language of the services they wish to access and from learning the language of the assistance system.

A natural enhancement of an expert assistance system would be building access aids in the language of the access system interactively with the user. Thus, while the system is totally responsible for providing aids to support the inexperienced user, aids might be cooperatively generated while interacting with the more experienced user. Such a mechanism is discussed in the next section describing a highly interactive system in which the user is always being guided through the available alternatives.

DYNAMIC TUTORIAL ASSISTANCE

A dynamic tutorial system provides for user/service interactions controlled by the access system. The purpose of the tutorial system is to guide the user through all possible classes of activity offered by an access system. The tutorial system continually prompts the user and takes an appropriate course of action dependent upon the user's action. As an example of the use of the tutorial system consider the sample interaction below. (The italicized portions are generated by the user and the other portions are generated by the tutorial system.) This example terminates in the invocation of the required access system commands to login to the sending system, and transfer a file to the receiving system. The profile system contains the default values for the sending and receiving systems. If the user elects to use other than the default systems, the user may then have the profile updated with the new values at the end of the session.

File Manipulation? yes
Type of Activity? transfer
Sending Host is HOSTA? yes
Receiving Host is HOSTB? yes
Filename? TESTER

File transfer complete.

At this point the transfer is completed. This is of course the simplest sequence of events. Either the sending or receiving system could be other than the default ones; if so, the tu-
tutorial system may have to request user numbers and passwords for the new systems if they are not already known.

The essence of the dynamic tutorial assistance system is that the system is in command. It guides the user through all activities provided by an access system and executes the required commands to achieve the user goals. Additionally, the user could have the option to have the tutorial system catalog specific interactions so that the user may invoke them directly at a later time. Notice that this procedure implies a cooperative effort between a dynamic tutorial system and an expert assistance system. Using the above example, the user could invoke a "transfer" command which would result in the transfer of a file from one system to another.

SUMMARY

This paper has addressed issues relevant to implementing access support at the network level. An overview of the area of network access was presented, related research efforts were identified, factors which complicate network access support were identified, and the major components of network access support were structured. A specific implementation for one of these components, expert assistance, was described as it is being constructed at the National Bureau of Standards.

Our development of a structure for network access support began with the observation that user categories and access support difficulties preclude the existence of a single general support mechanism. Further, if the objective is provision of support via a network level group, only the acquisition, initialization and termination stages are reasonable candidates. This reflects the reality that service utilization requires intricate knowledge of the implementation of a specific service on a given system; it is unreasonable to expect one group (namely, the developers of a network wide access mechanism) to possess that level of knowledge for all systems on one network—much less across network boundaries.

Although support of service utilization cannot be provided at the network level, it is appropriate to provide general purpose support mechanisms. To accomplish this, an expert assistance system was introduced which can partially automate tailoring general purpose access mechanisms to meet individual objectives. Such a system shields users of access support systems from the command languages of those systems, while enabling them to construct procedures to access services on target systems.

ACKNOWLEDGMENT

The authors would like to express their appreciation to Rob Rosenthal and Sig Treu for their helpful comments and discussion of network access support.

REFERENCES


23. ARPANET Resource Handbook, the Network Information Center, NIC 39335 Augmentation Research Center, Stanford Research Institute, Menlo Park, California, December 1976.

