USER-ORIENTED NETWORK: A SERIES
PART IV. A SUGGESTED SET OF NETWORK COMMANDS

Wm. Orchard-Hays

July 1975 WP-75-84

TABLE OF CONTENTS

PURPOSE AND CRITERIA

In the discussions in Parts II and III, several commands and their mnemonics were invented. This was done as occasion required without any overall description of style or extent. Before proceeding with further discussions of possible network facilities and features, it seems appropriate to define a more comprehensive set of commands in a consistent manner.

Any invented vocabulary is certain to be received with objections and disagreements. The usual result is that each group of system--or even subsystem--designers invents their own nomenclature. It is precisely this variety of terminology which creates many of the problems listed in Part I. If a truly integrated and user-oriented network is ever to be a reality, a process of standardization must begin. It cannot be done all at once but, at least in new top structures, there can be a standard command language which has enough scope and flexibility, both to accommodate various requirements as they exist (by use of appropriate translation and interfaces) and to provide the pattern for true standardization in the future.

I have no particular fondness for the mnemonics used in the sequel. I have chosen them by two criteria: the appropriateness of their English meaning for the function intended, and non-conflict with similar functions already in wide use. These criteria sometimes conflict and, on the other hand, sometimes the same mnemonic can be used at two levels without conflict or ambiguity. (See use of LOGIN and LOGOFF in Part III.) If others have better suggestions for command words, they are encouraged to put them forward.

The criterion of English meaning is troublesome. It would be nice to have international words and perhaps such words can be found or invented and agreed upon. However, the need for clear meaning is also important and it is often difficult to make exact translations.

I feel somewhat stronger about the meanings intended. It has too often happened that a specialized word has been generalized, or vice versa, leading to further difficulties in expanding a vocabulary. A classic example of this is the word "file" which now connotes many different concepts. The word seems to have no general equivalent and yet its use can be misleading, even in a fairly narrow context. Some other examples of words used ambiguously or in specific or hierarchical senses with no obvious rationale are:

LINK, ATTACH, DETACH, ACCESS, RELEASE, DELETE, ERASE, LOAD, USE.

Even experienced users often find it necessary to look up these words in order to determine the exact function which they trigger, even though any one of them, taken alone, would seem to be a fairly definite verb.

CLASSES OF FUNCTIONS WHICH MUST BE DEFINED

Before launching into a vocabulary list, we should try to classify the various functions which must be activated. The following list of classes seems necessary, and perhaps sufficient.

1. Commands and protocols relating to logging in to the network and off it.
2. Commands and protocols for gaining access to computing systems, inputting data files, controlling distribution of results, and terminating execution.
3. Commands for communicating with other users or groups of users, on a message, mail or bulletin basis.
4. Commands for transfer of files within the network.
5. Commands for accessing and querying information systems.

6. Commands for querying the network about systems, users, data banks and other facilities currently on line.

7. Commands for very remote accesses via gateways and other temporary connections. (May be restricted to use by network monitors.)

8. Commands for operating the network. (Possibly restricted use.)

9. Commands for updating network files, control programs, translation routines, etc. (Restricted to maintenance accounts.)

10. Commands for operating network accounting procedures. (Very restricted usage.)

RECOMMENDED COMMAND STYLE

Most Indo-European languages seem to have been invented by people who think backwards. This carries over into customs and conventions. Consider, for example, the American way of addressing a letter:

Mr. and Mrs. John Jacob Jones
Apt. 5B
1234 S. Fifteenth St.
Sometown, Co. 89102

The only thing of immediate interest to the receiving post office is the zip code, written last. The city and state serve no function at all unless the zip code is unreadable. Only the local post office at the destination cares about the number and street (in reverse order), and the addressees' names are only tentatively useful, in case they have moved or are visiting or other people have the same apartment. Sentence order in Western languages is also strange; it seems designed to emphasize nouns and to either muffle the verb or delay its announcement as long as possible. All this is wrong for a command language.

In a command, the primary word is the verb. It should stand first followed by whatever arguments, modifiers or subcommands may be necessary. When the verb is some form of "to be," then it need not even be stated and the predicate adjective plays the role of a verb. For example, in the sentence "The system is unavailable," only the word UNAVAILABLE is important. Even if something other than the system is unavailable, it doesn't make much difference; we can't get at the system we want.

Command statements should also be structured so that natural "drop-off" is possible, unless this makes the full form grotesque. For example,

COME BACK AFTER YOU FINISH THE JOB
COME BACK AFTER YOU FINISH
COME BACK AFTER(WARDS)
COME BACK (if right tone of voice)

might all convey the same meaning in the right context. Arguments which are order-dependent should be arranged with this principle in mind. Of course, there are statement forms where arguments should not be order-dependent.

It may be objected to these remarks that more "natural-language-like" forms should be used. My reply is that the almost universal tendency of continual users of interactive systems is toward more and more abbreviated forms. (In fact, many allowable abbreviations are too cryptic even for me.)

It is perhaps time to begin constructing a recommended vocabulary without further gratuitous remarks. We will take the classes in the order listed above.

Logging In and Off the Network

The commands introduced in Parts II and III seem to be adequate for this class. We summarize them here. Every terminal must be connected to either a GRP or a CON which also plays the role of GRP.

(from GRP)	GROUP xx IN OPERATION
(from TER)	LOGIN userid

(Note: lower case letters denote values provided by the user.)

(from GRP)	PASSWORD
(from TER)	password (typing will masked in some way)
(from GRP)	(any mail, bulletins or special conditions are typed at the TER. This is followed by a prompt which will be denoted by > .)
.	
.	(user session)
.	
(from TER)	> LOGOFF

The only commands are LOGIN and LOGOFF. Possible alternates are:

for LOGIN:	LOGON, ENTRE
for LOGOFF:	LOGOUT, EXIT, FINIS

It is usual to allow L as an abbreviation for LOGIN and LOG as an abbreviation for LOGOUT. It is an open question whether abbreviations should be honored at the network level. (I am somewhat opposed but suspect many users will insist on it.)

Accessing Computing Systems and Peripheral Operations

There was a fairly lengthy discussion of these functions in Part III. The forms introduced there will be used here but they will be defined more precisely and the set of them extended. Square brackets denote optional arguments. (Other conventions are explained as encountered.)

1. HOOKUP Command

HOOKUP system-id [AS user's-system-id]

where	system-id	is the network name for the system desired
	AS	is a keyword for the optional phrase

user's-system-id   is the userid by which the user is known to the system.

If the optional phrase is not used, the user's network userid and password will be used to access the system. If the optional phrase is used, the GRP will respond with PASSWORD: and the password known to the system must be furnished. The reason for separating the password is so it can be masked.

The HOOKUP command causes the network to automatically perform the log-in to the system for the user. If all units are in operation and available, the next response will be from the system requested, as if its log-in procedure had just been completed. If the system cannot be accessed for any reason, the network will return an appropriate explanatory type-out.

If the HOOKUP is successful, all further type-ins by the user will be routed directly to the system, unless the ATTN button (break key) is pushed first. The ATTN button will be denoted by (attn).

2. <u>ATTN Command</u>

```
(attn)
GRP xx > ATTN [n]
```

The ATTN command simulates the break key to the system. First, the user pushes the real break key, (attn). The connecting group then responds with

```
GRP  xx >
```

indicating readiness to accept network commands. This same protocol is used for any network command while the TER is hooked up to a system and will be referred to as the <u>(attn) protocol</u>.

If ATTN is typed after the (attn) protocol, a real break character is sent to the system and the (attn) status of the GRP is cancelled. The next response will be from the System's ATTN level.

Some systems utilize two successive breaks for certain purposes. If more than one break character is to be sent to the system, the number can be indicated by the optional n.

If (attn) is used when no system is hooked up, the response is simply

```
GRP xx >
```

verifying that the GRP is still "alive." If ATTN is then typed, the response is a question mark, as follows:

```
(No HOOKUP in effect)
(attn)
GRP xx > ATTN
?
```

No harm is done; any legitimate command can now be typed.

3. RELOG Command

```
(attn) protocol
RELOG system-id AS user's-system-id
```

This command is valid only when a HOOKUP is in effect. If used at any other time, the response is a question mark. If a HOOKUP is in effect, the response is

```
PASSWORD:
```

and the password known to the system must be typed.

The purpose of the RELOG command is to permit logging off one system userid and logging on a different one without breaking the connection. Although some systems allow this within their own repertoire, this cannot be permitted without notifying the network so message-routing addresses can be changed. The RELOG command performs this function and the CON to which the system is attached will simulate whatever LOGOFFs and LOGINs the system requires. This logic need only exist in the control routine in this one CON.

4. System Log-off and Disconnect Commands

When a user is through with the system, he logs off with the system's appropriate command. The connecting CON is notified, corrects its tables and notifies the GRP. The GRP corrects its tables and causes the following type-out:

SYSTEM system-id UNHOOKED

followed by a prompt. This will also occur if communication is lost in some way which the GRP can detect.

If the system has a "disconnect" command, allowing it to continue processing without an attached terminal, the effect is the same to the network as if a log-off had been issued.

5. INPUT COMMAND

This command is not issued by the user acting as himself but as the userid for a RDR or possibly a TAP. (Remote tape units may require special provisions for transmission of data, for example, buffering logic in GRPs and CONs. A tape mounted on a unit is a SYS's own installation is controlled with system commands or RJE statements.) The following conditions must exist before an INPUT command can be used.

a) A RDR must have a network address and also a distinctive userid in the GRP to which it is attached.

b) The RDR must be turned on and readied.

c) Some person must LOGIN as the RDR's userid

The command is then issued as follows:

```
INPUT  system id  [FOR userid  [AS file-id]]
```

where	system-id	is the network name of the system
	FOR	is a keyword for an optional phrase
	userid	is the real user's userid
	AS	is a keyword for a second optional phrase
	file-id	is the identification of the input file to be recorded by the system.

The FOR and AS phrases are optional only in a predetermined sense which depends on the conventions of the system. In many systems, this information is provided in the first cards of the input deck, particularly for RJE. In interactive systems, there is a virtual RDR for each user and only the FOR phrase may be needed to direct the input to the virtual RDR. However, sometimes there is an option to input a file directly to a user's mini-disk; then the AS phrase may be needed.

The action of the network is automatic and similar to a HOOKUP. The system may make an acknowledgement response. It may also be possible to enter many files or RJE jobs with one INPUT command, or the RDR may be left running waiting a subsequent input deck. These details depend on the systems used, the nature of the grouping computer, and the characteristics of the RDR and other equipment. Information bulletins will have to be posted in user centers regarding these conventions and protocols.

When the use of the RDR is complete for the system, a standard network LOGOFF or a QUIESCE command can be issued. If it is desired to free the TER before use of the RDR is finished, a CARRY ON command can be issued. An (attn) protocol is required.

6. OUTPUT Command

Before an OUTPUT command can be issued, the same conditions must exist for a PRT (or a RDR/PUN if card output is desired, or possibly for a TAP) as described for a RDR for the INPUT command. Then the user, acting for the PRT's userid, issues the following:

```
OUTPUT system-id
```

Assuming all necessary units in operation and available, all files in the output spool of the specified system with destination labels corresponding to the PRT's userid will be printed (or punched if a RDR/PUN is used). The action of the network is automatic and similar to a HOOKUP. If the system does not disconnect when all files are output (most do not), the OUTPUT command remains in effect indefinitely.

When use of the PRT (or RDR/PUN) is complete for the system, or it is desired to interrupt output, either a LOGOFF or a QUIESCE command can be issued. If it desired to free the TER before output is complete, a CARRY ON command can be issued. An (attn) protocol is required.

A LOGOFF will stop operation immediately. The action of a QUIESCE depends on whether or not the network can detect the end of a single output file.

7. <u>CARRY ON Command</u>

This command is only recognized at a TER from which a RDR, PRT, RDR/PUN (or possibly a TAP) is being controlled. The GRP control program continues executing whatever INPUT or OUTPUT command was previously initiated (between one specified system and one remote unit) but disconnects the terminal as if it were logged off. An (attn) protocol is required. The sequence is simply:

```
(attn)
GRP xx > CARRY ON
```

The "ON" is only for readability.

If a LOGIN for the unit's userid is later attempted (from the same or a different terminal), the response is:

```
UNIT unit-id BUSY WITH system-id
```

In order to issue further commands, an (attn) is required.

8. <u>QUIESCE Command</u>

This command is only recognized at a TER controlling a unit for which an INPUT or OUTPUT command is currently in execution. An (attn) protocol is required.

The purpose of the QUIESCE command is to terminate a prior INPUT or OUTPUT command. An INPUT command should not be terminated until the entire file or files have been read in; otherwise an incomplete file will exist in the system which may cause all kinds of trouble with the job. The QUIESCE command will permit reading to continue until the end of the current file. It may be necessary to devise special inter-file cards which the GRP can recognize but which are not passed on to the system. This is probably a good idea anyway.

In order for QUIESCE to work properly for OUTPUT, it must be assumed that each individual file has an end-of-file indication which the network can recognize. This is true for many systems. If not, QUIESCE will simply have to interrupt output peremptorily. Most systems will re-output an entire file whose transmission was interrupted.

After QUIESCE has completed (which may take several minutes or more if a large file is in process), the connection with the system is broken and a prompt is given at the TER. A different INPUT or OUTPUT command can now be issued. No (attn) is required at this point.

The protocol of the command is simply:

```
(attn)
GRP xx > QUIESCE
SYSTEM system-id UNHOOKED
>
```

9. OMIT Command

This command is only valid at a TER controlling either a PRT or a RDR/PUN for which OUTPUT is in execution. Its purpose is to suppress actual printing or punching of a long file which is not wanted but which is being transmitted from an output spool. It must be executed while output is actually going on, that is, someone must see the output start and then issue OMIT. Action is immediate and the file is terminated at the device but emptied from the system. The form is simply:

```
(attn)
GRP xx > OMIT
```

If the system recognizes a similar command (sometimes called TERM), the connecting CON will issue this. Otherwise, the CON will keep accepting output from the system until the end of the file, but will not forward it.

(See also UNHOOK command under Operating the Network.)

Network Communication Among Users

This subject was discussed in Part II with respect to network topology and message forwarding. The viewpoint here is that of the user and the commands necessary to communicate with other users.

If the TER is currently hooked up to a system, an (attn) protocol is required for any command. This will not be further indicated unless some other special action is required or implied.

1. SEND Command

This is the command usually called MESSAGE or MSG in existing interactive systems, but these mnemonics are purely English, even American, and are slightly misleading in any event. What is really meant is to send a single typed line to another user or a group of users who is (are) on-line.

```
SEND userid message-text
```

If a single user is specified and he is not on-line, a message is returned:

```
userid NOT ON-LINE
```

If he is on-line, the message-text is printed at his TER.

If multiple users are specified by a conventionalized userid (see Part II), the message is typed at TERs for all who are on-line. No notice is returned as to how many received the message.

2. POST Command

This function is often called a "mail-box" (another Americanism). Only a specific userid is allowed. The message may be up to some prespecified length (perhaps 256 or 512 characters). Multiple lines are accepted; the message is terminated by some convention such as two slashes (//), three dots (...) or a blank line.

```
POST userid message-text
(continuation lines, if needed)
(termination convention)
```

If the user is on-line, he is notified that there is post for him. If not, it is held some prespecified length of time (say 48 or 72 hours). If he logs in during that period, he is notified. The TYPE command is used to have the post typed out at his TER.

3. NEWS Command

This command is for news bulletins which are to be available for some specified period of time, to all users of either a GRP, a CON, or the network. It may be fairly lengthy and is printed only on request. Whenever a user logs in during the effective period (or if he is on-line when the news is promulgated), he is notified that there is news. He uses the TYPE command to have it typed out.

The complete format of the command cannot be specified without further design decisions being made. However, it will presumably look something as follows:

```
NEWS user-group-id  news-id  time-period
(text of news bulletin)
(termination convention)
```

This command should be restricted to network monitors.

4. TYPE Command

This command is used to cause specific information to be typed out at a terminal. It will probably have a large number of options. The two of interest for commands thus far defined are:

```
TYPE  POST
```

and

```
TYPE NEWS news-id
```

Transferring Files Within the Network

In addition to the facilities provided by the SEND, POST and NEWS commands, it appears necessary for the desired effectiveness of the network to be able to transmit regular files between users, or at least between user centers. This is no great complication to the facilities already described, except for one question: where do these files reside? In trying to answer this, one is led into another question: what kind of files is one concerned with?

In any SYS, there will be numerous files which are more or less specialized to the system. For example, an assembly language source file for an IBM 370 will be of no value to a user on a CYBER 70, and an EXEC file for VM/CMS on a 370 will be of no use to someone using a different system, such as TSO, even on another 370. If two users employ the same system in the same SYS, then system facilities can be used for sharing or copying files. But even these cases cannot be ignored, for, if a user has a good 370 assembly language routine in, say, the Pisa center and someone in, say, Sofia would like to have it for a program running in batch mode on a 370 there, the network should be able to provide the communication.

Of course, there are many other kinds of files. There may be data input tables, records from an information system, or selective output from data banks, which someone has accumulated at great effort and wishes to send to another area. If such things are to be possible, users must have some permanent storage capacity in the network itself, not just in a SYS which can be hooked to the network. The natural place is in the GRPs so that normal usage of such files would involve the least telecommunication.

It is an open question whether a full context editor with all the attendant file manipulations should be provided by the network itself. Good facilities of this kind are available in several systems. They are expensive to duplicate and require considerable storage to use. However, some simple file editing is probably necessary. No attempt will be made here to define commands and subcommands for this purpose. The EDIT facilities of VM/CMS and TSO can be recommended as good patterns; actually, the expanded version of EDIT under CP-67/CMS was superior, if documentation can be found for it.

Assuming that the network has local capacities for user files, then a set of file-manipulation commands are needed, independently of a context editor. A file-identification scheme is also needed. Again, the CMS scheme can be recommended; I have found it extremely useful and flexible. Each file-id consists of three parts:

filename	(8 chars)	assigned completely at user's discretion
filetype	(8 chars)	can be arbitrary but specific mnemonics have predefined meanings (e.g. ASSEMBLE, FORTRAN, TEXT, EXEC, MODULE, etc.)
filemode	(2 chars)	designates attached mini-disk and access restrictions (e.g. A1 means user's primary disk, read/write to user, read-only to others)

The CMS filemode does not appear very useful for network use although an access key should be provided. However, something similar representing the owner's userid is necessary. Reference can be made to all files of a certain filename, or a certain filetype, or a certain filemode, or combinations. Such references are used for listing, erasing, or constructing another executable file (file-type EXEC) which can be used for various kinds of manipulation of the subject files. The extent to which such provisions are meaningful depends on how elaborate the network file-handling design is. No attempt will be made to determine that here. However, in the following commands, "file-id" should be understood to mean whatever file identifier is appropriate, possibly concatenated symbols.

A few commands will clearly be required if any kind of user-file facilities are provided. The following are suggestions which may have to be modified considerably, depending on overall file system design.

1. GIVE Command

This command gives permission to some (other) user to copy a file. The to-user need not be on-line at the time. It remains in effect until the file is successfully copied by the other user.

```
     GIVE   to-userid   file-id
```

This command gives the owning user complete control over who copies his files and obviates the need for complicated accessing codes and keys. However, it may be desirable to also have a "public access" key which allows anyone to copy a file without a prior GIVE.

2. <u>COPY Command</u>

This command copies a file from another user who has given permission with a GIVE command (or which has a public access key).

```
                                        ⎡AS  new-file-id      ⎤
     COPY   from-userid   file-id       ⎣TO existing-file-id  ⎦
```

If no optional phrase is used, the original file-id is modified to correspond to the to-userid with the same status. If the AS phrase is used, the file-id is changed as specified; if an old file by that name exists, it is first deleted. If the TO phrase is used, the file is added to the end of the specified file; if it did not exist, it is initiated.

3. <u>ERASE Command</u>

This command erases an existing file from the owning user's storage.

```
     ERASE   file-id
```

If concatenated file-ids are used, this command may take a conventionalized file-id which means all files of a class. For example, SCRATCH-* might mean all files with filename SCRATCH, regardless of filetype.

4. <u>LIST Command</u>

This command lists all files in the owner's storage whose file-ids match the conventionalized file-id specified.

```
     LIST   file-id   [TO file-name]
```

If the optional phrase is used, the list is not typed but formed into a new file with specified file-name and appropriate type (such as EXEC). The new file may itself be typed (with TYPE command and a new option) or possibly used as an executable file depending on facilities provided.

5. COMBINE Command

This command appends one or more existing files to the end of another existing file. The appended files are not erased.

```
COMBINE to-file-id  file-id-2  [file-id-3  ...]
```

If filetypes are used, some rule must be defined for the case that all files are not the same type. The usual rule is that filetypes must be the same. Otherwise record length may be ambiguous.

6. EXTRACT Command

This command extracts specified records or lines from an existing file to form a new file which is a subset. It may have to be restricted to certain types of files, depending on file/record/blocking arrangements.

```
EXTRACT  new-file-id  FROM  old-file-id
         first-line-no.  last-line-no.
```

If "line numbers" are not appropriate, some other option may be provided; perhaps "from" and "to" character strings, etc. However, if such a scheme is carried too far, it quickly assumes the complexity of a context editor.

7. RENAME Command

This command changes the file-id of an existing file in the owner's storage. If concatenated file-ids are used, it is customary to define conventionalized file-id parts to mean "same as" and "all."

```
RENAME  old-file-id  AS  new-file-id
```

Additionally, the INPUT and OUTPUT commands need additonal options to permit a file to be read in from a RDR to a user's storage, from a user's storage to a PRT or RDR/PUN, and from a user's storage to a SYS or vice versa.

Accessing and Querying Information Systems

It is not possible to be as definite about this class of commands as even about the previous class. Many design, and even policy, decisions must be made before one can be very specific. In any event, the majority of commands in this class are not properly network commands but parts of particular repertoires for various facilities. It is possible to state broad areas which such facilities might be designed to serve.

Context editor:	This was discussed briefly in the prior section.
Scratch pad:	Although somewhat different in scope and usage, this type of facility is similar to a context editor (or probably includes one) and must be considered independently from general network commands.
Information Retrieval System:	My opinion is that a useful system of this kind can only be implemented on a large computing system and that therefore its usage would fall under accessing SYSs.
Artificial Intelligence:	My further opinion is that this would be in the nature of a research project which would utilize an information retrieval system as a tool. Hence it is even farther removed from standard network commands.

About all that can be said here about any of the above is that some command is necessary to hook up to or access the appropriate facility. Once this is done, one is either working with an extensive sublanguage or is interacting with a complete SYS.

There is one other possible kind of information system which would involve the network, per se. This is Computer Assisted Instruction (CAI) applied to the network itself. The purpose of such a scheme is to allow a complete neophyte to sit down at a terminal and, with only the barest of prior instruction, to be guided through the various facilities and conventions of the network, and prompted to ask questions meaningful to his interests. Although this all sounds wonderful, there are several practical objections:

1. An adequate CAI package is difficult and tedious to prepare, requires much debugging and modification, and is expensive (particularly in storage space) to operate. If it is stored centrally, then the communication is expensive.

2. A neophyte doesn't stay a neophyte long. Either he becomes proficient in the facilities he wants to use, or he quits.

3. Anyone who is likely to use a network can be given a 15-30 minute briefing in his own language by an experienced colleague. Some of this is necessary anyway, such as the logging-in protocol. "Playing around" with the local network for an hour or so will teach him more than a formal discussion, and he needs this practice anyway. (He really can't do any harm.)

4. Standard network facilities fill the need for disseminating specific information or answering immediate questions. Printed manuals are necessary in any event and are actually much easier to use than a programmed set of questions and answers.

It is rather obvious from the foregoing that I recommend against a CAI facility for the network itself. Preliminary instructions which can be typed out on request may be appropriate for specific facilities, such as a scratch pad.

Network Query Commands

In contrast to the negative and somewhat vague tone of the prior section, in the area of query commands, it is possible and desirable to be much more specific. It is perfectly reasonable and even essential that a user be able to get immediate specific information on the status of the network. Such a service is not at all difficult to provide. Its operational expense is almost solely in the transmission costs for messages, but by its nature this is unavoidable.

There really needs to be only one command for this service, and the natural mnemonic is QUERY. This command would activate a subroutine common to all GRPs which would recognize various subcommands and arguments. These should include those shown below but many more would undoubtedly be defined as the network evolved.

	Responses
QUERY USERS	number of users now on-line at local GRP
FILES	number of user's files, space use, space still available
userid	ON-LINE AT TER-id / NOT ON-LINE / RUNNING DISCONNECTED
GRP-id	CONNECTION UP / CONNECTION DOWN
CON-id	CONNECTION UP / CONNECTION DOWN
SYS-id	CONNECTION UP, NOT IN USE/AVAILABLE/OVERLOAD CONNECTION DOWN
TIME	time of day and connect time this session.

The SYS-id option should be generalized to any facility available on the network with appropriate modifications to the responses. Other kinds of options might include the current setting of the TER (line width, prompt mode, blip character) status of INPUT and OUTPUT commands, post and news items, etc. Many of these options are required anyway for debugging and operating the network, recovering from communication interruptions and other disasters, etc. They simply need to be programmed in the GRP control routines so that they are easily available to all users. Some, of course, may be restricted to a particular class of user.

Accesses via Gateways and Temporary Connections

It is very difficult to predict what kinds of commands may be required here. Most likely, some general command or commands should be defined which only trigger the standard

GRP control routines to load and execute some special subroutine or send a special message to an appropriate CON. Since a GWY is also a SYS, it is possible that standard network commands and features plus the standard system repertoire may be sufficient for remote access to another network. A similar problem may exist with data banks existing in systems once-removed from standard network facilities.

For temporary experimental hookups, one would hope that all the special functions would be programmed in the experimental unit to interface with standard network protocols and conventions. However, special situations are sure to arise.

It would appear that some sort of manual mode to trigger normally automatic network functions may be necessary. About all that can be said at present is that these considerations should be kept in mind when programming control programs for CONs and GRPs. Since these situations will be common during the initial implementation of the network, perhaps the matter will take care of itself with only a little attention to generalization of "bootstrapping" functions, which are always necessary in the early stages of implementation.

Commands for Operating the Network

It has already been suggested that network monitors will be required, certainly at CONs and probably at GRPs. This function would not be a full-time job for most monitors but only for a small number, say at IIASA and a few key points. Most likely these monitor-users would be systems programmers who developed the control programs. Some coding scheme must be built into their userids and/or passwords which gives them access to commands and options denied ordinary users. A breakdown in communication lines between two CONs, for example, may lead to situations which are unresolvable without manual intervention which would itself be dangerous to the integrity of the network in normal operation.

A full set of commands for operating the network cannot be defined until a number of design decisions have been made. However, the list below--which should not be regarded as complete--indicates the sorts of commands which will be needed. Many of the mnemonics have been taken from IBM's VM/370 CP commands, with the meaning adapted as appropriate. It does not seem possible for me to improve on their choice of words in these cases; perhaps others will disagree and have better suggestions.

1. <u>ATTACH Command</u>

The purpose of this command is to define internal referents for real devices. For example, a particular TER may be defined as the monitoring console, a particular printer may be assigned a certain internal userid, etc. Probably a standard set of such definitions will be loaded at start-up time, but provision for changing them must be made.

$$\text{ATTACH device To} \left\{ \begin{array}{l} \text{CONTROL AS device-addr} \\ \text{userid AS device-addr [R/O]} \end{array} \right\}$$

2. <u>DETACH Command</u>

This command is the reverse of ATTACH.

$$\text{DETACH device from} \left\{ \begin{array}{l} \text{CONTROL} \\ \text{userid} \end{array} \right\}$$

3. <u>DEFINE Command</u>

It may be necessary to define which communication line goes to another node, such as a superior or coordinate CON, or a subordinate GRP.

```
DEFINE  line-id  AS  node-id
```

One option for "node-id" should be DEAD

4. <u>ENABLE Command</u>

When all is in readiness to use a communication line, the local control program must be notified of this. This command and the next are for this purpose. It may also be applied to local disk drives, etc.

ENABLE line-id

5. DISABLE Command

DISABLE line-id

If the line is busy, execution of DISABLE should await a graceful break.

6. ECHO Command

The purpose of this command is to test network facilities. It is like a SEND command except the final output is inhibited and instead the message is routed back to the sender.

ECHO userid test-message

7. HALT and RESUME Command

It may be necessary to be able to instruct the line-transmission routines to suspend transmission temporarily without actually disabling or detaching the line. Commands for this purpose and then resuming transmission might be as follows:

HALT line-id
RESUME line-id

8. SET Command

A variety of local network and device parameters will need setting from time to time. Some of these may be permitted to any user, such as the line width he wishes used at his terminal.

SET (various parameter keyword phrases)

For keyword phrases, I favour the form: keyword = value, for example:

SET LINESIZE = 120

A series of these phrases can be strung together, separated by commas.

9. MONITOR and AUTO Commands

In tracing down trouble or testing proper performances, it may be necessary to visually monitor all messages over a certain line or to and from a specific terminal.

$$\text{MONITOR} \quad \left\{\begin{array}{l}\text{line-id}\\ \text{userid}\end{array}\right\}$$

This command causes all messages over the specified line or to and from the userid to be typed at the monitor's terminal. This includes network-generated messages. The following command stops this:

$$\text{AUTO} \quad \left\{\begin{array}{l}\text{line-id}\\ \text{userid}\end{array}\right\}$$

10. RESET and UNHOOK Commands

When line failures or other network malfunctions occur, internal tables may be left in a state of impasse. Someone may have to peremptorily undo them. The RESET command will cancel all implied references to a particular line; the UNHOOK command will log off a system, whether or not the normal network actions are possible.

```
RESET   line-id
UNHOOK  system-id
```

RESET is primarily for use at the GRP level and UNHOOK is exclusively for use at the CON level.

11. LOCATE, DUMP and PATCH Commands

When serious troubles occur or special actions are required, a system programmer may have to look at tables in real storage of the controlling computer, and possibley patch in new values in octal or hex. Commands for this purpose are necessary.

```
LOCATE  various-ids
```

This command locates the desired table or block in real storage and prints out its real machine address and extent.

DUMP start-loc end-loc (or equivalent)

This produces an on-line octal or hex dump. An option for printing a long dump off-line should also be provided.

PATCH real-storage-loc machine-values

This command inserts actual octal or hex values into real storage.

These commands may already be provided by the basic software for the computer. However, they must be available in a real-time environment.

12. SLEEP Command

The monitor may desire to leave his terminal logged in and able to type out messages, without being in a normally active state. The SLEEP command permits this; the connect time continues, but the userid has no processing unless a message is received at the GRP for him.

SLEEP

To "awake" the TER, an (attn) is required.

13. SHUTDOWN Command

Clearly an automated procedure for a graceful shutdown of the local facilities is required. This should be a single word

SHUTDOWN

but restricted to a single monitor's userid.

Commands for Network Maintenance and Accounting

While it is clear that commands for network maintenance and for accounting functions will be necessary, it is premature to attempt to define them now. It will be necessary to take considerable care that these do not become specialized to particular centers or computers. If the computers used for CONs and GRPs differ from place to place, then, of course, programs written for them will necessarily differ in detail though not in function. In this case, local maintenance procedures are bound to differ, also. A clear distinction must be maintained between coding, compiling, loading, etc. for a particular computer and true network functions which such routines implement. With regard to accounging, security provisions must be made as machine-independent as possible.