ARPA ORDER NO.: 3460/3681
9P10 Information Processing Techniques

N-1250-ARPA

August 1979

A NETWORK GRAPHICAL CONFERENCING SYSTEM

Michael T. O'Brien

A Rand Note

prepared for the

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under Contract No. MDA903-78-C-0029.

Published by The Rand Corporation

ARPA ORDER NO.: 3460/3681
9P10 Information Processing Techniques

N-1250-ARPA

August 1979

A NETWORK GRAPHICAL CONFERENCING SYSTEM

Michael T. O'Brien

A Rand Note
prepared for the
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

PREFACE

This Note describes the design goals for a current Rand project in the design of interactive software. It is being published to solicit feedback from the computer science community. A more formal report will follow within a year, setting forth the final project goals as well as providing initial feedback from early use of the system.

This work is being supported under the ACCAT (Advanced Command and Control Architecture Testbed) program of the Information Processing Techniques Office, Defense Advanced Research Projects Agency. The Note is intended primarily for an audience with some experience in computer science, particularly in the area of computerized teleconferencing and computer graphics.

SUMMARY

A network-oriented, color graphic conferencing system is being built by the Information Sciences Department of The Rand Corporation. This system is designed to allow several people to confer simultaneously over a network. Users share a common display, which is divided into individual sections (or "windows") for each user, and a common "blackboard" where users may take turns sketching out and modifying a graphical display in full color. The display uses bit-map color raster-scan technology.

Each user's private window is identified with his name and location. Text typed by a user appears in his window as he types it. Several users may be typing text simultaneously. A special symbol appears in the window of a user who is sketching, to indicate who is drawing the picture.

Several conferences may be going on at the same time; users may request a menu page showing who is participating in each conference. Conferences also may be "locked" to prevent others from joining.

The system uses other Rand-developed utilities for the design and display of arbitrary characters, as well as Rand's Virtual Terminal UNIX (TM)* operating system.

*UNIX (TM) is a registered trademark of Bell Telephone Laboratories.

ACKNOWLEDGMENTS

The author would like to thank Gary R. Martins, who gave much sage advice in the design and implementation of the conferencing system and lent invaluable assistance in the preparation of this Note.

CONTENTS

PREFACE iii

SUMMARY v

ACKNOWLEDGMENTS vii

Section
I. BACKGROUND 1

II. DESIGN GOALS 3

III. ORGANIZATION 6
The Text Screen 6
The Graphics Screen 7
Access Control 8

IV. NETWORK CONSIDERATIONS 10
Network Protocols 10
Network Arbitration 11

V. SYNCHRONIZATION 13

VI. CONCLUSIONS 15

Appendix
A. ACCAT GRAPHIC ORDER CODES 17

B. TEXT STREAM PROTOCOL 22

REFERENCES 25

I. BACKGROUND

This research is loosely based on work done on the PLATO Project, at the University of Illinois at Urbana. The PLATO Project is somewhat unusual in computer science history in that the system design was specified in large part by its users; system development and utilization were concurrent.

Very early in the history of the PLATO Project it was decided that some form of interterminal communication was required that went beyond the facilities provided by the operating system (which has a feature somewhat analogous to the TENEX LINK or UNIX "write" facilities). Therefore, a program was written[1] which allowed users to communicate with each other in groups of six; the conference thus created occupied the entire user display, with each user controlling one-sixth of the display area minus a line for identifying information. While this program did not originate with the systems support staff, it eventually achieved a somewhat ambivalent status at which point the system would have been significantly poorer without it. This program, together with the link facility and the system mail utility, allowed users thousands of miles apart who had never seen one another to collaborate effectively on non-trivial programming projects.

ACCAT is an ARPA-funded research and development program sponsoring efforts at several network sites in the development and support of

advanced man-machine interface tools for command and control. Rand has traditionally broken new ground and developed new tools which were later transferred to research centers elsewhere.

The tools thus far developed for ACCAT permit the design and graphical display of complex military situations, but there is no facility enabling several persons to confer via the system about the interpretation of such displays. The history of the PLATO conferencing program, together with its success in mediating between system users, led to the consideration of its concepts in the design of a conferencing system for ACCAT.

II. DESIGN GOALS

Because ACCAT uses graphical entities such as military situation displays, any conferencing system which attempts to meet the needs of ACCAT users must support the manipulation of graphical entities. In addition, ACCAT users are distributed across several host machines linked via the ARPANET*. Hence, a network-oriented, color graphic conferencing system was designed. The system is not designed around any advanced theories of the psychology of conferencing, but rather is loosely based on another such system which has worked very well in the past. The color graphics capabilities are viewed as a natural extension of the text-oriented conference, in much the same way that the use of a blackboard is a natural extension of a face-to-face conversation.

In fact, one explicit design goal is to have no preconceived notions of what the system is supposed to accomplish or exactly how users will accommodate themselves to it. A sufficiently powerful and general set of primitives should allow each user (or user community) to develop its own set of conventions for using the system. In our view, a good conferencing system does not represent a utility, but an environment which the users should "inhabit" in whatever fashion best

*In the following discussion, "host" refers to a host computer on the ARPANET, and "site" refers to a host plus associated graphics hardware.

suits their needs. The parent operating system, UNIX, exemplifies this concept, and it has proven quite successful.

Within these rather broad parameters, the conferencing system is designed to support the following features:

- o Simple window-oriented text conferencing.
- o Ease in entering and leaving conferences.
- o A sketch-pad graphics facility, capable of being used either via a keyboard, via more advanced graphic input devices, or both.
- o Ease of switching between textual and graphic contexts.
- o Simple editing features for both text and graphics.
- o Use of user-defined special characters in graphics windows (such characters are designed using a utility separate from the conferencing system).
- o No centralized "umpire" node in the network.

Note that a large number of extra features might be added later, but they have not been included in the original design. Among these possible extensions are a method for storing pictures drawn during a conference for reference later in the same conference or in a different conference; generation of displays from an image file; and maintenance of a log of all text typed. These features may be added

later, if experience gained from actual use of the basic system indicates their desirability.

The facility will be used by people with little or no training in the use of computers. This does not, at least at present, imply a large on-line program of assistance. Rather, it implies a cleanly designed system which is easy to learn either by trial or by example. It is expected that, as with any new tool, some practice will be necessary before a new user becomes proficient with the system. If the system is correctly designed, that time will be tolerably short, and there will be a minimum of inconsistencies or "surprises" after the basics have been learned. As experience is gained from use of the prototype, the system design will be improved to eliminate deficiencies which users encounter.

The prototype design consists of the network protocol and the UNIX prototype implementation of the user interface. It is a primary design goal that the network protocol be a "lowest common denominator" of the capabilities expected of an implementation, and that it not be tied to any one type of graphics hardware or host operating system. The actual interface presented to the user in terms of display handling, editing capabilities, etc., is implementation-dependent.

III. ORGANIZATION

The purpose of the system is to enable several users at distributed network sites to converse, using both text and graphics. Two logical screens are used, each of which is identical for all users. One is a text screen, within which each user controls a window. The other is a graphics screen, whereon graphic displays may be generated by the users. These may or may not be implemented on the same physical device.

THE TEXT SCREEN

Characters typed by a user appear in that user's window as he types them and are sent over the network to the other users' hosts, where they appear on the other users' screens as well. Since each person sees the same display, several people may be typing at once without confusion; each user's output is restricted to his own window.

Experience on the PLATO system has shown that it is rare for more than two people to type at once. Most people seem to find it difficult to type and read simultaneously, though some experienced users alternate between the two so rapidly that they appear to be doing so. Each user finds his own mode for using the system.

There are certain metacharacters which the user may use to control his display. These characters are used to:

- Clear the user's window.
- Enter the graphics mode, in which the user draws a picture on the graphics screen.
- Leave the conference.

In addition, a particular implementation may have certain editing functions used to alter text that has already been typed; it is the responsibility of the local host to transform these requests into the network protocol, which supports a lowest common denominator of editing capabilities.

THE GRAPHICS SCREEN

The graphics screen is a logically distinct screen, regarded as being a color raster display system with 512 x 512 pixels. This size was not chosen because of any magical capabilities, but merely as being representative of commonly available hardware. The physical realization of this display is, once again, an option of the local site. The prototype implementation is on a Genisco color raster graphics system. When a user who is typing on the text screen issues the "graphics" command, he is given control of the graphics screen and proceeds to draw a picture.

The mechanics of actually drawing a picture are a key subject of research. Those sites with joysticks, light pens, etc., may use them

in a local implementation to draw the picture, modulated by keyboard commands or whatever other control technology the site may possess. As in the case of text editing, the actual network commands are a lowest common denominator graphics protocol, based on the Rand-developed ACCAT Graphic Order Codes. One interesting avenue being pursued is that of drawing pictures entirely with the keyboard. This requires that the keyboard be redefined as a graphics control keyboard, with line drawing, color selection, and other functions. This feature will permit sites that are poor in graphics input technology to generate pictures, albeit more laboriously than they would with graphical input devices.

A user may request the graphics screen at any time, and he will be given a cursor on that screen. He may use this to point at some object being discussed or to alter the display. Hence several users may cooperate in altering a picture. Also, this permits the commands for pointing at an object to be identical to those for moving around while altering a picture, thereby eliminating the need for two separate modes of cursor motion.

ACCESS CONTROL

A user may find out what conferences are in session by inquiring of each host known to support the conferencing system what conferences are current on that machine. This time-consuming polling technique is required because of the constraint that there be no central "umpire"

node managing all conferences (see "Network Considerations" below).

The information is displayed as a list of all conferences, their status, and the users in each. The list also notes conferences that have special status.

Unless positive action is taken otherwise, a conference is open to all who attempt to join it, as long as the upper limit of users in a single conference is not exceeded. However, users may, at their option, "lock" a conference so that no one else may join without permission. When a new user attempts to join a locked conference, he is notified that it is locked and is given the choice of waiting to see if someone will let him in or leaving immediately. If he decides to attempt entry, a request message is sent to all other conferees in that conference, and the user attempting entry is forced to wait until someone sends a permission message back. The message may never come. He may back out of this "waiting for permission" state at any time, cancelling his request to enter. These protocols are subject to change as experience is gained from the prototype.

A conference may also be "frozen." In this state, no outsider may even attempt entry.

At an even greater level of exclusivity is the "hidden" conference. All of the sites handling a hidden conference refuse to acknowledge to outsiders that the conference even exists. It will not be listed in a poll of available conferences, and attempts to join it will be refused with a "no such conference" message.

IV. NETWORK CONSIDERATIONS

One of the features of the system design is that the physical capabilities of the stations participating in a conference should be as independent as possible of the network protocol used in carrying out the conference. This means that in communicating both text and graphics information, the network protocol should use a lowest common set of tools, or functions, to communicate the required information.

NETWORK PROTOCOLS

For graphics information, the ACCAT Graphic Order Codes (see Appendix A) provide such a set of tools. The text window, however, needs a similar set of low-order protocols. The guiding consideration behind these protocols is that they should be independent of the geometry of the text display. This means that the text must be treated as a pure character stream, without line breaks. The text is treated as being output at a "current cursor position." Text is output at the cursor position as it is typed, and the cursor moves one position to the "right" for each character typed. Actually, the local host implementation puts line breaks on the display at any positions that are felt to be convenient. There are several types of cursor control, as shown in Appendix B. One of these control characters begins a new "paragraph" of text. Each local conference program is

required to store the current "paragraph" for editing functions specified by the other control characters. This tends to limit the amount of text that must be remembered at each host.

In addition to the protocols that carry information concerning the conference material, there are certain types of messages that are used for control of the conference itself. These include:

- o Conferee entering.
- o Conferee leaving.
- o Allow conferee entry.
- o "Lock" this conference.
- o "Freeze" this conference.
- o "Hide" this conference.
- o List all conferences and conferees.
- o List of conferences and conferees follows.

NETWORK ARBITRATION

The architecture of the network connections used in a conference is such that there is no central arbiter of the conference; rather, each host engaged in a conference knows about all other participants.

When user A wishes to join a conference, he requests this by specifying that he wants a conference with user B. He may do so explicitly, or he may ask to join a conference by name, in which case his local host picks a single conferee from that conference. A's host then initiates a connection with a "server"* on B's host. The server informs A's host that users C and D are also in the conference. It is then the responsibility of A's host to initiate connections with the hosts of C and D. At the completion of this process, everyone knows about everyone else, and the integration of A into the conference is complete. New conferences are started by the initial connection of just two users.

If there is some condition that prevents A from completing the connection, it is the responsibility of A's host to explicitly issue a withdrawal message to those other hosts with whom it has already successfully negotiated. This procedure allows a distributed database of conferees, without requiring an "umpire" host whose untimely demise would kill the conference.

The servers are designed such that the death of a host causes the other hosts to withdraw the person on the dead host from the conference as if that person had willingly withdrawn, but with a message issued on the user's text stream to the effect that that person's host has died.

---

*A "server" is a program running on an ARPANET host computer that is not owned by any specific user on the machine. Its purpose is to initialize transactions requested by users on other ARPANET hosts.

V. SYNCHRONIZATION

Any computer system that operates across a network without the mediation of a central node is subject to problems of synchronization. Numerous projects in recent years have attempted solutions to these problems[2, 3]. However, there are few real concurrency problems in the present application, since the exact order of data received from different conferees is generally unimportant. Where concurrency problems can occur, they are solved in the usual way by breaking them down into a series of strictly bilateral concurrency negotiations between conferees.

One problem that must be addressed is the issue of synchronization among users. This becomes a consideration principally in the case where a new user has joined an ongoing conference and wishes to be informed of the context of the conference. The most desirable form of synchronization might be a replay, under user control, of the entire history of the conference up to that point, but this would put a large storage load on the hosts keeping the history and could take considerable time if the conference had been under way for some time.

Another approach is simply to provide no method for obtaining a context. In this case, a user obtains the context by the same method used in face-to-face conversation: by inference and explicit questioning. This is acceptable in informal contexts, but in a crisis situation, the possibility of misunderstanding--and the possible grave

consequences thereof--make this method unacceptable. (Nevertheless, because of stringent storage limitations, the PLATO conferencing system takes this approach.)

The method chosen here is a compromise: Local implementations are required to keep a context of the current contents of both the text and the graphics screens available for transmission upon inquiry by a conference member. This mechanism allows a user to bring his screen to "current" status by a single request. By "current contents" we mean the current paragraph in each window on the text screen, and everything since the last "clear screen" command on the graphics screen. The actual requests, of course, are handled automatically by the user's host after the user's own initial request for a context. One request is sent to each host in the conference for information on the current text windows handled by that host, and one host (chosen essentially at random) is requested to send the current contents of the graphics screen. Context updates for the text windows merge into the normal updating process; the graphics-screen update process is considered complete when all graphics commands known to that host have been sent. There is a synchronization protocol to prevent graphics commands from being sent by a third host while one host is sending an update to another.

Normally, a request for a context update would automatically be made for a user immediately after he joins a conference that has more than one other person in it (and hence might have been going on for a while).

VI. CONCLUSIONS

The network conferencing system described here does not break dramatic new ground in either conferencing systems or computer graphics display, but rather combines the two in a general fashion to provide a tool that benefits from the synergy generated by the combination. The network environment does not pose special difficulties in the present case, because matters of arbitration are settled in a "round-robin" fashion which, while not the fastest method available, has the advantage of simplicity. The entire project will yield, in addition to a valuable tool, a range of information on an important area of man-machine interaction, without substantial investment in new hardware or technology. Indeed, the real-time aspects of the conferencing system promise to provide a valuable alternative to the highly developed mail systems that are currently available on computer networks.

Appendix A

ACCAT GRAPHIC ORDER CODES

HISTORY

The ACCAT Graphic Order Codes were designed by Rand, in collaboration with the Information Sciences Institute of the University of Southern California, to provide a simple, convenient, low-overhead protocol for transmission of graphical information over the ARPANET. They were specifically aimed at the remote exploitation of bit-plane-based raster-scan color graphics systems over a network. Graphics protocol standards then being designed* were helpful and provided many specific ideas, but they were not suitable for direct implementation. They were found to be too general and difficult to implement and also lacked certain necessary functions, such as the management of separate bit planes. This is not surprising and does not reflect upon the value of such "general" protocols; at the time the ACCAT GOC's were designed, bit-map graphics was sufficiently rare that it was seldom considered in the design of other graphics protocols.

COMMAND FORMAT AND NOTATION

Every GOC is defined as a contiguous sequence of bytes (8 bits). Each GOC sequence begins with a single-byte Command Name, which

*Sproul and Thomas, "A Network Graphics Protocol," ARPANET Protocol Handbook, Network Information Center Publication NIC 7104, SRI International, Menlo Park, California.

identifies the command; GOC Command Name bytes are distinguished from the set of ASCII codes by the presence of '1' in the most significant bit. Command Names will be given here in octal representation.

Within a GOC, the Command Name byte is followed by zero or more bytes of argument information. The size and composition of the argument-byte sequence depends upon the GOC.

GRAPHIC ORDER CODES

- <No-op: 300> - This command is included as a convenience for graphics programmers. It has no arguments. One side effect is to terminate an open string of <Text> args.

- <Plane Mask: 301> <arg 1> <arg 2> - Select which bit-map planes are to have data written into them.

- <On Color: 302> <arg 1> <arg 2> - Select a color for subsequent display objects.

- <Off Color: 303> <arg 1> <arg 2> - Select a color for subsequent erasure operations (background color, etc.).

- <Load VLT: 304> <arg 1> <arg 2> <arg 3> - Select the color table. This command determines which colors may be displayed simultaneously.

- <Set Planes: 305> - The OFF color is written at every pixel in all active planes. This command provides an easy

way to color one or more entire planes in a single operation.

VECTOR CODES

Vectors may be classified as

LONG or SHORT or COMPACT,

DARK or LIT,

ABSOLUTE or RELATIVE.

LONG vectors provide 10 bits of vector magnitude plus sign information; SHORT vectors provide 6 bits of magnitude, plus sign; COMPACT vectors provide 2 bits of magnitude, plus sign.

DARK vectors imply "beam movement" without drawing; LIT vectors are visible.

All vectors have their origin at the "current beam position." The endpoints of ABSOLUTE vectors are specified relative to screen position (0,0) (the lower left-hand corner of the screen) and are therefore always positive. The endpoints of RELATIVE vectors are specified relative to the "current beam position" and may be positive or negative. SHORT and COMPACT vectors are always RELATIVE.

COMPACT, SHORT, and RELATIVE LONG vectors take signed arguments; ABSOLUTE LONG Vectors take unsigned arguments.

A graphics dot is defined as a vector of zero length.

- <COMPACT Vectors: 306> <arg 1> ... <arg n>

- <SHORT Vectors: 307> <arg X1> <arg Y1> ... <arg Xn> <arg Yn>

- <RELATIVE LONG Vectors: 310> <arg E1> <arg X1> <arg Y1> ... <arg En> <arg Xn> <arg Yn> - The E bytes contain extended bits for X and Y.

- <ABSOLUTE LONG Vectors: 311> <arg E1> <arg X1> <arg Y1> ... <arg En> <arg Xn> <arg Yn>

- <Text: 312> <arg 1> ... <arg t> - Each argument is an ASCII character code. The end of the text string is signaled by the appearance of a non-ASCII Command Name. NOTE: This is NOT the code signifying "change to text screen." It is used for including alphanumeric displays in pictures on the graphics screen.

- <Text Font: 313> <arg 1> ... <arg n> <null> - Args 1 through n spell a font name in ASCII characters. The name is terminated with an ASCII NUL character. The named font is activated; other fonts are made inactive.

- <NTDS Symbol: 314> <arg> - The argument number indexes the active NTDS symbol file and causes the selected symbol to be displayed at the "current beam position."

- <NTDS File: 315> <arg 1> ... <arg n> <null> - Args 1 through n spell the name of a file of NTDS symbols. The name is terminated with an ASCII NUL character. The named file is activated; other files of NTDS symbols are inactivated.

- <Cursor control: 316> <color, cursor #, enable> <hi x> <lo x> <hi y> <lo y> - This command will enable the hardware cursor of the graphics screen and display it at the specified x-y location.

- <Plane Display Select: 317> <arg 1> <arg 2> - Select which bit-map memory planes are to be turned on for output. This is separate from "plane select," above; using this code, a plane (or planes) may be output-disabled, a picture may be written into it (or them), and then output may be reenabled, causing the picture to appear as an instantaneous overlay.

- <Image Mode: 320> <direction+count 1> <count 2> <data 1> <data 2> ... - This command allows data to be written directly along or orthogonal to raster lines, to support the direct writing and modification of bit-map images.

Appendix B

TEXT STREAM PROTOCOL

In the following, entities between brackets <> represent control characters in the protocol. Capitalized entities are fixed, predefined numbers, while lower-case entities are variable quantities.

- o <BACK> <count> - Move cursor left "count" spaces, non-destructively.
- o <ERASE> <count> - Move cursor "count" spaces left, destroying "count" characters in doing so.
- o <FORW> <count> - Move cursor forward "count" characters; a zero "count" implies move to end of text stream.
- o <INS> - Change to "insert mode"; insert text at the current cursor position, non-destructively. Characters to the right of the cursor are displaced to make room for the new text.
- o <ENDINS> - Change to "normal mode"; characters typed at the cursor position destroy characters previously typed there.
- o <SUCKL> <count> - Suck characters from the left and delete them, leaving no holes.
- o <SUCKR> <count> - Suck characters from the right and delete them, leaving no holes.

- <GRAPH> - Enter graphics mode (start drawing on the graphics screen).

REFERENCES

1. Woolley, David R., talkomatic program, PLATO Project, University of Illinois at Urbana, 1972.

2. Thomas, R. H., "On the Design of a Resource-Sharing Executive for the Arpanet", AFIPS Proc., Vol. 42, 1973, pp. 155-163.

3. Chang, S. K., Database Skeleton - A Formal Model, Technical Report, Medical Information Systems Laboratory, Department of Information Engineering, University of Illinois at Chicago Circle, 1976.