ZNET - AN INTERPROCESS COMPUTER NETWORK FOR NAIVE TO EXPERT GRAPHICS USERS[1]

James R. Marselle
The Image Producers, Inc.
615 Academy Drive
Northbrook, IL 60062

ABSTRACT

A network has been developed which enables Z80-based microcomputers running ZGRASS, a user-oriented interpretive real-time graphics animation system, to signal and send messages to other ZGRASS microcomputers. Signals have associated event names while messages are short ASCII strings, which could be ZGRASS commands or programs, for example. The ZGRASS units are connected to a minicomputer running the UNIX[2] operating system and are seen as user terminals by UNIX. A new UNIX command processor (shell) was written providing the added network commands while retaining some of the standard shell features. Each terminal's shell process communicates with other shells by routing requests through a network manager process. This is accomplished through the use of locally written interprocess communication system calls. The network commands implemented allow the user to join and leave the network, signal and wait for events, send and receive messages, withdraw previously sent signals and messages, and flush any queued signals and messages. One ZGRASS unit can thus control/interact with one or many other terminal(s) allowing multi-processor graphics, interacting multi-screen displays and so on.

INTRODUCTION

The goal of this research was to design a network which enables graphics computers to communicate in various ways, is easy to learn by naive users, but is also sophisticated and general enough to allow the design of complex, interactive graphical environments. Some possible applications would be (1) synchronization of graphic displays with one terminal controlling others, (2) computerized conferencing with text and graphics, (3) distributed or parallel processing, and (4) computer assisted instruction, in which an instructor on one terminal could monitor progress of students on other terminals.

The Z80-based graphics microcomputers used in this work run ZGRASS, a user-oriented interpretive real-time graphics animation system. ZGRASS is the monitor which interprets user commands as well as the name of the language in which the graphics programs are written (called macros). ZGRASS consists of the Z80 microprocessor with 32K bytes of EPROM and 32K bytes of RAM, custom graphics circuits, floating point hardware, dual UARTs (universal asynchronous receiver/transmitters), a color tv monitor, and an alphanumeric terminal (or CRT). A history and description of ZGRASS can be found in [DeFa 78], [DeFa 80], and [DeFa]. One UART connects the CRT to the ZGRASS unit, providing stand alone graphics on the color monitor. The other is available for communication with another computer.

The second UART is here used to connect ZGRASS to a CALDATA CD135 minicomputer, running the UNIX operating system. For a description of UNIX see [Ritc 74] and [Ritc 78]. ZGRASS is regarded as just another terminal by UNIX and can send commands to UNIX to execute; any terminal output resulting from the execution of a UNIX command is received by ZGRASS. ZGRASS macros and pictures are easily stored on and retrieved from the UNIX file system.

The network (called "ZNET") is implemented entirely as user processes (shells) in UNIX which receive commands from the ZGRASS units to send/receive information to/from other ZGRASS units. A network manager process provides centralized control over the communication. Inter-terminal communication can be effected by user-typed commands or from within a running ZGRASS macro.

PREVIOUS WORK

There are many computerized conferencing systems available, some of which also provide graphics. These systems are keyboard driven, that is, the communication is initiated via user-typed commands. Examples are the Talkomatic and Termtalk programs on PLATO [Wool 72] and the network-oriented, color graphics conferencing system developed at RAND [O'Bri 79], which is loosely based on Talkomatic.

Brinch Hansen describes a multiprogramming system nucleus which provides interprocess communication [Brin 70]. A message queue for each process is maintained and a process may be suspended until a message arrives in its queue. Some restrictions are (1) the sending process must wait for an answer from the receiving process and (2) the communication primitives distinguish between sending and waiting for "messages" and "answers".

Walden refers to data paths as "ports" to and from a process (RECEIVE and SEND ports, respectively), which have an associated unique port number [Wald 77]. Some nice features are the RECEIVE ANY monitor call, which allows a process to receive a message from any other process, and the SLEEP call, which causes process suspension until a specified event occurs. Also useful is the restart location argument to the SEND and RECEIVE calls, which specifies a routine to enter when data transmission is complete.

The HXDP system at Honeywell [Corn 79] allows processes to have a number of windows which can be connected to windows belonging to other processes. All interprocess communication is performed through these windows. The system provides display (send) and view (receive) primitives but a window may only be a display or a view window, not both. Path assignments are static and a view window may be connected to one and only one display window.

Test enumerated some of the shortcomings of interprocess communication schemes in [Test 79]. Three desired features have been provided in ZNET, namely (1) support for more than 2-process dialogs, (2) removal of the restriction on FIFO ordering on messages, and (3) provision of transparent inter-machine communication. Test's system also allows multiple reader/writer processes to be connected together. One drawback is

1. This paper is based on a thesis submitted in partial fulfillment for the degree of Master of Science at the University of Illinois at Chicago Circle, Department of Information Engineering.

2. UNIX is a registered trademark of Bell Laboratories.

CH1756-6/82/0000/0283$00.75 © 1982 IEEE

cesses are required to access complex pointer es to effect the communication.

NETWORK COMMANDS

nunication between a shell and the network is interrupt driven. This is accomplished ɔcally written system calls which allow ɜ to send any of the fifteen available UNIX ɔo other processes. The shell interrupts the ɜhen it has a request. If the request requires ɜer to send or receive information to/from ells, it interrupts these shells, sends its o each and waits. The shells respond by ing the manager which in turn interrupts the requesting shell and provides the requested ɔn.. There are two distinct interrupts which a receive from the manager: a request for ɔn or a response to a request for information. the manager receives a request and a request interrupts from the shells. Descriptions of shell which was written to provide the network and the network manager process are found in

For simplicity we will consider the communities to be computer terminals - each termimmunicate with other terminals. A terminal d to by a unique two-character terminal name, e. To avoid confusion with existing UNIX all ZNET commands are prefixed with 'z'. In ing discussion "current terminal" refers to al issuing a command.

zjoin" and "zleave" commands are used to log- ɜnect and disconnect a terminal from the net- "zwho" command with no arguments is used to ɪe ttynames of all the terminals on the net- having obtained this information, a termi- inter-terminal communication commands which names as arguments. "Zwho am i" will report of the current terminal only.

sig" command is used to signal one or more f an event, which could be a semaphore indisome waited-for action has occurred, or a ɔ another terminal (or terminals) to perform An event has an associated name of six or ɜters. The "zsig" command takes four argu-

names to signal, specified as a string of racter ttynames. Note that a terminal can itself. If the argument is "al", all other ls are signaled.

ɔrity with which the signal is sent, 1 ɜhe highest and 255 the lowest priority. A ɔriority signal will arrive before a lower signal, if sent from different terminals eously.

t name (six or less characters).

ɔnal "-a" argument, which specifies that ɾent terminal is to be notified of the of all other terminals which were waiting ɜ signal when it was sent to them (an ackɲent).

executed, each terminal specified in the is notified of the event. If a terminal the event, the ttyname of the signaling he event name are printed on its terminal GRASS). Otherwise the signal is queued e retrieved later. A signal is sent with y than messages and broadcast messages.

The "zss" command provides information about queued event signals, that is, signals sent via the "zsig" command which have not yet been waited for. If the optional argument '-' is used, a list of signals queued on the current terminal is provided. The list contains (ttyname, event name) pairs indicating the signaling terminal and the event which was signaled. The list is in order of arrival of the signals. This information might be used to decide which (if any) signal will be waited for or if a desired signal has been received at all, analogous to a polling operation. If the argument is of the form "-ttyname" then the list refers to the terminal specified. If no argument is supplied to "zss", a listing of queued events and the ttynames which signaled them is printed for each terminal on the network, providing a snapshot of all signal activity.

The "zwait" command is used to wait for a signal which was sent by the "zsig" command. If a terminal is not waiting for a signal (executing "zwait") when it arrives, the signal is queued and may be waited for later. Otherwise the signaling ttyname and the event name of the waited-for signal are listed on the terminal. The arguments to "zwait" enable the user to specify certain criteria which must be met by a signal in order for it to be waited for. If a queued signal meets these criteria, it is waited for (that is, the ttyname and event name are listed on the terminal). If not, the terminal goes into a suspended state until either a proper signal is received (in which case the ttyname and event name are listed) or the "zwait" command times out (nothing is listed). The arguments to "zwait" are:

1. The ttynames from which we will wait for a signal, specified as a string of two-character ttynames. If this argument is "al", then the ttyname is a don't-care and a signal from any terminal (including the terminal issuing the "zwait" command) will satisfy, provided the event name criterion is met.

2. The event name(s) we are waiting for, separated by blanks. This allows waiting for one or more events with one "zwait" command. If this argument is "al", then the event name is a don't-care and any signal will satisfy, providing the ttyname criterion is met.

3. The optional number of seconds to wait for an acceptable signal if one is not queued. The default is ten seconds.

If more than one queued signal meets both the ttyname and event name criteria, the waited-for signal will be the earliest such signal received, since the signals are queued in the order in which they arrive.

The "zusig" command allows a terminal to remove a signal which was previously sent via the "zsig" command. This feature would be useful in an application in which a terminal sends a signal to other terminals and then removes it after some time interval, thus denying the signal to the terminals which had not yet executed "zwait" for it. As mentioned in the description of the "zwait" command, the signals are queued in order of arrival. The "zusig" command will cause the most recently sent signal meeting the criteria described below to be removed from the queues of the specified terminals, that is, the signal sent via the latest "zsig" command from the current terminal. The arguments to "zusig" are:

1. The ttynames from which to remove the signal, specified as a string of two-character ttynames. If the argument "al" is used, all other terminals will have the signal removed (except those

terminals which have already waited for the signal or never received it).

2. The event name of the signal to be removed. If the argument is "al", the last-sent signal is removed; the event name is a don't-care.

The "zsend" command is used to send messages to one or more terminals on the network. These messages could be ZGRASS commands, arguments to macros, or a character string relevant to the communication context. The "zsend" command takes four arguments:

1. The ttynames to send the message to, specified as a string of two-character ttynames. Note that a terminal can send a message to itself. If the argument is "al", the message is sent to all other terminals.

2. The priority with which the message is sent, 1 being the highest and 255 the lowest priority. A higher priority message will arrive before a lower priority message, if sent from different terminals simultaneously.

3. The optional "-a" argument, which specifies that this terminal is to be notified of the ttynames of all other terminals which were waiting for this message when it was sent to them.

4. The message string.

When "zsend" is executed, each terminal specified is sent the message. If a terminal is waiting for the message, the ttyname of the sending terminal and the message are printed on the terminal. Otherwise the message is queued to possibly be received later. A message is sent with lower priority than signals and broadcast messages. If the "-a" argument is specified, the ttyname of each terminal which was waiting for the message when it arrived is reported. Otherwise "zsend" just returns.

The "zms" command provides information about queued messages, that is, messages sent via the "zsend" command which have not yet been received. The syntax and operation of "zms" is similar to "zss".

The "zrecv" command is used to receive a message which was sent by the "zsend" command. If a terminal is not waiting for a message (executing "zrecv") when it arrives, the message is queued and may be received later. Otherwise the sending ttyname and the message are listed on the terminal. The syntax and operation of "zrecv" is similar to "zwait". There are two arguments: the ttynames from which to receive the message and the wait time.

The "zusend" command allows a terminal to remove a message which was previously sent via the "zsend" command. The syntax and operation of "zusend" is similar to "zusig". The sole argument is the list of ttynames from which to remove the most recently sent message.

The "zbroad" command is used to send broadcast messages to one or more terminals. The arguments are identical to those of the "zsend" command except that there is no priority argument. The difference between "zbroad" and "zsend" is that broadcast messages are never queued at the receiving end; instead an attempt is made to print them on the sent-to terminal(s) by initiating a handshaking protocol between UNIX and ZGRASS. If the ZGRASS terminal has enabled broadcast messages, it will be printed, otherwise it will be ignored. A broadcast message is sent with lower priority than signals but higher priority than messages. If the "-a" argument is specified, the ttyname of each terminal which accepted the broadcast message is printed. Otherwise "zbroad" just returns.

The "zflush" command is used to empty a terminal's signal and/or message queues. "Zflush" with no arguments flushes both queues. If the argument is "-s" the signal queue is emptied, if "-m" the message queue is emptied.

CONCLUSION

The network which was designed and implemented in this research provides a powerful tool for inter-terminal communication in a computer graphics environment. The extent to which this tool can be utilized has not yet been examined and remains a subject for further scientific and artistic research. It can be concluded that a dedicated message switching module could successfully replace UNIX, providing a more nearly real-time network.

REFERENCES

[Brin 70] Brinch Hansen, P. "The Nucleus of a Multiprogramming System", Comm. ACM, Vol. 13, No. 4, April 1970, pp. 238-241, 250.

[Corn 79] Cornhill, D.T. and W.E. Boebert, "Implementation of the HXDP Executive", Proc. Spring Compcon '79, March 1979, pp. 219-221.

[DeFa 78] DeFanti, T., J. Fenton, and N. Donato, "BASIC Zgrass - A Sophisticated Graphics Language for the Bally Home Computer", Computer Graphics, Vol. 12, No. 3 (August 1978), pp. 33-37.

[DeFa 80] DeFanti, Thomas, "Language Control Structures for Easy Electronic Visualization", BYTE, Vol. 5, No. 11, Nov. 1980, pp. 90-106.

[DeFa] DeFanti, Thomas, ZGRASS Documentation, unpublished.

[Mars 81] Marselle, James, "ZNET-An Interprocess Computer Network for Naive to Expert Graphics Users", Master's Thesis, Dept. of Information Engineering, Univ. of Illinois at Chicago Circle, April, 1981.

[O'Bri 79] O'Brien, Michael T., "A Network Graphical Conferencing System", Rand Note N-1250-ARPA, August 1979.

[Ritc 74] Ritchie, Dennis M. and Ken Thompson, "The UNIX Time-Sharing System", Comm. ACM, Vol. 17, No. 7, July 1974, pp. 365-375.

[Ritc 78] Ritchie, D.M. and K. Thompson, "The UNIX Time-Sharing System", Bell Sys. Tech. J., Vol. 57, No. 6, July-August 1978.

[Test 79] Test, Jack A., "An Interprocess Communication Scheme for the Support of Cooperating Process Networks", Proc. 1st Intl. Conf. on Dist. Computing Systems, Oct. 1-5, 1979, pp. 405-411.

[Wald 72] Walden, David C., "A System for Interprocess Communication in a Resource Sharing Computer Network", Comm. ACM, Vol. 15, No. 4, April 1972, pp. 221-230.

[Wool 72] Wooley, David R., Talkomatic Program, PLATO Project, Univ. of Illinois at Urbana, 1972.