/NETPL, 09/11/68 1719:29 DGC ; .NBL=2; .PLO=1; .DPR=0;

Abstract

This paper represents JFR's and JDH's current views on how the NET problems should be attacked. The problems covered are those local to the 940 Monitor and NLS.

These two areas, NLS and the Monitor, bring out most of the low-level NET communication problems, and we hope that a general NET solution can evolve from these ideas.

This paper is currently for AHI communication, but we would like to pass it out at the next NET meeting in Utah, so we need responses in time to resolve difficulties.

General Notions

Aim of Paper

This paper gives a fairly detailed explanation of an IMP-940 Monitor implementation scheme that we feel would make the whole mess work.

It attacks the single problem area of interactive display terminals, and does not get into complex interactive programming running. Interaction between user programs can easily be accomodated within our proposal.

This is a two-pass paper. One may not expect all new concepts to be introduced before they are needed.

Comments on Other Papers

TRAC

S. Carr's memo on number of translators

Network Software Interfaces

GULP

GS -- Graphics System

Elmer's memos

Letter dated July 5 1968

Letter dated May 28 1968

Letter dated August 12 1968

Memo dated 12 August 1968

Memo dated 15 August 1968

Memo dated 26 August 1968

ARPA Specifications for the IMPs

Results from Meetings

Familiarity with each others' consoles

Not much else, a lot of hardware talk

NET vs. Data Communications

Make full use of host computer

Real-time on-line consoles need it

Large number of different machines

Using Your Console

Expect no console degeneracy

Even expect extra features at times

Division of Labor

Host shoud not be aware of actual console at user site execpt in very general ways,

User's own computer can tackle most of the formatting and translation problem.

Extra Thoughts on General Notions

AHI MONITOR - IMP Communication

INTRODUCTION

AHI NET Controler

The AHI 940 has a running link to the Project Genie 940 at Berkeley. Viewed from the AHI machine, the link is a special, home-built controller connected to two 2400 bit/s duplex data lines. The duplex feature is currently ignored. During the FJCC these lines will be used for the transmisson of all the information necessary to run an AHI console, except the video. This controller will be the one used for the IMP control as well as the GENIE link. It operates much like the other special-purpose hardware devices.

In the 940 resident, writable, core memory there is a user reference cell (URC) reserved for the NET controller. This cell contains the absolute core address, using the full core memory space and no fancy relabeling, of the NET command table.

The command table contains a contiguous list of double word entries. Each entry does the following:

Points to another absolute core address

Specifies a transfer size in words.

There is a single 940 instruction (EOM) associated with the NET controller. When the instruction is executed the controller goes to the URC, gets its address, and begins stepping through the list of transfers in the command table. The single interrupt which the NET controller can issue is used to signal END OF MESSAGE when the orders in the command

table have been completed.

Kinds of Control Needed

From many programming points of view, the IMP may be considered a peripheral input/output device. It sends and receives control information and blocks of data, much like a tape drive. The expected transmission of a huge number of single-character messages, however, requires special consideration. If an excessive "call and response" syndrome develops between the IMP and the 940, the time used for interrupt interpretation, message preparation and acknowledgment, and other kinds of overhead will severely restrict the service of the TSS.

There are only three kinds of information flow between the IMP and and 940 Monitor.

The first is control information for the IMP itself. This may be timing informaton for the block "race problem," dynamic status of the 940 or the IMP, or table indices. This should not be a large volume of information and should not require particularly high transfer rates.

The user-host and host-user control information and single-character messages, although they may be only a few bits, possibly require the bulk of service from the 940 Monitor. They should, therefore, be sent and received with a minimum of interrupts and translation.

Large, bulk, bit transfers such as files and data bases make up the third kind of data transfer. The 940 Monitor is unconcerned with the bits themselves. User programs will receive the blocks directly from the IMP, and must do any

translation or sorting they see fit.

NET Users as Ordinary Users

At some of the meetings and in the memos Elmer has written there appears to be apprehension about the number and kind of users that will be serviced by a host at one time.

If each individual person at a remote console is treated as a standard user within the 940 TSS, nothing special is needed to service as many different users as the system will normally bear.

Making each NET user a regular 940 user, with an internal job number and fork structure, has all the added benefits of interjob protection, file storage, and minimization of programming effort.

At first, user names may be site names to avoid overflowing our job tables.

Using Teletype Lines

Interrupts and Character Transfer

The following hardware configuration for connections between the 940 and the IMP fits well into the above considerations.

The IMP would have eight Teletype lines connected directly to our general Teletype patch-panel. These lines will operate exactly like standard Teletypes as far as the 940 Monitor is concerned. The figure eight is determined by the expected Monitor size and the number of total Teletypes that the system will be assembled to be plugged into.

The Imp would go through the already existing 940 NET controller for its bulk transfers. The control and

synchronization of the transfers is discussed below. The bits transferred are ultimately controlled by the URC and command table.

Teletype Line Usage

The eight Teletype lines would be divided into two sets: one of seven devoted to user use, and the last line devoted exclusively to IMP-940 Monitor communication.

To the 940 Monitor, the seven user lines would appear exactly like standard Teletype lines. Characters coming over the lines would be packed in the Teletype buffers in the normal way. They would then be passed off to the Executive or the user program. This character stream could contain information about bulk transfers for a user program, but this is of no concern to the Monitor.

The last Teletype line is used exclusively by the Monitor for IMP control. It would have a fixed number, say 16. No user could ever log in on Teletype 16.

If a line were not devoted to control, characters would have to be taken out of the set used for user messages. This approach opens a Pandora's box which should stay closed. With the entire range of characters open to user host, the standard set will never be changed, programs will not be affected by changes in the Monitor-IMP relationship, and everyone will be happy.

This line is currently conceived of as serving two purposes.

Suppose that there are a number of users currently running under the TSS from the IMP. They may all be in

the process of receiving bulk transfers.

When a user program is ready to accept a packet of bits it calls on the monitor with a BRS. The monitor must first prepare the URC and command table, after making sure they are not in use.

It must then notify the IMP that it can send the message, but it must be sure that it gets the message for the right user. This information is sent over the reserved Teletype line.

When the transfer is complete the IMP, through the controller, interrupts the 940. The monitor notes the fact, and the user may be fired up again when he comes up in the queue.

All the information about the IMP, the NET, and the 940 is also sent on this line. This includes information about the loads, up status, and anything else not related to, requested by, or sent by users. Everything on a user line goes to a user, and nothing is gobbled up in a secret way by the Monitor, IMP, or anything else.

Initial Sign-On

When someone wants to log onto the 940 system, a request of some kind will eventually reach the local IMP. This request would probably be initiated on the user's machine and sent to the remote IMP via the control line.

When the request finally reaches the local IMP, it (the local IMP) must seek a Teletype line to connect the remote user with the local host. Once the line is found, it

remains constant and may not be changed or used by another user until LOGOUT.

After the line has been found, the IMP may send acknowledgement back to the user site. It must also send a rubout to the 940 on the newly established line.

Log-In

When the 940 gets a rubout from a Teletype line, whatever it is attached to it goes into its normal log-in procedure.

It first sends back a short line of text with the current system name, date and time, a few carriage returns and line feeds, and the word ENTER.

The user must then enter a user name, and he is off and running on the EXEC level in the TSS.

Rubout Conventions

The 940 Monitor has some very special conventions about rubout characters that come in on Teletype lines.

A user program that is allowed executivity can request rubout control if it wishes, or leave the control up to the monitor.

If it requests control, then rubouts are sent to the program just as any normal character would be.

If control is not requested, and a rubout comes in, the Monitor wipes out the lowest-level fork in the structure connected to the Teletype. This provides a way out of loops.

In any event, if two rubouts come through in less than half a second, the Monitor wipes out the lowest fork, no matter what the user requested. This is the ultimate breakout, and a necessary

feature in the TSS.

Somehow, this feature must be handled by the IMPs in the NET. Two different solutions are immediately obvious.

The IMP could keep track of the time between characters and not lose the timings during the transfer. This does not appear to be a good approach.

Alternatively, the IMPs could send double rubouts as special control information between themselves and decode this information to local conventions at the host sites. In our case it would be two fast rubouts; somewhere else it might be a special interrupt.

AHI User Program - NET Communication

Definition of a User Program

For our purposes a user program is a piece of code that runs in user memory, protected from others, from itself, and from executing I/O instructions.

User programs make their requests for I/O, as well as other system operations, through a special system call known as a BRS.

The Monitor in the 940 will do a minimum amount of work to run the NET. All it will do is buffer enough characters to bridge the gap between swaps and provide control for bulk transfers. Everything else is done by user programs.

NLS is a user program. Special NET work for NLS will be done by NLS.

Utility routines, such as file transfers, will also operate on the user level when they are written. They will be swapped like everything else, and should not put any more burden on the Monitor that other kinds of Executive request.

Teletype Input

A user program running from the IMP would not normally know that it originated on a device other than a Teletype. The commands to read and write characters will operate in exactly the same way.

BRS's Needed

The BRS's necessary to operate the IMP from a user program are similar to those necessary to read and write files. In fact a user program can think of the IMP as two files.

A BRS would be needed to open the IMP for reading, and another for writing. These would work like the file-opening BRS'S and return

an associated file number that would be used in all future data transfers.

In addition, there would be a standard block read/write BRS. One would put the file number, the starting location, and the number of words to transfer in the three registers and issue the BRS. The file number determines the read or write mode.

At this point the monitor would go through all the usual routines for block transfers. If the starting location at and the number of words match page boundaries, the page could be frozen and the transfer done directly into user memory. If things are at odd locations a buffer could be used, as it is for drum transfers.

The data would not be sorted by the IMP, and so each user program must do its own sorting. A "read" to the IMP means to read the next N words as they were received by the IMP.

It is expected that some control information about bulk transfers will be sent over the Teletype lines. For example, the Teletype lines would send a character stream that the user program would interpret as a signal that a block of a certain size was being sent. The user program would then issue the read to the Monitor. The Monitor would synchronize with the IMP and get the message. The user program would then be started up again. It is likely that for files, this first bulk message will be conrol information about a coming series of even larger bulk messages.

NET Standard Translators

Universal Hardware Language

To minimize the numer of translators needed to map any facility's user codes to any other facility, there is a universal hardware language.

This language is simply a large, syntactically simple language in which statements can be made about any hardware device in the NET. The language may never be used in transmission, however. Its primary use is as a set of specifications for the coding of the NSTs at each user site.

Introduction to the NST

Suppose that a user at a remote site, say UCSB, is entered in the AHI system and wants to run NLS.

The first step is to enter NLS in the normal way. At that time the UCSB system will request a symbolic program from NLS.

This program is written in an algebraic, parsing-oriented hybrid language. It is called the NLS Encode Translation Program.

The program accepts input in the NET hardware language and translates it to a form usable by NLS.

It may pack characters in a buffer, and also do some local feedback.

When the program is received at UCSB it is compiled and loaded to be run in conjunction with a standard library.

All input from the UCSB console first goes to the NST. It is processed, parsed, blocked, translated, etc. When the NST receives a character appropriate to its state it may finally

initiate transfers to the 940. The bits transferred are in a form acceptable to the 940, and maybe in a standard form so that the NLS need not differentiate between UCSB and other NET users. Information sent to UCSB from NLS must go through a similar process.

NLS will request a symbolic program from UCSB. This program maps standard NET hardware language into packs specifically made up for the UCSB machine. It is known as the NLS Decode Translation Program.

This program is compiled and run along with NLS. It operates out of user memory just as NLS does, and does not affect the Monitor.

Function of NST

Advantages of Dual Translation

After each node has implemented the library part of the NST, it need only write one program for each subsystem, namely the symbolic file it sends to each user that maps the NET hardware language into its own special language. Each user must also write a decode translation program for the new subsystem. The decode program should normally be rather small compared to the rest of the worries of using a remote system.

This is the minimum programming that can be expected if each console is used to its fullest extent.

Since the NST which runs the encode translation is coded at the user site, it can take advantage of hardware at its consoles to the fullest extent. It can also add or remove hardware features without requiring new or different

translation tables from the host.

Local users are also kept up to date on any changes in the system offered at the host site. As new features are added, the host programmers change the symbolic encode program. When this new program is compiled and used at the user site, the new features are automatically included.

The advantages of having the encode translation programs transferred symbolically should be obvious. Each site can translate any way it sees fit. Thus machine code for each site can be produced to fit that site; faster run times and greater code density will be the result.

It is expected that when there is matching hardware, the symbolic programs will take this into account and avoid any unnecessary computing.

(nst) The NST library is the set of programs necessary to mesh efficiently with the code compiled at the user sites from the encode translation it receives.

Universal Hunk of Bits

The Universal Station

AHI NLS - User Console Communication

Block Diagram

The right side of the picture represents functions done at the user's main computer; the left side represents those done at the host computer.

Each label in the picture corresponds to a statement with the same name.

There are four trails associated with this picture. The first links (in a forward direction) the labels which are concerned only with network information. The second links the total information flow (again in a forward direction). The last two are equivalent to the first two but in a backward direction. They may be set with pointers t1 through t4 respectively. [">tif"] OR [">nif"]; ["<tif"] OR ["<nif"];

User-to-Host Transmission

(keyboard) is the set of input devices at the user's console. Input bits from stations, after drifting through levels of monitor and interrupt handlers, eventually come to the encode translator. [>nif(encode)]

(encode) translator maps the semi-raw input bits into an input stream in a form suited to the host subsystem which will process the input. [>nif(hrt)<nif(keyboard)]

The Encode program was supplied by the host subsystem when the subsystem was first requested. It is sent to the user machine in symbolic form and is compiled at the user machine into code

particularly suited to that machine.

It may pack to break characters, map multiple characters to single characters and vice versa, do character translation, and give immediate feedback to the user.

(ldm) Immediate feedback from the encode translator first goes to local display management, where it is mapped from the NET standard to the local display hardware.

A wide range of echo output may come from the encode translator. Simple character echoes would be a minimum, while command and machine-state feedback will be common.

It is reasonable to expect control and feedback functions not even done at the host user stations to be done in local display control. For example, people with high-speed displays may want to selectively clear curves on a Culler display, a function which is impossible on a storage tube.

Output from the encode translator for the host goes to the invisible IMP, is broken into appropriate sizes and labeled by the encode translator, and then goes to the NET-to-host translator.

Output from the user may be more than on-line input. It may be larger items such as computer-generated data, or files generated and used exclusively at the host site but stored at the user site.

Information of this kind may avoid translation, if it is already in host format, or it may undergo yet another kind of translation if it is a block of data.

(hrt) It finally gets to the host, and must then go through the host reception translator. This maps and reorders the standard

transmission-style packets of bits sent by various Encode programs into codes acceptable by the host subsystem in use. This part may not even exist if things work out.[>tif(net mode)<nif(encode)]

Host-to-User Transmission

(decode) Output from the host initially goes through decode, a translation map similar to, and perhaps more complicated than, the encode map.[>nif(urt)>tif(imp ctrl)<tif(net mode)]

This map at least formats display output into a simplified logical-entity output stream, of which meaningful pieces may be dealt with in various ways at the user site.

The Decode program was sent to the host machine at the same time that the Encode program was sent to the user machine. The program is initially in symbolic form and is compiled for efficient running at the host machine.

Lines of characters should be logically identified so that different line widths can be handled at the user site.

Some form of logical line identification must also ne made. For example, if a straight line is to be drawn across the display this fact should be transmitted, rather than a series of 500 short vectors.

As things firm up, more and more complicated structural display information (in the manner of LEAP) should be sent and accomodated at user sites so that the responsibility for real-time display manipulation may shift closer to the user.

(imp ctrl) The host may also want to send control information to IMPs. Formatting of this information is done by the host

decoder. [>tif(urt) <tif(decode)]

The other control information supplied by the host decoder is message break up and identification so that proper assembly and sorting can be done at the user site.

From the host decoder, information goes to the invisible IMP, and directly to the NET-to-user translator. The only operation done on the messages is that they may be shuffled.

(urt) The user reception translator accepts messages from the user-site IMP and fixes them up for user-site display. [>nif(d ctrl)>tif(prgm ctrl)<tif(imp ctrl)<nif(decode)]

The minimal action is a reordering of the message pieces.

(d ctrl) For display output, however, more needs be done. The NET logical display information must be put in the format of the user site. Dispay control does this job. Since it coordinates between (encode) and (decode) it is able to offer features of display management local to the user site.[>nif(display)<nif(urt)]

(prgm ctrl) Another action may be the selective translation and routing of information to particular user-site subsystems. [>tif(d ctrl)<tif(urt)]

For example, blocks of floating-point information may be converted to user-style words and sent, in block form, to a subsystem for processing or storage.

The styles and translation of this information may well be a compact binary format suitable for quick translation, rather than a print-image-oriented format.

(display) is the output to the user. [<nif(d ctrl)]

User-to-Host Indirect Transmission

(net mode) This is the mode where a remote user can link to a node indirectly through another node.[>tif(decode)<tif(hrt)]

SRI Usage of Other Stations

UCSB

UCLA

Utah

Implementation Procedure

NST

Universal Hardware Language

The first step is the specification of the universal hardware language. This should not be too difficult. The language can be syntactically straightforward; probably a left-liner grammar will suffice. The only worry is providing for expansion in the language so that any new device may be easily accomodated.

Work should begin as soon as possible. The man-hours required should not be excessive, say a week from each site. However, the elapsed time could easily grow because of the communication problems.

Specifications and Compiler for Encode/Decode Language

The only way to design the Encode/Decode Language is to try and write an Encode program. Maybe DIA and JFR shoul try this for NLS. After an initial guess is made at the syntax, it could be written up other sites could undertake a similar task. After everyone has tried once, we could all get together and work out the details.

NST Library

The compiler and the library must be worked out at the same time, but after the syntax has settled down a bit.

TSS

As specified above, changes to TSS are not very great. It may be wise to wait until more is known about the IMPs before much serious work is done on TSS.

NLS

Fix Up Display

Encode Package

High-Speed Tubes

Storage Tubes