JGN 15-MAY-75 16:19 32520

i got your message about new elf kernel programmers guide.

see title. i'll be glad to get it. 1

i got your message about new elf kernel programmers guide.

(J32520) 15-MAY-75 16:19;;;; Title: Author(s): J. Gregory Noel/JGN; Distribution: /FGB( [ ACTION ] ) ; Sub-Collections: NIC; Clerk: JGN;

The author recently finished a study on consumer demand for pay television. The study was conducted at SRI for the (U.S.) Office of Telecommunications Policy. This paper discusses the outlook for cable in the big cities, if the study's results are correct. Output processor directives are included.

# Pay Television and Cable Television:
# Back to the Wild Blue Yonder? 1

Ra3y Panko
Communication Analyst
Stanford Research Institute
333 Ravenswood Avenue
U.S.A. Menlo Park, California 94025
(415) 326-6200 ext.4213 1a

Prepared for NTC-75 1b

SUMMARY

2

Pay television is already a commercial success, but is it the messiah that will lead cable television into the big city markets? We find that the outlook for big-city cable is excellent in the long run, uncertain but promising in the next five years or so. Since pay television does not require two-way transmission, the outlook for home shopping, voting, etc., is not bright in the short run. 2a

Ra3y Panko

# Pay Television and Cable Television:
# Back to the Wild Blue Yonder?

3

During the 1960's, most analysts were bullish about cable television. Inexpensive, two-way broadband communications, they argued, would bring a host of new services. Such terms as the "new communication revolution" were used with great confidence. Unfortunately, cable television's future was dependent upon the emergence of "blue sky" services, such as pay television and in-home shopping. Without them, cable television would remain the peripheral service it was in the 1960's, offering only clearer reception and a few extra signals in rural areas and the suburbs of big cities. 3a

Cable system owners managed to convince investors that the prevalent monthly revenues in the 1960's - roughly $5.00 per subscriber household - would eventually rise to $15.00 or $20.00, and that most American househeholds would become cable subscribers. Funded well, cable television grew rapidly in the 1960's. Then the bottom fell out of the market, when the first two-way systems found little demand for blue-sky services. Investors virtually abandoned the industry, and many multiple system cable operators came close to bankruptcy. 3b

Yet while caution was making itself felt throughout the cable industry, the first of the blue sky services was quietly and cautiously entering the consumer marketplace. This was pay television. In 1968 the Federal Communications Commission authorized broadcast pay television (court tests withheld implementation of the authorization until 1970). In 1970, the FCC authorized pay cablecasting, or more precisely regulated it. To date, no broadcast pay stations have offered public service, but prospects are good and at least one company may soon enter the market. Pay cable, on the other hand, did not remain in limbo. In 1972, the first major pay television operator began service. This was Home Box Office, which began operation in Wilkes-Barre, Pennsyslvania in November 1972. By the end of 1973, national subscribership reached 50,000 to 100,000 households, but the evidence for pay cablecasting's ultimate viability was still weak; during 1974, growth was only modest for a new industry. Yet during 1974 Optical Systems managed to demonstrate that intensive marketing would bring profitable operation. Other pay companies, following Optical's lead, began intense marketing and also obtained good results. 3c

With intensive marketing, a pay cable operator could charge $8.00 per month (above and beyond the basic cable fee) and attract about 30 percent of the subscribers on a cable system having 50 percent penetration or higher. Where the penetration of the host cable system was lower, a higher percentage of cable subscribers would subscribe. This could be done with a relatively simple program offering, about 10 movies each month (roughly Hollywod's monthly output of major films). Pay operators anticipated even higher revenues once sports, other entertainment programs and nonentertainment programs could be added. 3d

In the spring of 1975, Home Box Office took a bold step and announced the formation of a national satellite network for pay television. Once again, investors became enamoured by pay television and cable as well. The fate of a 1974 study by Kenneth Penchos, then at the Stanford Research Institute, illustrates how thoroughly the situation had changed. At the 1974 convention of the National Cable Television Association, the study was bitterly attacked by industry spokesmen as absurdly optimistic. Jack Valenti, president of the Motion Picture Association of America, said that his children could do a better job. In the spring of 1975, shortly ater the HBO announcement, the president of a major MSO complained privately that Penchos' projections were too low to show investors. 3e

What will the future really bring? of course nobody can answer this question definitively today, but at least the rough outlines of the industry's future can be drawn. For example, the

Penchos study projected national pay television (broadcast and cable) subscribership at 1.5 million households in 1976 and 25 million households in 1985. It projected annual revenues of $200 million in 1976 and $4 billion in 1985. Another SRI study, conducted in another department under a project team led by the author of this paper (Ra3y Panko), projected somewhat lower demand for the long term. For the end of 1977, broadcast and cable pay television subscribership was projected at 2.1 million households, with annual revenues of $249 million. But the 1985 projections in Panko (1975) were 16.2 billion households and $1.9 million in annual revenues. In comparison, the theatrical movie industry now has annual revenues of about $1.5 million, and advertiser-supported television has annual revenues of about $4 billion. 3f

The key difference between the long-run projections in the Panko and Penchos studies lay in their analyses of consumer demand for a broader program mix. The Penchos study concluded that demand growth would come from special-audience programming. The Panko analysis indicated that movies, today's attractions, tap about 60 percent of the long-run demand of a typical subscriber household. This conclusion was drawn from analysis of various "analogous attractions", including consumer spending at live events, television viewing, projective budget consumer surveys and early pay television experiments. In all of these data sources, theatrical films were found to contribute 30 to 45 percent of the revenues of a broad offering. Moreover, in the future, movies will have to compete with other pay television programming, so that imperfectly elastic consumer demand will limit revenue growth. 3g

Although a typical household's demand, according to the panko anaysis, will increase by about 70 percent between today and 1985, revenues from a typical commuity will increase much more rapidly. As the program mix expands, more people will subscribe to pay cable. For a reasonably typical community, subscribership will double between today and 1985. However new subscribers will, again according to the Panko analysis, have less pay television demand than early adopters. While subscribership will double, revenues from a typical pay commuity served by pay cable television will increase about 160 percent. In the long run, a community that now has a host cable system penetration of 50 percent can expect 45 percent of its population to take pay television, and the average subscriber will pay about $10.50 per month. 3h

The Panko study found that consumer demand varies with the penetration of the host cable system. It was estimated that today about 23 percent of initial cable subscribers and 7 percent of initial noncable households would take pay cable if it is offered in a community. If initial penetration of the host cable system were 50 percent, then about 27 percent of final cable subscribers would take pay cable today, and cable penetration would rise to 53.5 percent, a 7 percent increase. Where the host cable system has lower penetration, pay cable will have a larger impact. For example, if the initial penetration of cable is 20 percent, the Panko study would predict a rise to 25.6 percent, a 28 percent increase in subscribership; about 40 percent of these final cable subscribers would be pay subscribers. These projections, which were derived during 1974, have been verified in the Theta Cable system in Los Angeles. Penetration there jumped from 18 percent to 25 percent after the introduction of pay cable (Broadcasting, 1975). 31

What will pay cable do for the cable operator's profits? Pay cable will bring revenues from two sources. One is the increase in subscribership to cable itself. Because the marginal cost of adding an additional subscriber to a trunk line is small, cable penetration increases translate almost directly into increased profit. If the basic cable service price is $7 per month, a cable operator will see an increase in revenues of $0.50 percent per initial subscriber if his initial cable penetration is 50 percent, or an increase of $1.96 per month per initial subscriber if his initial cable penetration is 20 percent. The second source of profit is the cable operator's share of pay cable revenues. Suppose pay service is sold for $8 per month and the cable operator receives 25 percent, after expenses. Then if initial cable penetration is 50 percent, revenue sharing will bring $0.64 per initial cable subscriber, and if penetration is 20 percent, revenue sharing will bring $0.80. 3j

Low-penetration cable systems will benefit most from pay cable, but will pay cable be enough to bring cable into the big city markets where carriage of off-the-air signals will bring only 20 percent or so penetration? For the distant future the answer is almost certainly yes. The Panko study would predict a final cable penetration of 44 percent in 1985: this is certainly enough for profitability, especially if revenue sharing is added. But we have noted that penetration would only be about 26 percent today. And even if the Panko estimate of demand among nonsubscribers is low by a factor of two, cable penetration would only be 32 percent, probably not high enough for profitability in most big city markets. In the intermediate term, the situation is also uncertain. At the end of 1979, cable pentetration would be about 37 percent under the Panko model - perhaps enough for any market, perhaps only enough for some markets. 3k

Another factor that may stall cable's entry into the big cities, is broadcast pay television. Current FCC rules allow broadcast pay television, usually called subscription television or STV, in approximately 38 of the largest television markets. STV operators can bring service directly to consumers, relieving the consumer of the basic cable charge. How pay cable will fare against STV competition is unclear. First, STV may not be viable. Although the Panko study predicts that an STV station today would attract about 10 percent of the residents in a community, no STV stations have been built since the inconclusive Hartford STV experiment in the 1960's, so consumer response is conjectural. Second, although STV enjoys the advantage of bypassing cable, this is also a disadvantage. The FCC apparently will allow only one STV station per market. As the program mix broadens, STV will have a difficult time competing against multichannel pay cable. All in all, the outcome of competition depends on aspects of consumer demand that are not measurable today. 31

Another question is whether pay cablecasting will open the door for other exotic services. It had been speculated in the 1960's that pay cable would pay for the introduction of two-way terminals into the home. These terminals could then be used for voting, in-home shopping, security, and so forth. But pay cable is progressing nicely without two-way communication, and when two-way billing does come, it seems likely to use the telephone system. But the situation is not completely pessimistic. By 1980, the Panko study projects that 54 percent of all cable systems will offer pay television. This may be a large enough market for the creation of full two-way hardware, whether using cable or the telephone system. 3m

On the whole, there are many things to be optimistic about today. Cable seems likely to ultimately enter the big city markets, but not as rapidly as some analysts had hoped. When entry will happen, however, and what this will mean for services other than pay television, are unclear now. For the near future, we see cable and pay cable growing rapidly in the suburbs and in cities with comparatively poor television service. We see a gradual penetration of cable into the top markets, entering first where there are natural terrain problems and gradually spreading until, by 1985, the nation is wired and almost all Americans have the choice of whether or not to take cable and pay television. 3n

REFERENCES

4

Anonymous "Future Seen Now as HBO, UA-Columbia Pact First Affiliation," Broadcasting, April 21 1975, p.17. 4a

Panko, R.R., G.C. Edwards, K. Penchos, and S.P. Russell, Analysis of Consumer Demand for Pay Television, Stanford Research Institute (May 1975). prepared for the Office of Telecommunications Policy. 4b

Penchos, K., The Outlook for Cable Television, Volume 5: Pay Television, Stanford Research Institute (May 1974). This is a multi-cllient study. 4c

(J32521) 15-MAY-75 16:26;;;; Title: Author(s): Raymond R. Panko/RA3Y; Distribution: /RWH( [ INFO-ONLY ] ) BELL-CANADA( [ INFO-ONLY ] ) ; Sub-Collections: SRI-ARC BELL-CANADA; Clerk: RA3Y; Origin: < PANKO, PAYTV.NLS;7, >, 14-MAY-75 16:00 RA3Y ;;;;

####;

People, for your information: the new password for logging into the ELF directory at SRI-AI is ELFIST. Once again, please acknowledge receipt of this message. Thanks. Frank 1

(J32522) 15-MAY-75 17:01;;;; Title: Author(s): Frank G. Brignoli/FGB; Distribution: /NAVIMP( [ ACTION ] ) ; Sub-Collections: NIC NAVIMP; Clerk: FGB;

Idents

Paul, Dave's new ident is jdb2 in the NAVIMP directory. Anna Watson is AW2 in the NAVINFO directory. The rest of ident stuff (phone numbers,etc.) is fixed up. Regards. Frank 1

Idents

(J32523) 15-MAY-75 17:05;;;; Title: Author(s): Frank G. Brignoli/FGB; Distribution: /PCB( [ ACTION ] ) JDB2( [ INFO-ONLY ] ) AW2( [ INFO-ONLY ] ) ; Sub-Collections: NIC; Clerk: FGB;

FGB 15-MAY-75 17:07 32524

Ident for McGilvary (JCM3)

Jim, JCM3 is the new ilent for McGilvary in NAVIMP directory.
Regards, Frank 1

Ident for McGilvary (JCM3)

(J32524) 15-MAY-75 17:07;;;; Title: Author(s): Frank G. Brignoli/FGB; Distribution: /JPS( [ ACTION ] ) ; Sub-Collections: NIC; Clerk: FGB;

DLS 16-MAY-75 06:29 32527

NETWORK PERFORMANCE STUDY

This is a message received from Walden at BBN, which summarizes the findings and fixes applied to the TIPs as a result of the March and April 'flurry' of activity generated by SRI's use of the net. Note the possibility of dropping 2741 code to gain more buffer space. This might be necessary when we get a lot of lineprocessors.

4-MAY-75 1659-EDT WALDEN at BBN-TENEX: NETWORK PERFORMANCE STUDY: Status Report on the TIP Distribution: NET PERFORMANCE TECHNICAL GROUP [BBN]<MCKENZIE>NPTG.TXT: Received at: 5-MAY-75 08:49:49-PDT 1

1. TIP Access Control and User Account* 1a

In our Quarterly Technical Report No. 6 we reported our design of a mechanism to provide access control and user accounting for Terminal IMPs. The mechanism we designed was based on the use of the TIPSER/RSEXEC and attempted to balance the desire for giving the user the feel of a large system against the extremely limited TIP core memory available for program and device buffers. In our Quarterly Technical Report No. 8 we reported that during the last month of 1974 we were able to begin the release of the TIP software system (TIP software version 327) which implemented the access control and user accounting mechanisms. 1b

During the early part of the first quarter we completed the release of TIP version 327 to all TIP sites. The access control mechanism was enabled to the extent that if a user's name was contained in the authentication data base then the correct password was required; however, if for some reason a user's name was not contained in the authentication data base then any password was accepted and the user was permitted to continue his TIP session. This mode of operation was intended to permit tardy organizations to submit user authentication data without immediate loss of access. Altogether, about 650 users were identified in the data base. Beginning in January the Network Control Center collected and processed the available TIP accounting data (i.e., data pertaining to those TIPs running software version 327). Sample accounting summaries were produced and sent to ARPA for review. 1c

TIP software version 327, in addition to the access control and user accounting code, contained improved diagnostic messages, corrected several minor errors, and considerably tightened the checking performed during the initiation of communication between a user terminal and a service-providing host. This last improvement, in fact, revealed protocol violations in a few hosts which had been in use for some time, including the PDP-15 at ARPA> Unfortunately, once the TIP detected a protocol violation it merely stopeed attempting to being communication (informing the user of this action) rather than attempting to "clean up" the faulty communication. This led to significant problems in attempting to access these Hosts from TIPs, and Version 327 was modified to take additional corrective actions when protocol violations were detected. 1d

Within a short time after release of Version 327 to all network TIPs, several problems became evident, as described below: 1e

- - - - - - - - - - 1f

*The text in this section is taken from our Quarterly Technical Report for the first quarter of 1975. 1g

1) Most TIPs are equipped with 28 kilowords of core memory; of this 16K is dedicated to the IMP and the remainder to the TIP. The 12K TIP core must accommodate both the TIP code (which occupies the majority of the space) and terminal buffering. The new code needed for the access control and user accounting mechanisms reduced the amount of space available for terminal buffering (in a 28K TIP) to about two-thirds of that available with the preceeding software version. Although this buffer reduction occurred in all TIPs, its effects (frequency of the user typing fast enough to completely fill his input buffer and noticeable "stuttering" on output) were most strongly felt at those TIPs supporting large numbers of terminals. 1h

2) Several organizations had failed to submit user authentication data prior to installation of version 327. In many cases, the users at these organizations had names (particularly last names) identical to the names of users whose authentication data had been entered in the data base. When these users identified themselves by last name only, the authentication system demanded that they submit the password corresponding to the given last name, but of course the user did not know this password. In addition, many users perceived the mechanism for modifying the authentication data base (a mechanism which involved the user, his organization, RML, ARPA, and BBN) as cumbersome and unresponsive. 1i

3) Use of a service host computer from a Terminal IMP required the user to first authenticate himself to the TIP, next open a logical connection to the host, and finally authenticate himself to the host before actually beginning to make use of the host's services. Although the actual time and effort required of the user to complete these steps was not large, many users had strongly negative reactions to this process of "double login"; rather than perceiving the two instances of authentication as providing additional security many users perceived the process as forcing them to do the "same thing" twice. 1j

Due to the problems described above, ARPA requested us to remove the mechanisms for access control and user authentication from the TIPs pending review and modification of the problem area. First, because of the terminal buffering problems, we reverted to the preceding TIP software version at a few heavily used TIPs which had only 28K of core memory. It is anticipated that each of these machines will eventually be retrofitted with an additional 4K words of core memory (we have submitted proposals for most of the necessary memory units) and that version 327, or subsequent software versions, will be installed at each such site as memory becomes available. 1k

Second, the access control and user accounting software in version 327 has been "turned off" by program patch in all other TIPs except at BBN. Thus, at most sites, although the code is physically present, the users are not confronted with a requirement to "login" to the TIP or with the problems of 1l

modifying the authentication data base. We have retained the access control and user accounting mechanisms at BBN so that we can continue to gain experience with, and discover desirable modifications to, these mechanism. 1m

Third, we have suggested to ARPA an administrative mechanism which we belive will make substantial improvements to the procedures for modifying the authentication data base. 1n

Finally, we have proposed to ARPA further modifications to the TIP (and to the TIPSER/RSEXEC) which would make it possible for service hosts to learn the identity of a TIP user based on the authentication data provided at the time of the TIP "login" rather than requiring the user also to authenticate himself to the host. Hosts choosing to make use of this mechanism would be required to modify their own software, and other hosts desiring to retain their existing authentication mechanisms need not make these changes. 1o

## 2. TIP Version 337 1p

TIP program version 337 is currently running at all TIP sites which had ended up running TIP version 327 as discussed in the previous section; further, as additional memory has been added to TIPs running the version before version 327, these TIPs have been going directly to running version 337 without the intermediate step of version 327. TIP version 337 has little that is new. It is primarily a clean reassembly of version 327 including assembling in all the patches that were made to version 327 and including fixes to problems which were discovered in version 327 but too difficult to patch. To enumerate the changes briefly: a) a suspended connection is cleared correctly in the case when its link gets reused by the remote host for some other connection; b) the TIP's "logger" will allow ICPs to distinct sockets on a single host to proceed in parallel; c) the "logger" can abort ICPs more cleanly; d) suspended and restored TENEX connections are handled more cleanly; e) minor bugs in the user accounting and authentication code were fixed; e) a number of miscellaneous minor bugs unrelated to network performance were fixed; f) routines were added to enable TIP tables to be sent through the network to the Network Control Center; and g) the copy-down loop in the TIPs RFNM logic was modified to allow the IMP (which co-resides with the TIP) to service I/O interrupts while the TIP is in its copy loop. The mechanism of point f has proved very useful for obtaining status and diagnostic information on the TIP. The change of point g was made to benefit the IMP. As with TIP version 327, TIP version 337 has the user authentication and accounting code turned off by program patch except at BBN where we are continuing to gain experience with and tune this mechanism. 1q

3. TIP Ports "Running Open"* 1r

In our Quarterly Technical Report No. 7 we reported on the development of a Multi-Line Controller modification to suppress the transmission of continuous breaks which are caused by turning off some types of terminals without first closing their network "connections". During the fourth quarter of 1974 and the first quarter of 1975 we installed this modification in almost all Multi-Line Controllers (four remain to be modified), including the MLC in the Tymshare TIP. However, the investigation of performance problems revealed that, at times, continuous streams of break characters were still being transmitted by the Tymshare TIP. In this case the terminals were connected to the TIP via leased communications circuits and 4800 baud modems; due to the high speed of the modems the port was operated in external clock mode. 1s

Since the lines were leased rather than dialed up there was no automatic hangup provided by the modems when the terminals were powered down and, in fact, the modem/terminal combination was wired in such a way that the Data Terminal Ready signal was always held on at the TIP end. However, the modification made to the MLC to suppress continuous breaks was designed on the assumption that externally clocked devices would provide clock signals only when there was meaningful data to be transmitted; this assumption not only seemed reasonable to the designers but also permitted an extremely low cost modification design. We have begun to consider changes to the MLC design which would enable the MLC to suppress continuous breaks from externally clocked devices, but it was felt that the best solution to this particular problem was for SRI to modify the terminal/modem interface to use the Data Terminal Ready signal to indicate the on/off status of the terminal to the TIP. 1t

4. Too Fast TIP Clock 1u

One of the diagnostic mechanisms we have developed is a facility for real-time monitoring of IMP/TIP "load average". One of the results of this monitoring was the discovery that the Tymshare TIP when totally idle appeared busier than any other totally idle TIP in the network. We quite rapidly came to the hypothesis that one of the real time clocks (there are two, one for the IMP and one for the TIP) was interrupting more often than it was supposed to be. With a trivial program patch to count the frequency of the clocks it was found that the TIP clock was running at twice its nominal frequency and therefore causing interrupts twice as often as it should and therefore wasting lots of machine cycles running all the TIP code which runs off the TIP clock interrupt more often than need be. A field engineer had
- - - - - - - - - - 1v

*This section is taken from our Quarterly Technical Report for the first quarter of 1975. 1w

little problem correcting the hardware difficulty which caused the clock to run at the too fast speed. As a result of this discovery at Tymshare, all of the other IMP/TIP clocks in the network were also tested, and no other was found to be operating at the wrong frequency. Further, the Network Control Center has established a procedure for routine periodic testing of all the clocks in the network for correct frequency. 1x

5. The Link 0 Blocking Phenomenon 1y

Link 0 is used by the host to host protocol for all its control messages. In particular, ALLocate control messages must flow from the TIP to another host if data is to flow from the other host to the TIP (to be printed on a TIP terminal) on a data connection. A phenomenon we have observed is that when a number of terminals on a given TIP are simultaneously attempting to print output from the same other host, terminal printing will be frequently interrupted for a second or a fraction of a second. This phenomenon has been traced to a problem with contention among the several terminals for link 0 between the TIP and the other host: a terminal interrupts printing because the TIP has received no further output from the other host; the host has not sent further output because it has not received the necessary ALLocate message from the TIP; the TIP has not sent the ALLocate because link 0 to the other host is blocked waiting for a RFNM for a previously sent ALLocate; and host to host protocol currently prohibits a second ALLocate from being sent while the RFNM for a previous ALLocate is still outstanding. Thus, although the other host is ready to send more output and the TIP is ready to print it, the entire system must wait for the outstanding RFNM to arrive, for the necessary ALLocate to be sent and find its way across the network to the other host, and for the now-allowed data message to traverse its way back across the network to the TIP. We have developed an "extension" to the protocol which allows the TIP to send several ALLocate messages on a single link 0 without waiting for the RFNM for each ALLocate before sending the next ALLocate, thus removing much of the latency from the ALLocate sending process and enabling smoother 1z

data output from the other host to the TIP. We have implemented this extension, and a version of the TIP software including this extension is currently undergoing final checkout prior to release of the new version throughout the network (at least to the sites which would otherwise be running version 337). We have been fortunate in discovering a mechanism which does not require any hosts other than the TIP make changes to their Network Control Programs. 1a@

6. Optional Removal of the 2741 Handler 1aa

A number of times in the past we have pointed out the possibility of removing the TIPs capability to handle IBM 2741 terminals (and their equivalents) on an optional basis. That is, for each TIP site a decision could be made as to whether the TIP 1ab

was to have the capability to handle 2741s or not. At sites where is was decided not to support the 2741 capability in the TIP, a significant amount of memory normally dedicated to 2741 handling could be used instead for terminal buffering. Although this option has been suggested several times in the past, before recently we have not managed to actually begin implementing the option. Recently, while constructing the mechanism which leads to the solution of the link 0 blocking phenomenon discussed in point 5 above, we figured out a relatively easy way to finally implement the 2741 removal option. In fact, the version of the TIP containing the mechanism to get around the link 0 blocking phenomenon is already a significant step toward the 2741 removal option, and changes we are planning to make to the TIP in other areas will naturally result in certain additional TIP structures being changed which will bring things very close to the point where removal of the 2741 capability is a real option. 1ac

7. TIP ALLocate and Buffering Strategy 1ad

There has been an exchange of messages between interested parties at SRI and ourselves regarding the TIP's strategy for buffering traffic arriving from another host for printing on a TIP terminal and the TIP's strategy for sending the host to host protocol ALLocate messages which control the flow of data from the other host to the TIP. For completeness we reproduce the message correspondence here. 1ae

*****Message from Irby of SRI -- March 18, 1975***** 1af

Our understanding of the TIP buffer allocation policy is as follows: 1ag

The TIP has a pair of output buffers of equal size, say 800 bits (100 characters) for each terminal. The TIP initially allocates to the sending host 1 message and 800 bits. One of the buffers is always being used to output to the terminal while the other is used to accept data from the sending host. 1ah

Lets call the buffer currently pointing at the terminal the Tbuf and the buffer currently pointing at the network the Nbuf. 1ai

When the first message arrives from the network that data is put into Nbuf. 1aj

The buffers are toggled and a new allocation is sent of 1 message and "l" bits where "l" is the length of the previous message. The buffer just filled, now Tbuf, is output to the terminal. When the next message arrives from the network it is put into Nbuf. 1ak

When Tbuf is empty the buffers are toggled and a new allocate is sent of 1 message and "l" bits. 1al

We do not understand why the message allocation is limited to one at a time. It seems to us that it would be possible to allocate several messages and append the data that arrives to the data already in Nbuf.
********** 1am

*****Message from Walden of BBN -- March 19, 1975***** 1an

WE DO THE TIP ALLOCATION THE WAY WE DO BECAUSE IT IS EASY THIS WAY -- DOUBLE BUFFERING IS SOMEWHAT SIMPLER THAN A CIRCULAR BUFFER WHICH IS WHAT YOU HAVE IN MIND AS THE ALTERNATIVE. 1ao

OUR ASSUMPTION IS THAT THE SENDING HOST, IF TRYING FOR HIGH

THROUGHPUT, WHICH IS WHAT YOU WANT WITH YOUR DISPLAYS, WILL SEND MESSAGES FULL TO THE ALLOWABLE BIT ALLOCATIONS -- THIS SHOULD BE NO BIG PROBLEM GIVEN THE SMALL SIZE OF THE TIP BUFFERS (I REALIZE THAT TENEX MAY NOT ALLOW THE USER TO CONTROL HOW FULL MESSAGES ARE AND TO THIS EXTENT OUR ASSUMPTION IS NOT REALISTIC). 1ap

TO GO TO A CIRCULAR BUFFER SCHEME PROBABLY ADDS SOME WORDS OF CODE (MAYBE LESS THAN 100). HOWEVER, SOME OF THESE WORDS ARE IN THE INNER LOOP (I.E., END TESTS ARE HARDER IN A CIRCULAR BUFFER SCHEME WHERE THE BOUNDARIES BETWEEN MESSAGES VARIES AROUND IN THE BUFFER AND WHERE ONE HAS TO BE CONCERNED ABOUT WRAPING AROUND THE END OF THE BUFFER. WORSE, GOING TO A CIRCULAR BUFFER SCHEME MEANS ONE HAS TO REMEMBER EXACTLY HOW MANY OUTSTANDING ALLOCATES THERE ARE SO RETRANSMISSION OF ALLOCATES CAN BE DONE CORRECTLY -- THIS WILL REQUIRE AT LEAST A NEW (64 WORDS TIMES 11 BITS PER WORD) TABLE TO HOLD THE ADDITIONAL BIT ALLOCATE MEMORY. 1aq

GIVEN THE SMALL SIZE OF THE TIP BUFFERS ALREADY, IT IS NOT CLEAR THAT THE ADDITIONAL LOSS OF MEMORY TO GO TO A CIRCULAR BUFFER SCHEME WOULD BE COMPENSATED FOR BY THE LESS BREAKAGE THAT A CIRCULAR BUFFER SCHEME OFFERS OVER A DOUBLE SCHEME. 1ar

FINALLY, GIVEN SUFFICIENT BUFFERING, IN THE TIP AND A HOST PUSHING AS HARD AS ALLOWED, DOUBLE BUFFERING SHOULD BE ABLE TO COVER ANY NETWORK (AND OTHER) DELAYS. I THINK THE FUNDAMENTAL PROBLEM, GIVEN THE LARGE DELAYS ACROSS THE NET AND THROUGH THE HOSTS, IS LACK OF MEMORY, NOT THE ALLOCATION STRATEGY. ANY ALLOCATION STRATEGY WOULD WORK POORLY WITH THE SMALL AMOUNT OF BUFFERING AVAILABLE ON MANY TIPS. ********** 1as

The message exchange reproduced above is pretty much self explanatory, although one technical point was not covered adequately; we elaborate on this point now. Assume the available TIP buffers for a port are well matched to the speed of the terminal connected to the port and to the network round trip time (i.e., printing half the buffer takes the same time it takes for an ALLocate to go to the sending host and for a data message to make its way back across the network to the TIP -- it is unlikely that the TIP will have more than this much buffering due to its very limited buffering capacity). In this case, the TIP's buffering and allocation strategy is optimal. The sending host cannot in general know the speed of the terminal to which it is sending. Therefore, the only clue the sending host has available about when the TIP is ready for more data is receipt of an ALLocate. As was mentioned in the second of the above messages, if the sending host follows a general policy of sending messages of length of less than half the total available buffer space, smooth (i.e., maximum rate) output cannot be achieved, because the TIP quickly prints the data in the small message it has received and has to sit idle while the ALLocate travels from the TIP to the host and a data message travels from the host to the TIP. Possibly less obvious is the fact that if the sending host is allowed to send messages with length greater than half the available TIP buffering space and the host frequently follows a policy of sending such larger messages, smooth (i.e., maximum) throughput again can not be achieved, because after the TIP has printed the large message it has nothing to print while the ALLocate is sent to the host and more data is sent back to the 1at

TIP. The implication of these facts are that it is difficult to find a simple strategy of sending allocations which permits the sending host to effectively take advantage of the incremental possibilities of a circular buffering strategy. It is true that if for one reason or another the sending host cannot generally send the maximum data allowed by the TIP's allocation, then the TIP's double buffering system is less flexible from the point of view of the sending host than would be a circular buffering system. 1au

In any case, we have no current plans to change the TIPs buffering and allocation strategy. 1av

8. TIP Bandwidth (i.e., Throughput Capacity) 1aw

We have recently spent a significant amount of effort attempting to understand the TIP's bandwidth (or throughput capacity). We have counted instructions in the TIP to discover the cost of various TIP functions and we have constructed mathematical expressions which relate these costs to the various parameters of the system (e.g., number of terminals, mix of terminal speeds, available buffering, efficiency of host packing of data into messages, average number of ALLocates packed into a single control message from the TIP to a host, etc.). Unfortunately, the results are very sensitive to the choice of 1ax

values for the various parameters. For instance, the capacity of a TIP may be less than 10Kbs in a configuration with 63 ports all doing output at 150 baud with each message for these ports only containing one character. On the other hand, if the ports are running at 1200 baud and the size of the messages is increased by a factor of eight, the TIP's capacity may be four times as great. Other mixes of parameter values result in still higher capacity results. Our present problem then is to turn our mathematical expressions into graphs which provide insight into the capacities of which the TIP is capable and where on the curves any given site is running. 1ay

There have been some immediate results of our TIP capacity investigations. For example, we have confirmed (as we always suspected) that the TIP's code for sending ALLocates is very costly to run. Further, because of the TIPs allocation strategy, this code is run very often (approaching once for every message that is received). And because of the TIP's limited buffering capacity, messages are received very often. We are studying methods of reducing the cost of the TIP's ALLocate sending mechanism. This example is typical of the sort of vicious cycle in which the TIP finds itself. Because the TIP has limited space, its data structures and code tend to be optimized to take little space and therefore generally are very costly to use. But because of the TIP's limited memory for buffering, the very costly (in terms of throughput capacity used) code must be run very often. the very costly (in terms of throughput capacity) code must be run very often. 1az

Also as a result of our capacity investigations we have discovered that the TIP has been looping through the IMP background loop more often than absolutely necessary resulting in a 15% decrease of TIP capacity in some cases. It has also been discovered that certain IMP computations are taking an inordinate fraction of the computer effectively reducing the TIP's capacity. Steps are being taken to correct these problems, and we will continue to be on the lookout for other such problems as we conclude our study of the TIP's capacity. 1b@

8. Conclusion 1ba

We have found a number of problems and sub-optimalities in the performance of specific TIPs or with the TIP system in general. However, we have found no fundamental problem with the TIP's design or the way it has been operating that would not be alleviated with the addition of TIP memory and the reduction of the delay across the network and through the hosts with which the TIP is communicating. 1bb

(J32527) 16-MAY-75 06:29;;;; Title: Author(s): Duane L. Stone/DLS; Distribution: /MAW( [ INFO-ONLY ] ) DFB( [ INFO-ONLY ] ) RFI( [ INFO-ONLY ] ) JLM( [ INFO-ONLY ] ) TFL( [ INFO-ONLY ] ) EJK( [ INFO-ONLY ] ) FJH( [ INFO-ONLY ] ) ; Sub-Collections: RADC; Clerk: DLS;

DLS 16-MAY-75 07:04  32528

TENEX Paging & Efficiency

A message received from Lynch at SRI-AI, which sheds some light on paging tradeoffs in the TENEX environment.

29-APR-75 1010-PDT LYNCH at SRI-AI: TENEX Efficiency Distribution: NET PERFORMANCE TECHNICAL GROUP [BBN]<MCKENZIE>NPTG.TXT:, untulis, victor at bbnb Received at: 29-APR-75 10:30:10-PDT 1

I sent the following to Bob Balzer as a response to his question to me about Lisp response on KI/KA systems at ISI. It occurs to me that most of you are more involved with the issues raised in the note than are the Lisper's are. 1a

29-APR-75 08:55:17-PDT,7832;000000000000 Mail from SRI-AI rcvd at 29-APR-75 0855-PDT Date: 29 APR 1975 0852-PDT 1b

From: LYNCH 1c

Subject: Re: KI TENEX Lisp Measurements 1d

To: BALZER at USC-ISIB, LYNCH, LICKLIDER at ISI, RUSSELL at ISI, 1e

To: PEPIN at ECL, DALE at ISI, SCHULZ at SUMEX-AIM 1f

cc: HARTLEY at BBN, LEWIS at BBN, GOODWIN at BBN, 1g

cc: BOBROW at BBN, NICKERSON at BBN, BALZER at ISIB, 1h

cc: GREENFELD at ISIB, FEIGENBAUM at ISI, ELLIS at ISIB, 1i

cc: TEITELMAN at PARC, DEUTSCH at PARC, MASINTER at PARC, 1j

cc: BOBROW at PARC, WILBER at AIC, BOYER at AIC, 1k

cc: GARVEY at AIC, RAPHAEL at AIC, LYNCH at AIC, 1l

cc: HEATHMAN at SUMEX-AIM, NII at BBN, KRD at SU-AI 1m

In response to the message sent 25 APR 1975 1644-PDT from BALZER at USC-ISIB 1n

Bob, I fully sympathize with your puzzled reaction to the Lisp benchmark runs that were made on the various KI systems and your subjective feeling that Interlisp runs faster on your KA system. Your suggestion/request that I redo the benchmarks and measure the elapsed time on each system is a tall order. I will attempt to explain why I have not done such a thing with any satisfactory results. 1o

Timesharing in a demand paging environment is a complex phenomenon. The "classical" parameters that are considered figures of merit are described by Kleinrock, Denning and Coffman in many papers over the past decade. Recently Kleinrock has rather solidly formalized much of his queueing theory and timesharing considerations into a set of books that are very worthwile reading for the serious student of Computer Science. Unfortunately, most of what is considered important to the theoreticians is not measured directly by TENEX. 1p

In the large, a timesharing system is a server that sees requests for cpu and memory space (and manipulation of files on secondary memory devices). A simple scenario is that a request for service is honored by having the server (TENEX) dole out its resources to each requestor in a manner that is "fair" in some sense. The usual method is to give each user a "burst" of cpu time and then to go on to the next user requesting service and give it a burst while the others wait. The nature of the waiting is what is most studied by the theoreticians. Now if it happens that while a user (requestor) is utilizing its "burst" of cpu time it encounters a "page fault" then the cpu server has to give up on processing this particular user and pass it on to the server that handles the requests for "not-in-core" pages. This server will be called the "swapper". Thus it is seen that a timesharing system contains a number of internal servers who give resources to the user. The presumed independent variables of a server system are: 1q

(1) arrival time of user, and, (2) amount of time consumed by requestor before it leaves the system. 1r

The above variables give rise to the following "figures of merit": (more exactly - "parameters of merit") 1s

N = LAMBDA * T 1t

Where 1t1

N = average number of customers in the total system (those being served plus those waiting for service) 1t1a

T = average time spent in the system/customer (both being served and waiting for service) 1t1b

LAMBDA = average arrival rate of customers to the system 1t1c

In TENEX the Load Average is defined as the sum of those processes (customers) who are either: 1u

(1) Running now (max of 1 with one cpu involved), (2) Waiting for the swapper to bring in a page, (3) Waiting for the cpu only [i.e., (2) is satisfied], and, (4) Waiting for an initial core allocation. 1v

If a process (customer) is in one of the first three classes it is said to be "in the balance set". That means that the sum of the recent average core usage requirements (working sets) is not greater than the amount of physical core available for all customers to share. Thus it can be seen that the Load Average is a good candidate for serving as N in the above central equation. However, it does not say much about what a customer can expect for T if the customer does not know anything about LAMBDA! It also completely ignores the various priority schemes that may be present in the cpu-allocation mechanism that usually try to favor those customers with small LAMBDA (the so-called "interactive" user). It also masks the characteristics of the swapper rather well. In fact, if the swapper is "slow" then one can see that customers who need to use the swapper often (Interlisp) do not get to use the cpu very often and thus their average service time (T) is quite large compared to the customer who asks for the same amount of cpu time but does not demand much of the swapper. 1w

In TENEX there is no attempt to "prioritize" access to the swapper on any basis. It doesn't do much good to give a swap-bound customer priority to the cpu if it is spending most of its time waiting for the swapper to serve it. Instead, one should simultaneously try to improve the performance of the swapper in absolute terms and possibly put a priority scheme on access to the swapper. Swapper performance can be improved by two methods: 1x

(1) Faster hardware, and, (2) Smarter software 1y

Faster hardware usually means getting a device that has the smallest average access time to a page that one can buy. Smarter software means that one should establish as many servers to the swapper as there are independent access channels to it. This last scheme is very important for swapper performance. Denning's article on scheduling swapper activity shows dramatic improvements for loaded systems. (SJCC 1967) 1z

Specifically, the KI systems that were measured all have different swappers. The cpu-servers are all similar in that they are running a 131 version that has been modified to give interactive jobs very high cpu-priority. The swappers are summarized to the best of my knowledge as follows: 1a@

ISI-KI (ISIB) 1 3330 disk drive with swap space at end of pack 1a@1

ECL-KI 4 3340 drives with swapping across middle (except that the swappable monitor is in the fixed head part of two of the drives) 1a@2

SUMEX-KI 1 drum (1800 rpm, 13 pages per rev) 1a@3

IMSSS-KI 1 drum (3600 rpm, 16 pages per rev) 1a@4

ISI-KA (ISIA) 2 3330 disk drives with swap space in center of packs 1a@5

SRI-AI-KA 1 drum (1800 rpm, 18 pages per rev) 1a@6

The drum systems are all running software that has a separate server for each sector (page) around the revolution. 1aa

The exact amount of performance improvement for any given system as a function of swapping server efficiency is not easy to calculate. I am still working on the precise models for describing each of the above systems and will report my results at a later time. I hope to be able to develop a model for each general class of hardware and appropriate software to derive the maximum throughput capacity for all reasonable configurations of equipment. The formulas involve making assumptions about the "page-fault-rate" for each system. I will have to make a family of curves for the range of cases that are useful to know. If anyone has any thoughts on the subject, please let me know them. Denning's original formulas are known to be a bit incorrect and Coffman's formulas for waiting time are lengthy (JACM Jan 1969). Also, modeling a 3330 system is tricky because it really is two systems (I think). 1ab

All of this suggests that there is much room in the TENEX system for adopting service policies at each of the internal queues that will favor one class of service over another and that there is no single parameter that will suffice to describe to a user what he can expect from the system. However, it appears that the addition of a few meters (and their appropriate display) will give a user a much clearer picture of the service he can expect. 1ac

Regards, Dan 1ad

(J32528) 16-MAY-75 07:04;;;; Title: Author(s): Duane L. Stone/DLS;
Distribution: /JPC( [ INFO-ONLY ] ) TFL( [ INFO-ONLY ] ) EJK( [
INFO-ONLY ] ) WER( [ INFO-ONLY ] ) FSL( [ INFO-ONLY ] ) MAW( [ INFO-ONLY
] ) ; Sub-Collections: RADC; Clerk: DLS;

aclnowledgement

received 32522 and 32507 1

aclnowledgement

(J32529) 16-MAY-75 08:01;;;; Title: Author(s): Robert D. Archer/RDA; Distribution: /FGB( [ ACTION ] ) ILA( [ INFO-ONLY ] ) ; Sub-Collections: NIC; Clerk: RDA;

PCB 16-MAY-75 08:10 32530

THANKS RANK

THANKS FOR FIXING DAVES IDENT. SAME FOR ANNA.
regards......paul....this may fly the fist shot 1

THANKS RANK

(J32530) 16-MAY-75 08:10;;;; Title: Author(s): Paul C. Bishop/PCB; Distribution: /FGB( [ ACTION ] ) JDB2( [ INFO-ONLY ] ) AW2( [ INFO-ONLY ] ) ; Sub-Collections: NIC; Clerk: PCB;

attempt number three

attempt number three

(J32531) 16-MAY-75 08:14;;;; Title: Author(s): Paul C. Bishop/PCB; Distribution: /FGB( [ ACTION ] ) JDB2( [ INFO-ONLY ] ) AW2( [ INFO-ONLY ] ) ; Sub-Collections: NIC; Clerk: PCB;

RH 16-MAY-75 09:25 32532

Weekly Report of My Activities

Finished up the course reports with JMB. Worked with Martin Hardy and Rod Bondarunt trying to repair broken Lineprocessor. Met with JCN regarding ARPA etc. Sent out viewgraph recommendations. Shipped out terminals and equipment from SRI-WDC and ARPA to Menlo Park. Worked on understanding and using the ARPA MRAO's. Reviewed Course III for class next week. Made a somewhat successful attempt at writing a commands branch. Assisted a few users via links. One user in particular, Paul Bishop (PCB) of NCSL is having an open-house at his lab tomorrow and was wondering if I could be on the system for a while tomorrow so that he could link to me. Took off 4 hours sick leave today. 1

Weekly Report of My Activities

(J32532)   16-MAY-75 09:25;;;;   Title:  Author(s): Rita Hysmith/RH;
Distribution: /US( [ INFO-ONLY ] ) ; Sub-Collections:  SRI-ARC US;
Clerk: RH;

Larry, Here is a talk I just had with JimShores

Shores 5-16-75 1

Tentatively, yes they would be interested in doing H-FE process in ELF and default driver. He will talk to his people and call back next week. If he is still interested, we will get together and go into further details. 1a

His people are John McGilvary who is doing C/SP simulation in PDP 11 and must learn ELF in detail before doing so. Other PDP 11 people would be available if funddng were. (The new Advanced Combat Systtems Lab headed by Roger Praeger has lots of PDP 11 people; this was recently created in organizational shift) 1b

Jim plans to go in by C/SP simulation route. Still in process of collecting info on it. Not sure how process to process would work. They are starting off by worrying about time sharing and working up. 1c

He will make software available to those interested. However, they are currently running level 31 EXEC and unsure what effect level 32 will have, etc. NBS is currently doing C/SP simulation using another mini. 1d

(J32533) 16-MAY-75 12:27;;;; Title: Author(s): Frank G. Brignoli/FGB; Distribution: /ILA( [ ACTION ] ) ; Sub-Collections: NIC; Clerk: FGB;

NOTES ON DEVELOPMENT OF DDPC PROGRAM

DOCUMENT DEVELOPMENT, PRODUCTION, AND CONTROL 1

David R. Brown, Director
Information Science Laboratory
2

ISL-ARC cooperation in the development of a program in the area of document development, production, and control (DDPC) has been the subject of numerous discussions, meetings, and notes during the past year or so. At the present time, we are seeking the definition of a program that we can promote, and a marketing strategy. This note is supposed to be a step toward that objective. 3

As background, the following should be noted. 4

1. ARC has promoted a growing community known as the Knowledge Workshop Architects Community (KWAC). Members of this community are NLS enthusiasts, and their number is growing. They are potential clients for a future program in DDPC. 4a

2. SRI is an active member of the KWAC. Several groups within SRI are using NLS (and have saturated the available resources). 4b

3. ISL has developed and operates a text-editing system known as MAE on the PDP-11/20. This system is currently used by a number of groups at SRI, including Chemical Information Services and the SIME-DIME project. 4c

4. An SRI-sponsored project for the preliminary design of an Institute-wide, automated, text-handling system was completed in 1974. The project is presently suspended and probably will not be funded by SRI Central Administration in 1975. 4d

5. A proposal to NSF for the development of an experimental editorial processing center was reviewed favorably by NSF. The proposal was recently withdrawn by SRI for several reasons. 4e

a. SRI was unwilling to share in the costs as required by the proposal. 4e1

b. The objectives imposed by NSF appeared to be far from optimum (from SRI's point of view). 4e2

With this somewhat mixed bag of a background, we now want to proceed to develop a program, and (hopefully) bring all of our talents together in the process. 5

We want to have a program in the development of systems for DDPC, sustained by a multitude of clients with similar problems and interests. Our work for a prototype client would be the kind of work we proposed for the Institute-wide, automated, text-handling system. 6

The program we visualize would not consist of some number of separate, one-shot projects. It would be a sustained, cooperative, multiclient project, but be flexible and tailored to the needs of each individual client in the program. Each client would start from a different place and go through phases of system analysis, system design, implementation, operation, evaluation, etc. Not all clients would have to be in the same phase at the same time. (The phases tend to overlap and become blurred anyway.) A client undertaking a major upgrading, such as the one we proposed to SRI, would be in for a three or four-year effort. At the end of that time, the client would be urged to stay in the program for continued benefits and upgrading of his system. 7

Our method of approach is: 8

1. Prepare a description of what we intend to sell 8a

2. Formulate a marketing strategy 8b

3. Target prospects 8c

4. Bring them into the program. 8d

We are presently in the midst of (or perhaps the beginning of) step 1. 9

I propose that we not attempt to complete each step in comprehensive and final form, but proceed rapidly to put something together, to develop a preliminary marketing strategy, and to contact some local potential clients. I think we should be making our first contacts with prospects before July 1. 10

We need a promotional document that we can give to potential clients that describes the program. It should be brief (a few pages), and typewritten on blue banner. This would not be a proposal. A proposal to a prospective client would have to come later, after we had more knowledge of that prospect's specific problems. 11

The program is to be carried out primarily through the Information Systems Group of the Information Science Laboratory, with some help from ARC, and possibly other parts of SRI in the future. The promotional effort is expected to cost something over $10,000 for the first six months. (By the end of six months we should have some idea about the success of the program, and make a decision about its continuation.) 12

The program would offer an option to each prospective client to join the special interest community (SIC) of the KWAC for DDPC. See <GJOURNAL,32404,>. Thus, the program might include two kinds of clients, those who choose the option, and those who don't. (It's conceivable some would choose to join the SIC and not take the other part; so be it.) 13

The program could have some important byproducts in the form of hardware and software. These might be developed by SRI, or obtained through the preparation of RFQ's, evaluations of bids, and awards of contracts to outside organizations. 14

NOTES ON DEVELOPMENT OF DDPC PROGRAM

(J32534) 16-MAY-75 13:26;;;; Title: Author(s): Kathey L. Mabrey/KLM;
Distribution: /DOCPLAN( [ INFO-ONLY ] ) GAS2( [ INFO-ONLY ] ) DRB( [
INFO-ONLY ] ) ; Sub-Collections: NIC DOCPLAN; Clerk: KLM;
Origin: < MABREY, DDPC-PROGRAM.NLS;4, >, 16-MAY-75 13:21 KLM
;;;;####;

NWC Conversation

Oh well....

NWC Conversation

Larry, 1

Have just finished talking to John Zenor from NWC. First the good news: he is willing to co-operate with a person doing H-FE in the PDP-11 in developing the NWC driver. He doesn't have anyone working on software yet, only a UCSB graduate student to generate listings, etc. Next year, they hope to have one software and one hardware type working on networking. 1a

Unfortunately, contrary to my earlier impression, he wants to go in as a Communications Terminal Module Controller (CTMC). This controls 32 or so CTM's which can be ordered to meet a variety of specificaions (e.g., pass raw data thru to the 1108). He feels that these can handle interactive, FTP, RJE, and process-to-process via read/writes. He thinks C/SP is a poor way to go and,in any case, they don't have C/SP software on their system. He doesn't know much about C/SP but suggested we contact JPL for info on C/SP as they apparently have done a lot of software work on it. NWC wants to go in via PDP channel interface to CTMC. 1b

He also mentioned, for informaion only, that Jim Shore's shop was until recently strictly batch oriented and they are just now getting into demand processing. 1c

He also suggested we look into the DEC comunications protocl. Apparently, they have developed a more sophisticated protocol based on their DDCMP which they hope to supply as standard with their O/S's. Also, they are developing a ROM for "across the net" type loading. 1d

Let's talk Monday. 2

Also, he wants to know who pays for channel interface. 3

Frank 4

NWC Conversation

(J32535) 16-MAY-75 17:07;;;; Title: Author(s): Frank G. Brignoli/FGB; Distribution: /ILA( [ ACTION ] ) ; Sub-Collections: NIC; Clerk: FGB;

DCA Visitors - May 20-22, 1975

Robert Brownfield and Ed Alexander from DCA will visit ARC at 9:30 Tuesday to discuss the NIC contract and aspects of the transfer of the Arpanet management to DCA. (To those who were involved - they have reservations at the Holiday Inn and will be arriving Monday. They do not need an airport limo as they will have a car) 1

DCA Visitors - May 20-22, 1975

(J32536) 18-MAY-75 15:22;;;; Title: Author(s): Elizabeth J. Feinler/JAKE; Distribution: /JCN( [ ACTION ] ) DCE( [ ACTION ] ) RWW( [ ACTION ] ) MEH( [ ACTION ] ) RLL( [ ACTION ] ) DCL( [ ACTION ] ) SRI-ARC( [ INFO-ONLY ] ) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

Test Msg

Look in the file JJZ.MAIL for my msg
- Jim Guyton 1

Test Msg

(J32537) 18-MAY-75 18:48;;;; Title: Author(s): John J. Zenor/JJZ;
Distribution: /JJZ( [ ACTION ] ) ; Sub-Collections: NIC; Clerk: JJZ;

Test MSG

This NLS sndmsg isn't too s{imple - - look in the file
JJZ.MSG for some mail. - Jim Guyton 1

Test MSG

(J32538) 18-MAY-75 18:51;;;; Title: Author(s): John J. Zenor/JJZ;
Distribution: /JJZ( [ ACTION ] ) ; Sub-Collections: NIC; Clerk: JJZ;

ANNEX I

# SPECIFICATION OF THE SOFTWARE INTERFACE BETWEEN CDC 6000/CYBER-70 EQUIPMENT AND A PDP-11 BASED ELF SYSTEM FOR THE PURPOSE OF PROVIDING ARPA NETWORK ACCESS

## 1. INTRODUCTION

This document contains the specification of the software interface between a CDC 6000/Cyber-70 computer system and an ELF system required to accomplish the objective of phase 1 of the AFSCNET project as stated in, "A Plan for an Air Force Systems Command Computer Network (AFSCNET)," dated November 1973:

> "The objective of phase 1 is to determine whether networking within AFSC is operationally feasible and economically advantageous on a large scale."

Although not specifically referenced in the above cited plan, it was assumed at the time of its preparation that during phase 1 the ELF would support numerous peripheral devices such as disk and tape drives and both interactive and remote job entry (RJE) terminals. Due to budgetary constraints, the current, revised intention is to configure the ELF as a "bare-bones" system supporting only a keyboard/printer operators' console and perhaps a DEC tape drive. Naturally, this revision requires a reduced level of both ELF and CDC software development. The ELF software will reside in a PDP-11/40 or 11/45 interfaced directly to a CDC 6000 data channel through a DEC CDC 6000 channel coupler which is described in, "A PDP-11 Interface to CDC 6000 Series or Cyber-72/73/74 Computers," dated October 1974.

The overall objective of phase 1 will be accomplished through the evaluation of a testbed configuration consisting of at least two ELF/CDC 6600 ARPA network nodes located at currently interested AFSC computer installations (ASD, ADTC, AFWL). Each of these nodes will support the following functions:

1. Outgoing batch job file transfer - a user will be able to submit a batch job to his local CDC machine through any of the currently available input devices and direct that job file to be transferred to any remote ELF/CDC site for processing.

2. Incoming batch job file transfer - each node will accept batch jobs from other remote ELF/CDC sites, process the jobs and return the output file(s) to the appropriate destination.

3. Incoming data file transfer - a user will be able to request that a data file from any remote ELF/CDC node be transferred to the local CDC 6600 on which his job is processing. The user may then use this file as he would use any local file (catalog, dispose, etc.).

4. Incoming interactive support - each ELF/CDC node will support low speed interactive terminals (of the type currently supported locally) over the ARPA network.

During phase 2, items 1 through 3 above will be expanded to include non-ELF/CDC nodes and the ELF peripherals mentioned earlier will be supported. In addition, the support of job files to and from network remote job entry (RJE) terminals will be added. These terminals may be either parasite ELF nodes or other "standard" RJE terminals. It is currently anticipated that basic RJE terminal protocol will be mapped by the ELF into the File Transfer interface protocol contained in the document. It should be noted, however, that in phase 2 the interface herein described will probably need to be expanded to support both "enhanced" RJE service and possibly process-to-process communication.

This paper is divided into three main sections intended to cover the three major protocol functional classes:

1. General Definitions and Techniques - those procedures which are required for all connections using this interface protocol, and not uniquely applicable to either file transfer or interactive support of future processes to be incorporated.

2. File Transfer Processes (FTP).

3. Interactive Processes.

The referenced document used in designing the conventions described herein was "NIC 7104-ARPA Network Current Network Protocols", including NIC update #17759 for File Transfer Protocol conventions."

11. GENERAL DEFINITIONS AND TECHNIQUES.Center

A. ELF Protocol Support Requirements

1. Standard Protocols

The following standard ARPA network protocols must be implemented in the ELF:

TELNET

Host to IMP

Host to Host

Initial Connection

File Transfer (subset)

2. RJE Protocol

The Remote Job Entry (RJE) protocol was not specified for phase 1 due to its lack of universal acceptance and the fact that file transfer protocol can accomplish the CDC mainframe to mainframe transfer of job files adequately.

3. ELF/CDC 6600 Protocol

In addition to supporting the above-mentioned standard network "front-end" protocols, the ELF must support a "rear-end" protocol with which to communicate with the CDC 6600. This protocol will consist of a group of commands issued by the 6600 and a set of responses returned by the ELF. The protocol will be largely procedure-oriented toward the overall control of the interface and the transfer of data. It will be the ELF's responsibility to map this "rear-end" protocol into the necessary ARPA standard protocols. It is our intent that this "rear-end" interface be applicable to other non-CDC computer systems desiring to access the ARPA network by way of an ELF front-end.

B. Processing Classes

Each type of processing handled by the ELF will be known as a processing class. The universal utility functions done by the ELF will be class 0 tasks. The other two processes described in this paper are interactive processing which will be class 1, and file transfer processing which will be class 2. As further processes are added they will be assigned classes 3 through 7. The processing class value will be used as a three bit subfield in numeric values passed between the ELF and the 6600 to aid in the consistent classification of numeric values.

C. Logical Connections between the CDC and ELF Systems

1. Definition of Line

Each logical connection between the CDC and ELF systems is called a "line". A unique line must be defined for each incoming or outgoing file transaction and for each interactive terminal that is active. The ELF will be responsible for line management, mapping lines into ARPA network connections, and internally maintaining line status tables to be delivered to the CDC 6600 on demand.

2. Line Numbering Conventions

Each line has a unique 12-bit line number of the form:

aaa bbb cccccc

where

a: bits 9-11: unused

b: bits 6-8: processing class

c: bits 0-5: line ordinal

This form yields three distinct groups of line numbers (in decimal).

Line 0 (zero): This line is for communication with the ELF itself. (See Section II.E).

Lines 64-127: These lines are reserved for interactive connections.

Lines 128-191: These lines are reserved for file transfer connections.

3. Line Status Tables

a. Structure

The ELF must maintain a line 0 status table (Section II.E.3). The ELF must also maintain a line status table for each process class with an entry for every possible active line. The generalized format of these line status tables is a contiguous block of 12-bit bytes divided into fixed length entries. Each entry consists of two sections. The first section will be used by the ELF to pass information to the CDC 6600 and can be changed only by the ELF. The second section will be strictly for the use and convenience of the CDC 6600. It can be changed only by the CDC 6600 (with the ELF's help). The data in section two is meaningless to the ELF. The detailed formats of the two sections differ for each defined process class and will be described in detail in the appropriate section of this specification.

b. Status Table Commands

There are two utility commands that the CDC 6600 can issue that affect the status tables. These are the REQUEST STATUS command and the UPDATE STATUS ENTRY command. These commands are detailed in Section II.1.

c. Status Value Conventions

Any numeric status values set by the ELF in section one of a status table entry will be in the following form:

aaa bbb cccccc

where

a: bits 9-11: unused

b: bits 6-8: processing class

c: bits 0-5: status ordinal

This corresponds to the following ranges of status values for the defined processes.

Interactive Processing, 64-127 (decimal)

File Transfer Processing, 128-191 (decimal)

D. Information Exchanges between the CDC 6600 and ELF Systems

1. Definition of "byte"

The word size of the CDC 6600 Peripheral Processor is 12 bits. This 12-bit quantity of data will be the standard unit of information to be transferred between the CDC 6600 and ELF in either direction. In this paper, this 12-bit unit is referred to as one byte.

a. Structure

Any block transferred between the CDC 6600 and the ELF will be in the form of a contiguous group of 12-bit bytes containing a check sum, a byte count, a line number, and a block identifier. A variable number of data or parameter bytes may follow. All fields within the block will be right-justified. The block format is:

Check Sum

Byte Count

Line Number

Block Identifier

Data or Parameters

2. Format of a Standard Transfer Block

a. Structure

Any block transferred between the CDC 6600 and the ELF will be in the form of a contiguous group of 12-bit bytes containing a check sum, a byte count, a line number and a block identifier. A variable number of data or parameter bytes may follow. All fields within the block will be right-justified. The block format is:

Check Sum

Byte Count

Line Number

Block Identifier

Data or Parameters

b. Check Sum

The check sum byte will be a longitudinal parity check of either the remaining three required bytes in the block (byte count, line number, and block identifier bytes) or all of the remaining bytes in the block. Section II.E.3.b. tells when each check sum is used. Even parity is used; that is, the successive application difference to the bytes being check summed and the check sum byte should result in a value of zero.

c. Contents of Header

The byte count will reflect the length of the entire block which must be a minimum of four bytes long. The line number is as described in Section II.C.2. The block identifier is a 12-bit number in the following form:

Block Identifier - aa b cc dddddd

where

a: bits 10-11: unused

b: bit 9: origin bit; if 0, this block was sent by the 6600; if 1, this block was sent by the ELF.

c: bits 6-8: processing class number

d: bits 0-5: block ordinals

Block identifiers sent from the ELF to the 6600 are in the range 512 to 1023 decimal. Block identifiers sent from the 6600 to the ELF are in the range 0 to 511 decimal.

3. Technique for Information Exchanges

All exchanges of information between the CDC 6600 and the ELF will consist of two standard transfer blocks. The first block will be issued by the CDC 6600 and will consist of a command block or a data block. The second block will be issued by the ELF as a response to the block. The ELF will signal that its response block is ready by setting the 2 bit (DEC processor is requesting a CDC Read/DEC Write Operation) in the CDC status word (see page 9 of DEC Coupler Manual). The CDC 6600 must not send two consecutive blocks to the ELF and should not function the coupler to read until the ELF has set bit 2 indicating that his response is ready. During the actual transfer, the sender will always disconnect the channel. During a DEC Write/CDC Read, the CDC 6600 will set its word count at least one higher than the maximum expected. On a DEC Read/CDC Write, the ELF should be prepared for the longest possible command or the full length of a data block as applicable.

4. Transmission Block Sizes

There is an established maximum length for all transmissions. The maximum length of data transmissions is fixed when the ELF begins a specific process (FTP, interactive, etc.) and is less than or equal to maximum values acceptable to either the ELF or the CDC 6600. The maximum length of a command block from the CDC 6600 or a non-data block or undefined length from the ELF should be an assembly parameter with an initial value of 100 bytes (not counting the header).

E. Line 0

1. Line 0 Concept

Line 0 will be used to denote the ELF. The CDC 6600 has occasions when it must request information from the ELF that does not concern an existing interactive or file transfer connection. The ELF will maintain a status table for line 0 that will contain the system configuration parameters (Section II.E.3).

2. Line 0 Commands

The commands issued to line 0 are for the overall control of the interface and for utility purposes. They do not correspond to any existing interactive or file transfer connection. The commands issued on line 0 are:

Initiate FTP Processing

Initiate Interactive Processing

Stop FTP Processing

Stop Interactive Processing

Change Processing Flags

Request Status

Open New FTp Line

Update Status Entry

Resend Last Response

Return Invalid Block

These commands are fully described in the appropriate sections of this paper.

3. Line 0 Status Block Format

a. Structure

The line 0 status block will be used to store processing flags, the number of FTP and interactive lines being used, maximum buffer sizes, and any pertinent data the 6600 might want to remember. The block consists of two sections, each of which is a fixed length. Section one is defined below. It should be initialized to zero. The processes by which these entries are given values are described later. Section two is for the use of the CDC 6600, and its length is defined by an assembly parameter whose range is 0-63 and whose initial value is 10.

The format of the line 0 status table when it is delivered to the CDC 6600 is as follows:

| | |
|---|---|
| | Check Sum |
| 19 | Byte Count (decimal) |
| 0 | Line Number |
| 516 | Block Identifier (decimal) |
| | Processing Flags (see next section) |
| | Number of FTP Lines |
| | Maximum FTP Buffer Size (in buffer units) |
| | Number of Interactive Lines |
| | Maximum Interactive Buffer Size (in bytes) |
| | 10 Bytes used by CDC 6600 |

b. Processing Flags

The processing flags are bits of the ELF which will set or clear at the request of the CDC 6600 and which control some phase of the ELF processing. These flags are initialized to zero and changed by means of the CDC 6600 CHANGE PROCESSING FLAGS command (Section II.I.5). Only two flag bits are presently defined. The other 10 bits are currently unused.

bit 0: Check Sum Bit: if zero, only the other three header bytes are check summed; if one, the entire block is check summed.

bit 1: FTP Suspend Bit: if one, do not accept any new FTP connections from local CDC 6600 or from the network.

F. General Purpose Responses

Every command received by the ELF has a response dictated by the command. There are two responses used by the ELF to acknowledge a command receipt whenever no specific information needs to be returned to the CDC 6600. These responses are:

1. Positive Acknowledgement (ACK) Block

The function of the Positive Acknowledgement response is merely to tell the CDC 6600 that the last command was understood and the ELF has finished working on it.

Check Sum

6 Byte Count

Line Number (same as issued command)

513 Positive Acknowledgement Block Identifier (decimal)

Last Command Ordinal Issued

Parameter Byte (not used at this time)

2. Negative Acknowlegement (NAK) Block

The negative acknowledgement is the ELF's way to turn down a request by the CDC 6600. This means that the ELF cannot or will not do the desired action at this time. It is not an error condition.

Check Sum

6 Byte Count

Line Number (same as issued command)

514 Negative Acknowledgement Block Identifier (decimal)

Command Ordinal Being Denied

Parameter Byte (not used at this time)

G. Procedure to be followed when the ELF receives an Invalid Command

1. Error Response Block

When the ELF receives a command or data block that is invalid for any reason, its response should always be an Error Response Block. This block is in the following form:

| | |
|---|---|
| | Check Sum |
| 6 | Byte Count of this Block |
| 0 | Line Number |
| 315 | Error Response Block Identifier (decimal) |
| | Error descriptor (see below) |
| | Byte Count of Block Received |

Values for the error descriptor parameter

1 = Incorrect Check Sum

2 = Incorrect Byte Count

3 = Undefined Line Number

4 = Unknown Command Ordinal

5 = Command issued on Type Line Number

6 = Command received out of sequence

7 = Parameter Values wrong

8 = Other

2. Returning an Invalid Block

If the CDC 6600 wishes to receive the invalid block back from the ELF, it can issue the RETURN INVALID BLOCK command and the ELF will respond with the invalid block as shown in the following diagram.

Check Sum

Byte Count of Total Block

0 Line Number

517 Block Identifier for Returned Invalid Block

(
(
( Block as the ELF received it from
(
( the CDC 6600

H. Character Conversion Convention for TELNET Character strings

1. Scope of this Convention

These conventions are limited to strings of characters received from or going to, a TELNET connection. It is not to be used to convert data being transferred via a data connection during some FTP process. The places this specification requires this convention are when the ELF is handling interactive data, the pathname parameters for FTP STOR and RETR commands, and FTP error messages.

2. Procedure to Convert Display Code Characters to ASCII Characters

When display code characters, that must be converted to ASCII characters, are received by the ELF from the CDC 6600, each byte contains two six bit display code characters. The character in bits 6-11 is to be considered the first character. Using the conversion table in section II.H.4, the ASCII character corresponding to the given display code character is the proper value to send. Where more than one ASCII value corresponds to a given display code character, the first ASCII value listed should be chosen; for example, a display code of octal 01 would convert to a ASCII code of hexadecimal 41.

3. Procedure to Convert ASCII Characters to Display Code Characters

ASCII characters are received from the network in a string and must be converted to display code and packed into a buffer with two display code characters per byte. The character received first is to be placed in bits 6-11 of the byte. If an odd number of characters is received, the ELF should fill bits 0-5 of the last byte with an octal 55, a display code blank.

The conversion table in the next section is used to convert ASCII characters in the range of hexadecimal 20 to 7E to display code characters in the range of octal 00 to 77. This is a many to one mapping in some cases. For example, both ASCII hexadecimal codes 41 and 61 are converted into a display code of octal 01. ASCII hexadecimal values 00 to 1F are not mapped into a display code character. These values are ignored.

I. Utility Commands

The commands described on the following pages are utility commands that allow the CDC 6600 to obtain information about the processing states, store information in the ELF, and handle error conditions. Each command is in the format of a standard transfer block (Section II.D.2). The descriptions contain the specific details about which line numbers are valid, what the parameters are and how many there should be, under what conditions the command is valid, and the processing requirements of the ELF. The last section is, "Status Changes and ELF Responses to Command." Status changes only refer to section one of line status tables entries, the section that can be changed by the ELF. "Responses" are the types of information the ELF is allowed to send to the CDC 6600 after receiving this command. Some response must be made to every command the CDC 6600 issues.

COMMAND: Request Status

BLOCK IDENTIFIER: 1 (decimal)

LINE TYPE: Any line

PARAMETERS: If the line number is non-zero, parameters are optional and ignored. If the line number is zero, the first parameter byte tells which status table is desired. See below for meaning of parameter.

PURPOSE OF COMMAND: To request that an entire status table or an entry from a particular status table be sent to the CDC 6600 in order to determine the present state of the processing.

WHEN CAN COMMAND BE ISSUED: If the line number is zero, this command can be issued anytime. If the line number is non-zero, the processing class for that line number must be active for the command to be valid.

ELF PROCESSING REQUIREMENT: If the line number is non-zero, select the status table entry corresponding to that line number. If the line number is zero, use the first parameter byte to determine which entire status table to select.

Parameter Values
- 0 Line 0 status table
- 1 FTP status table
- 2 Interactive status table

The parameter value of zero allows the CDC 6600 to request the line 0 status table correctly no matter which check summing mode is in effect.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: No status changes. Response is the selected status table or status table entry.

COMMAND: Update Status Entry

BLOCK IDENTIFIER: 2 (decimal)

LINE TYPE: Any line

PARAMETERS: The data to be placed in section two of the status table entry corresponding to the line number. The number of parameter bytes is undefined (maximum limit of 100 bytes).

PURPOSE OF COMMAND: The CDC 6600 is allowed to keep data defined only to itself in section two of each status table entry. This allows the CDC 6600 to change that data.

WHEN CAN COMMAND BE ISSUED: If the line number is zero, this command can be issued at any time. If the line number is non-zero, the processing class for that number must be active for the command to be valid.

ELF PROCESSING REQUIREMENT: The ELF will replace, in order, the bytes beginning at the start of section two of a status table entry with the parameter bytes. If more parameter bytes exist than bytes in section two, excess parameter bytes are ignored. If fewer parameter bytes exist than bytes in section two, the remaining bytes in section two remain unchanged.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: No status changes. The response is the status table entry just changed.

COMMAND: Resend Last Response

BLOCK IDENTIFIER: 3 (decimal)

LINE TYPE: 0

PARAMETERS: None

PURPOSE OF COMMAND: The last transfer block sent by the ELF was not understood by the CDC 6600 for some reason. It wants to have the same block sent again.

WHEN CAN COMMAND BE ISSUED: Anytime

ELF PROCESSING REQUIREMENT: The ELF should save every block sent to the CDC 6600 until the next command is received. This is due to the possibility of the RESEND LAST RESPONSE command asking for the previous block to be resent. If any command (with one exception) is received, the ELF can discard the previous response block. The exception is the CHANGED PROCESSING FLAGS command when it is changing the check sum bit. This is an exception since the CDC 6600 may want to change the check sum on the last block received and look at it again. This means that the response to a CHANGE PROCESSING FLAGS command, with the check sum bit being changed, cannot be recovered by the RESEND LAST RESPONSE command.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: No status changes. Send the last response again.

COMMAND: Return Invalid Block

BLOCK IDENTIFIER: 4 (decimal)

LINE TYPE: 0

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 received an Error Response Block from the ELF as its last response. The CDC 6600 wants to examine the block that the ELF received. Primarily, this is a debugging tool.

WHEN CAN COMMAND BE ISSUED: Only when the last ELF response was an Error Response Block.

ELF PROCESSING REQUIREMENT: Place the block that was in error in the parameter section of a Returned Invalid Block. Section II.G.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: No status changes. The response is a Returned Invalid Block.

COMMAND: Change Processing Flags

BLOCK IDENTIFIER: 5 (decimal)

LINE TYPE: 0

PARAMETERS: One byte change mask

PURPOSE OF COMMAND: This allows the CDC 6600 to change the processing rules in the ELF that are affected by the processing flags.

WHEN CAN COMMAND BE ISSUED: Anytime

ELF PROCESSING REQUIREMENT: The ELF should perform a logical difference between the change mask in the parameter byte and the present value of the processing flags in the line 0 status table. The result should be placed in the processing flags byte of the line 0 status table. If the check sum bit is being changed, do not discard the previous response.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The only status change is a possible change in the processing flags in the line 0 status table. The response is the line 0 status table.

III. File Transfer Processing

A. Objectives for Phase I Implementation

The primary objective for Phase I implementation is to be able to transfer job files between AFSC CDC 6600 sites. Secondary objectives for Phase I are subordinate to the primary objective and could be sacrificed, if necessary, due to funding limitations without affecting the primary objective. One secondary objective is to achieve the ability to recover a file whose transmission has been interrupted, without starting the transmission from the beginning. Another secondary objective is to be able to transmit and accept generalized text files. This capability would be required when communicating with non-CDC 6600 computers. These two secondary objectives are not included in this paper, but would merely be additional commands and statuses and would not change the interface structure as it will be described. Another paper is presently in the planning stage. This paper will contain the additional specifications necessary to implement the secondary objective of handling generalized text files.

B. Protocols Required

All protocols cited in Section II and a subset of the File Transfer

Protocol must be implemented. The following list of commands from the File Transfer Protocol are those which must be implemented.

Access Control Commands

USER
PASS
BYE

Transfer Parameter Commands

BYTE - Currently requires a byte size of 12 bits.

SOCK - This command is internal to the ELF software and is not affected by the requirements of the 6600/ELF interface.

TYPE - Type IMAGE must be implemented.

STRU - RECORD structure must be implemented.

MODE - BLOCK mode is needed. However, data blocks with restart markers can be ignored and data blocks with the "suspected data" bit set should be accepted as good data.

FTP Service Commands

RETR - Must be implemented with a maximum pathname length of one hundred characters.

STOR - Must be implemented with a maximum pathname parameter of one hundred characters.

ABOR

C. Site Descriptor Table

1. Purpose

The Site Descriptor Table will be internal to the ELF software and will be used to furnish a large number of access control command parameters, transfer parameter command parameters, and site destination values. This table method is used primarily to avoid many interchanges between ELF and the CDC 6600 of usually redundant data, but also to allow flexibility when Phase 2 is implemented and many different values may occur.

2. Contents and Format

Since the Site Descriptor Table is internal to the ELF software, its format is of no concern to the CDC 6600. The ELF must be able to correlate a Site Descriptor Table ordinal to values needed for the initial connection to another site and parameters for the access control and transfer parameter commands. The values needed for each particular Site Descriptor Table ordinal are:

- site number
- host number
- socket number (parameter for SOCK command)
- user name (parameter for USER command)
- password (parameter for PASS command)
- byte size (parameter for BYTE command)
- representation type (parameter for TYPE command)
- file structure (parameter for STRU command)
- transfer mode (parameter for MODE command)

The Site Descriptor Table ordinal, when passed by the CDC 6600 as a command parameter will be a binary number.

3. Setting the Table Values

In order to avoid reassembly of the ELF software each time a new set of site descriptor parameters is needed, the ELF should have a utility that will accept from an ELF input device, especially tape and software developers may specify the format of the input data, but it must be broad enough to accept all acceptable parameter values as specified in the File Transfer Protocol. This must include the possibility of not specifying some values. The input data must be allowed to assign the Site Descriptor Table ordinal, which will be a number in the range of 1-999. The utility should accept commands that would reset the entire table or would change a particular ordinal.

D. FTP Status Table

1. Structure

The FTP Status Table consists of a contiguous block of 12 bit bytes divided into fixed length entries. The number of entries in this table should be an assembly parameter with a maximum value of 10. The original value will be 4. Each entry consists of two sections. The first section belongs to the CDC 6600 and its length is determined by an assembly parameter whose range is 0-5. The original value will be 3 bytes.

2. Section I Usage

The two bytes in the first section of an FTP status table entry are named the status byte and the auxiliary status byte. The ELF status byte is used whenever one byte of auxiliary information must be passed to the CDC 6600. If more information is to be passed to the CDC 6600 than can fit in the auxiliary status byte, then an auxiliary input block (Section III.E) must be used.

3. FTP Status Values

The following is a list of the numeric values the ELF can place in the status byte of an FTP status table entry. Each number is biased by the decimal number 128, which is obtained by using the processing class value of 2 for file transfer processing. The conditions under which the ELF should use a particular status are explained throughout the rest of Section III.

| Status | Numeric Value (decimal) |
|---|---|
| Available | 128 |
| New FTP Connection | 129 |
| Stor | 130 |
| Retr | 131 |
| Access Not Possible Now | 132 |
| Error in Some Parameter | 133 |
| Waiting for CTC | 134 |
| Clear to Output | 135 |
| Waiting for CTI | 136 |
| Clear to Input | 137 |
| Waiting for Data | 138 |

| | |
|---|---|
| Waiting for Xfer Ack | 139 |
| Waiting for FPC | 140 |
| Final Processing Completed | 141 |
| Transfer Complete | 142 |
| Connection Broken | 143 |

4. Section II Usage

The second section of the status table entry is strictly for the use and convenience of the CDC 6600. Upon request, the ELF must set this section of a particular entry to the values contained in the parameter section of the UPDATE STATUS ENTRY command.

5. Format of FTP Status Blocks

The ELF must deliver to the CDC 6600, upon request, either the entire FTP status table or a particular entry from the table. The form of the transfer block when one entry from the FTP table is being sent to the CDC 6600 is as follows:

Check Sum

9 Byte Count (decimal)

Line Number of the FTP Status Table Entry

642 Block Identifier (decimal)

Status Byte )

Auxiliary Status Byte ) Section I

3 bytes for CDC 6600 use ) Section II

When the entire FTP status table is sent to the CDC 6600 the format is as follows:

Check Sum

24 Byte Count (decimal)

0 Line Number

641 Block Identifier (decimal)

) 4 contiguous status table entries

) of 5 bytes each ordered by FTP

) line number

E. Auxiliary Input Block

1. Purpose

The auxiliary input block is used when the ELF must transfer more auxiliary information than can fit in the auxiliary status byte. The ELF is required to hold auxiliary input until the connection is broken on a particular line and deliver it on request to the CDC 6600. The CDC 6600 will only ask for the auxiliary input for one line at a time. How the ELF stores and references this information is of no concern to the CDC 6600. When delivered to the CDC 6600, ASCII character strings must be converted to display code using the procedure to convert from ASCII to display code described in Section II.H.3.

2. Structure

When the auxiliary input is delivered to the CDC 6600 it must be in the following form:

Check sum

Byte Count

Line Number of FTP Line

643 Block Identifier (decimal)

Auxiliary Input Type

Byte Count of actual auxiliary input

(

( Auxiliary Input (maximum of 98 bytes,

( see Section II.D.4.)

Auxiliary Input Type Codes

1 - Pathname from STOR command

2 - Pathname from RETR command

3 - Error Message from STOR command

4 - Error Message from RETR command

5 - Empty

6 - Other

F. Overall Control of FTP Processing

1. Starting FTP Processing

a. Description of Process

When the CDC 6600 wishes to begin FTP processing, it must issue the INITIATE FTP PROCESSING command. If FTP processing is already in progress, an Error Response Block should be returned to the CDC 6600. The ELF should do any internal checking it desires, but should include verifying that access to the network is possible, and that the Site Descriptor Table is installed. The ELF should then configure the FTP lines and buffers using the parameters included with the INITIATE FTP PROCESSING command and algorithm described in the next section. If FTP processing cannot be initiated, the ELF should respond with a NAK block. If the ELF can start FTP processing, the final values of the configuration parameters should be placed in the line 0 status table. The status byte of all configured FTP lines should be set to AVAILABLE status. The line 0 status table should be sent as the affirmative response to the CDC 6600 command.

b. Algorithm to FTP Buffers and Lines

This algorithm is used to determine the number of FTP lines to be used and the buffer size to be used for each line. It is not our intent that the FTP lines share buffers and be forced to interlace with each other. It is preferred to have fewer file transfers in progress simultaneously and to do each at maximum speed. Therefore, the ELF should have a dedicated buffer for each line.

A "buffer unit" is a fixed length block of 12-bit bytes. This length is an assembly parameter that will be set initially to 320 bytes. This corresponds to one disk sector on the CDC 6600.

There are 5 factors which effect the final sizes of the parameters:

L6 - Maximum number of lines that can be supported by the CDC 6600 (first parameter on the INITIATE FTP PROCESSING command).

Le - Maximum number of lines that can be supported by the ELF.

B6 - Maximum buffer size that can be supported by the CDC 6600, in buffer units (second parameter on the INITIATE FTP PROCESSING command).

Be - Maximum buffer size that can be supported by the ELF, in buffer units.

T - Total buffer space available in the ELF, in buffer units.

Let L = minimum L6, Le)

Let B = minimum (B6, Be)

If the product of L and B are less than, or equal to, T, then L and B are the configuration parameters. If this product exceeds T, then L and/or B must be lowered until the product is less than, or equal to T. However, do not lower L below 2; reduce B instead.

When the total buffer space required is less than or equal to T, these are the new configuration parameters. Place these values in the line 0 status block.

2. Control of FTP Lines

a. Assignment of FTP Lines

An FTP line is unassigned when the status byte contains AVAILABLE status. Only the ELF can assign an FTP line. When the Elf receives a request for an FTP connection from across the network it can select a line from the available lines and change the status from AVAILABLE to STOR or RETR. If the local CDC 6600 wishes to get an FTP line, it must issue an OPEN NEW FTP LINE command. The ELF should be the response to the CDC 6600 command. If no FTP line is available, the ELF should respond to the command with a NAK block.

b. Releasing of FTP Lines

Once an FTP line is assigned by the ELF, it cannot be assigned again until some command issued by the CDC 6600 releases that line. The five CDC 6600 commands that release a line are:

- FINAL PROCESSING COMPLETED

- BYE

- ERROR IN SOME PARAMETER

- ACCESS NOT POSSIBLE NOW

- ABORT FTP PROCESS

The ELF should set the status to AVAILABLE, and then do the processing dictated by the particular command.

c. Broken Connections

Should the logical connection be broken or an FTP ABOR command be received from the system with which the ELF is communicating, the ELF should not release the line but should place CONNECTION BROKEN status in the status byte. The CDC 6600 will then release the line with a BYE or ABORT FTP PROCESS command.

3. Dropping FTP Processing

a. Stop Accepting New FTP Connections

Prior to dropping the FTP processing, the CDC 6600 has the opportunity to cycle down by allowing existing FTP transfers to complete, but not allowing any new FTP connections. To accomplish this, the CDC 6600 can set bit 2(1) in the processing flags byte of the line 0 status table. This is done by using the CHANGE PROCESSING FLAGS command. While this bit is set, the ELF will not accept any new FTP connections from either another host or the local CDC 6600. This condition can be reversed by using the CHANGE PROCESSING FLAGS command to clear bit 2(1) in the processing flags byte.

b. Command to Drop FTP Processing

At any time the CDC 6600 can drop the FTP processing by issuing a STOP FTP PROCESSING command. An FTP ABOR command is to be issued to remote hosts for all active FTP lines. The FTP parameters in the line 0 status are cleared. The ELF should not accept any new FTP connections.

G. Process Descriptions

1. Processes Available

Phase I implementation will be limited to the basic process of transferring files using the FTP service commands STOR and RETR. Other FTP service commands may be implemented in Phase 2. Each command will require two distinct sections of the ELF software; a section for the sender of the command, and a section for the receiver of the command. They will use common procedures that will be described in the upcoming sections. The detailed desciption of the four process procedures is in Section III.G.4.

2. Format of the FTP Data Block

The process of the STOR and RETR commands is to transfer files across the network. The format of data blocks being transmitted across the network is dictated by the host-to-host protocol. However, the data blocks being transmitted between the ELF and the CDC 6600 must be in the following format convenient to the CDC 6600.

FTP Data Block:

| | | |
|---|---|---|
| | ( | Check Sum |
| Section 1 | ( | Byte Count of Total Block |
| | ( | FTP Line Number |
| | ( | Data Block Identifier |

Section 2 ( Data Block Descriptor

( Byte Count of Section III

Section 3 Data Being Transferred

The first section is 4 bytes long and is the general block header for all transfers. The line number will be a non-zero FTP line. The block identifier will be either 644 (decimal) for ELF to CDC 6600 transfer, or 141 (decimal) for CDC 6600 to ELF transfer.

The second section is the data header and consists of 2 bytes. The first byte is the data block descriptor and has only 3 acceptable values.

0 = data block

1 = end of data block is EOF

2 = end of data block is EOR

The second byte is a byte count of the data in the third section. This byte count plus six must equal the total byte count of the transfer block.

The third section is the data section. This section may be empty if the data block descriptor is non-zero.

3. Data Transfer Procedures

a. Procedure to Transfer FTP data from the CDC 6600 to the ELF

Data is transferred from the CDC 6600 to the ELF during the processing of the STOR command by the sender of the command, and during the processing of the RETR command by the receiver of the command. The blocks received from the CDC 6600 are in the format described in Format of an FTP Data Block (Section III.G.2). They must be reformated by the ELF according to the dictates of the host-to-host protocol. The data transfer begins after the STOR or RETR command has been received and the initial processing is completed. The ELF sets the CLEAR TO OUTPUT status in the status byte when it can accept a data block from the CDC 6600.

*When the CDC 6600 detects CLEAR TO OUTPUT status on a line and is ready to output a block, it issues the TRANSFER FTP DATA TO ELF command. A parameter contains the byte count of the data bytes. The ELF responds with that line's status table entry with the status byte containing WAITING FOR DATA status.

The CDC 6600 sends the data block as the next command block.

If the ELF receives the data correctly, it responds with the status table entry with the status byte containing WAITING FOR CTO. When the ELF has transmitted the data block to the remote host and is ready to accept another data block from the local CDC 6600, it sets CLEAR TO OUTPUT status in the status byte. If the ELF receives the data correctly and the data block descriptor contained the EOF value, then the ELF sets the status byte to WAITING FOR FPC status.

If the ELF does not receive the data correctly, it responds with an error response block and sets the status to CLEAR TO OUTPUT status. The data block that was just received is ignored.

If the developers of the ELF software feel that they do not need the TRANSFER FTP DATA TO ELF command to warn them of an impending data block transfer, they can omit the exchange described in the paragraph above marked with an *.

b. Procedure to Transfer FTP data from the ELF to the CDC 6600

Data is transferred from the ELF to the CDC 6600 during the processing of the STOR command by the receiver, and during the processing of the RETR command by the sender of the command. Data blocks received from a remote host must be reformatted before being passed to the local CDC 6600. The headings required by the host-to-host protocol and data representation type must be removed and a data block built according to the format of the FTP datablock (Section III.G.2).

The size of the data block passed to the CDC 6600 must be a whole number of buffer units, unless the Data Block Descriptor contains an EOR or EOF value. The definition of Buffer Unit is given in the Algorithm to Configure FTP Buffers (Section III.F.1.b). The ELF should always attempt to send as large a data block as possible to the CDC 6600. The maximum allowed block size is stored in the line 0 status block.

The data transfer begins after the STOR or RETR command has been received, and after the initial processing is completed. The ELF sets the CLEAR TO INPUT status in the status byte when it has a data block ready to send to the local CDC 6600. When the CDC 6600 detects CLEAR TO INPUT status and is ready to input the block, it issues the TRANSFER FTP DATA FROM ELF command. The ELF's response is the data block. The ELF changes the status byte to WAITING FOR XFER ACK status.

If the CDC 6600 receives the data correctly, it issues a GOOD FTP DATA TRANSFER command. The ELF discards the data block and responds with the status table entry with the status byte containing WAITING FOR CTI. The ELF can now accept more data from the remote system with which it is communicating. When the ELF has received another data block, reformatted it, and is ready to transmit it to the CDC 6600, it changes the status to CLEAR TO INPUT status.

If the CDC 6600 did not receive the data correctly, it will send a RESEND LAST RESPONSE command and the ELF should resend the data block. The CDC 6600 must eventually send a GOOD FTP DATA TRANSFER command or an ABORT FTP PROCESS command for this line. If an ABORT FTP PROCESS command is received, the ELF should set the status to AVAILABLE and send an FTP ABOR command to the remote system.

If the data block sent to the CDC 6600 has the EOF value (2) set in the Data Block Descriptor and if the CDC 6600 sends a GOOD FTP DATA TRANSFER command as the transfer acknowledgement, then the ELF will place the WAITING FOR FPC status in the status byte instead of WATING FOR CTI status.

4. ELF Processing Procedures

a. STOR Command

(1) Procedure used by ELF to a transfer file to another system when directed by a CDC 6600 SEND FILE TO ANOTHER SYSTEM command

When the ELF receives a SEND FILE TO ANOTHER SYSTEM command from the CDC 6600, it should respond immediately with the status table entry with the status byte containing WAITING FOR CTO status.

The ELF should then use the Site Descriptor Table ordinal parameter to obtain the parameters needed to make a connection to the desired host. The ELF should use the Initial Connection Protocol to contact the desired host. If the other host is not available, the ELF should change the status byte to ACCESS NOT POSSIBLE NOW and set the auxiliary status byte to 1 to indicate that the desired host is dead.

If the initial connection is successful, the ELF should issue the access control and transfer parameter commands, using the parameters from the Site Descriptor Table. If any of these commands receives a failure response, the ELF should set the status byte to ERROR IN SOME PARAMETER and log the error on the ELF console typewriter.

If no errors occurred, the ELF should then issue the FTP STOR command received from the CDC 6600. The ELF must convert the pathname to ASCII code using the procedures described in Section II.H.2. If the FTP reply, from the remote host, to the STOR command is other than a 250 code or a 451 code, the ELF should set the status byte to ERROR IN SOME PARAMETER status. If an error message accompanies the error response code, it should be converted to display code using procedures in Section II.H.3, and become auxiliary input for the FTP line. The auxiliary status byte should contain a byte count of this error message. If the FTP reply to the STOR command is a 451 code (File Access Denied to you), the ELF should set the status byte to ACCESS NOT POSSIBLE NOW status. The auxiliary status byte should be set to a 2. In both of these cases, the CDC 6600 should send a BYE command to close the line.

If the reply to the STOR command is a 250 code (FTP file transfer started correctly), the ELF should set the status byte to CLEAR TO OUTPUT status. The ELF will then receive data blocks from the local CDC 6600 using the Procedure for FTP Data Transfer from the CDC 6600 to the ELF (Section III.G.3.a). The ELF should send data blocks to the remote host using the host-to-host protocol over the data connection. If during the course of the data transfer, the connection is broken, or an error condition cannot be resolved between the ELF and the remote host, or an FTP ABOR command is received, the ELF should set CONNECTION BROKEN status in the status byte when it would normally be set to CLEAR TO OUTPUT.

When the ELF receives a data block from the CDC 6600 with the EOF value (2) in the Data Block Descriptor, the ELF should set WAITING FOR FPC in the status byte. The ELF sould then send this last data block to the remote host and wait fo a secondary reply to the STOR command.

When a successful reply code (252 or 257) is received from the other system, the ELF should set FINAL PROCESSING COMPLETED status in the status byte. The CDC 6600 will acknowledge the FINAL PROCESSING COMPLETED status with a BYE command and the ELF should set the status byte to AVAILABLE. The ELF can then clear connections with the remote host.

(2) Procedure Used by ELF to Receive a File from Another System after an FTP STOR Command has been Received From a Remote Host

Prior to receiving a STOR command, the ELF will have been in contact with the remote system during the Initial Connection Protocol and the access control and transfer parameter procedures. If acceptable transfer parameters or correct access control parameters have not been received, the ELF should reject the STOR command with an appropriate failure reply. If the initial processing was correct, but the ELF has no FTP line to the CDC 6600 available, the ELF should send a failure reply of 451 to the remote system.

If the initial processing was correct and an FTP line to the CDC 6600 is available, the ELF should place STOR status in the status byte of the available line. The pathname received with the STOR command should be converted to display code, using the procedures in Section II.H.3, and held as auxiliary input for this line. The auxiliary status byte should contain a byte count of the auxiliary input.

The CDC 6600 will return (not necessarily soon) with a command on the FTP line. If the command is ACCESS NOT POSSIBLE NOW, the ELF should set the status byte to AVAILABLE and send a 451 failure reply to the STOR command. If the CDC 6600 command is ERROR IN SOME PARAMETER, the ELF should set the status byte to AVAILABLE and send a 457 failure reply (transfer parameters in error) to the STOR command. An error message (in display code) may accompany the ERROR IN SOME PARAMETER command. If it does, it should be converted to ASCII, using procedures in Section II.H.2, and accompany the 457 response. The CDC 6600 considers the attempt to be ended and the ELF software should clear up connections according to established procedures.

If the ELF receives an INITIAL PROCESSING COMPLETED command from the CDC 6600, the ELF should set the status byte to WAITING FOR CTI and send a 250 (successful) reply to the STOR command. The ELF should now wait for a data block sent from the remote system. The ELF should receive this data from the other system using the host-to-host protocol. The data block should be reformatted and sent to the CDC 6600 according to the Procedure for Transferring FTP data from the ELF to the CDC 6600 (Section III.G.3.b). If during the course of the data transfers, the connection is broken, or an error condition cannot be resolved between the ELF and the remote host, or an FTP ABOR command is received, the ELF should set CONNECTION BROKEN status in the status byte when it would normally put CLEAR TO INPUT.

When an EOF block is received from the other system and the block has been passed to the CDC 6600, the ELF should set the status byte to TRANSFER COMPLETE. When the ELF receives a FINAl PROCESSING COMPLETED command from the CDC 6600, it should set the status byte to AVAILABLE and send a secondary reply of 257 to the remote host.

If the CDC 6600 is unable at any time to continue the process, it will issue the ABORT FTP PROCESS command. The CDC 6600 now considers the transfer complete. The ELF should send a 452 failure reply as a secondary reply to the FTP STOR command. The ELF and the other hosts should clean up the connections according to established protocols.

(3) Flowchart: Sending a File from one AFSC CDC 6600 site (6600-A) to another AFSC CDC 6600 site (6600-B)

The following flowchart combines the information in the two previous sections. Those sections described the actions required of the sending ELF and the actions required of the receiving ELF. This flowchart is meant to show the relations between the four involved systems when transferring a file between two AFSC CDC 6600 sites.

The events that take place during this transfer are usually sequential. However, during the transfer of data, simultaneous operations can occur. For example, as shown in the flowchart at the triangle symbol just before connection 9, the path goes in two directions. This is immediately after the local ELF (ELF-A) has sent a data block to the remote ELF (ELF-B). ELF-A is engaged in obtaining the next data block from 6600-A, while ELF-B is involved in sending this data block to 6600-B. This is not a standard flowcharting technique, and some license has been taken to illustrate the point.

b. RETR Command

(1) Procedure used by ELF to Retrieve File from Another System when directed by a CDC 6600 RETRIEVE FILE FROM ANOTHER SYSTEM Command

When the ELF receives a RETRIEVE FILE FROM ANOTHER SYSTEM command from the CDC 6600, it should respond with the status byte set to WAITING FOR CTI status.

The ELF should then use the Site Descriptor Table ordinal parameter to obtain the parameters needed to make a connection to the desired host. The ELF should use the Initial Connection

Protocol to contact the desired host. If the other host is not available, the ELF should change the status byte to ACCESS NOT POSSIBLE NOW and set the auxiliary status byte to 1 to indicate that the desired host is dead.

If the initial connection is successful, the ELF should issue the access control and transfer parameter commands, using the parameters from the Site Descriptor Table. If any of these commands receives a failure replay, the ELF should set the status byte to ERROR IN SOME PARAMETER, and log the error on the ELF console typewriter.

If no errors occurred, the ELF should then issue the FTP RETR command using the pathname parameter from the command received from the CDC 6600. The ELF must convert the pathname to ASCII code, using the procedures in Section II.H.2. If the FTP reply from the remote host to the RETR command is other than a 250 code or 451 code, the ELF should set the status byte to ERROR IN SOME PARAMETER status. If an error message accompanies the error response code, it should be converted to display code using the procedure in Section II.H.3, and become auxiliary input for the FTP line. The auxiliary status byte should contain a byte count for this error message. If the reply to the RETR command is a 451 code (File Access Denied to you), the ELF should set the status byte to ACCESS NOT POSSIBLE NOW status. The auxiliary status byte should be set to 2. In both of these cases, the CDC 6600 should send a BYE command to close the line.

If the reply to the RETR command is a 250 code (FTP: File Transfer Started Correctly), the ELF should wait until it receives a data block from the remote system. The ELF receives data from the remote system using the host-to-host protocol. The data blocks should be reformatted by the ELF and the status byte should be changed to CLEAR TO INPUT status. Data is sent to the CDC 6600 as described in Procedure for Transferring FTP Data from the ELF to the CDC 6600 (Section III.G.3.b).

If during the course of the data transfers, the connection is broken, or an error condition cannot be resolved between the ELF and the remote host, or an FTP ABOR command is received, the ELF should set CONNECTIONS BROKEN status in the status byte when it would normally put CLEAR TO INPUT.

When the ELF receives a data block from the other system with an EOF value (2) in the Data Block Descriptor, it should wait until the secondary reply (252 or 257 code) is received. The ELF should then pass the last data block to the CDC 6600 and acknowledge the

CDC 6600 GOOD FTP DATA TRANSFER command by setting TRANSFER COMPLETE status.

The CDC 6600 will send a FINAL PROCESSING COMPLETE command to close the line. The ELF should set the status to AVAILABLE. The ELF should then clear all connections with the other system according to established protocols.

(2) Procedure used by ELF to send file to another system after an FTP RETR command has been received from a remote host

Prior to receiving a RETR command, the ELF will have been in contact with the remote system during the Initial Connection Protocol and the access control and transfer parameter procedures. If acceptable transfer parameters have not been received or correct access control parameters have not been received, the ELF should reject the RETR command with an appropriate failure reply. If the initial processing was correct, but the ELF has no FTP line to the CDC 6600 available, the ELF should send a failure reply of 451 to the remote system.

If the initial processing was correct and an FTP line to the CDC 6600 is available, the ELF should place RETR status in the status byte of the available line. The pathname received with the RETR command should be converted to display code using the procedure in Section II.H.3, and held as auxiliary input for this line. The auxiliary status byte should contain a byte count of the auxiliary input.

The CDDC 6600 will return (not necessarily soon) with a command on the FTP line. If the command is ACCESS NOT POSSIBLE NOW, the ELF should set the status byte to AVAILABLE and send a 451 failure reply to the RETR command. If the CDC 6600 command is ERROR IN SOME PARAMETER, the ELF should set the status byte to AVAILABLE and send a 457 failure reply to the RETR command. An error message may accompany the ERROR IN SOME PARAMETER command. If it does, it should be converted to ASCII code using the procedure in Section II.H.2, and accompany the 457 reply to the RETR command. The CDC 6600 considers the attempt to be ended and the ELF software should clean up connections according to established protocols.

If the ELF receives an INITIAL PROCESSING COMPLETE command from the CDC 6600, the ELF should set the status byte to WAITING FOR CTO. A successful reply (250) should then be sent to the RETR command. The ELF should now set the status byte to CLEAR TO INPUT.

The ELF will now receive data blocks from the CDC 6600 using the Procedure for FTP Data Transfer from the CDC 6600 to the ELF (Section III.G.3.a). The ELF must reformat the data blocks and send them to the other host using the host-to-host protocol. If during the course of the data transfers, the connection is broken, or an error condition cannot be resolved between the ELF and the remote host, or an FTP ABOR command is received, the ELF should set CONNECTION BROKEN status in the status byte when it would normally put CLEAR TO INPUT status.

When the ELF receives a data block from the CDC 6600 with the EOF value (2) in the Data Block Descriptor, the ELF should set WAITING FOR FPC in the status byte. The ELF should then send this last block to the remote host. After a short delay, the ELF should send a successful secondary reply code of 257 (closing the data connection, transfer completed) to the remote host. The ELF should set the status byte to FINAL PROCESSING COMPLETED status. The CDC 6600 will then issue a BYE command to close the line and the ELF should set the status byte to AVAILABLE.

(3) Flowchart: An AFSC CDC 6600 site (6600-A) retrieving a file from another AFSC CDC 6600 site (6600-B)

The following flowchart combines the information in the two previous sections. These sections described the actions required of the sending ELF and the actions required of the receiving ELF. This flowchart is meant to show the relations between the four involved systems when transferring a file between two AFSC CDC 6600 sites.

The events that take place during this transfer are usually sequential. However, during the transfer of data, simultaneous operations can occur. This is a similar situation to the flowchart in Section III.G.4.a(3) and occurs at flowchart symbol . The flowchart symbols are the same as used in the previous flowchart.

H. List of FTP-Oriented Commands Issued by the CDC 6600

The following pages describe the FTP-oriented commands and the conditions under which these commands can be issued by the CDC 6600. Each command is in the format of a standard transfer block (Section 11.D.2). The descriptions contain the specific details about what line numbers are valid, what the parameters are and how many there should be, under what status conditions the command is valid, and the processing requirement of the ELF. The last section is, "Status Changes and ELF Responses to Command." Status changes only refer to section one of the FTP line status table entries, the section that can be changed by the ELF. "Responses" are the types of information the ELF is allowed to send to the CDC 6600 after receiving this command. Some response must be made to every command the CDC 6600 issues.

COMMAND: Initialize FTP Processing

BLOCK IDENTIFIER: 129 (decimal)

LINE TYPE: 0

PARAMETERS: Parameter 1: Maximum number of FTP lines.
Parameter 2: Maximum FTP Buffer Size (in buffer units).

PURPOSE OF COMMAND: The CDC 6600 wishes to begin file transfer processing.

WHEN CAN COMMAND BE ISSUED: When FTP processing is not active.

ELF PROCESSING REQUIREMENT: The ELF must verify that network accessing is possible and that a Site Descriptor Table is installed. It must configure the number of FTP lines and the size of their buffers and place these values in the line 0 status table. See Section III.F.1.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If FTP processing is started, set the status byte of all configured FPT lines to AVAILABLE status. Response is the line 0 status table.

2. If FTP processing cannot be started, there are no status changes, and the response is a NAK block.

COMMAND: Open New FTP Line

BLOCK IDENTIFIER: 130 (decimal)

LINE TYPE: 0

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 wishes to initiate a file transfer and is requesting that the ELF assign it an FTP line.

WHEN CAN COMMAND BE ISSUED: When FTP processing is active and bit 2(1) in the line 0 processing flags byte is zero.

ELF PROCESSING REQUIREMENT: Select an available FTP line and assign it to the CDC 6600. See Section III.F.2.a.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If an FTP line is assigned, set the status byte for the assigned line to NEW FTP CONNECTION status. Response is the FTP line status table entry for the assigned line.

2. If no FTP line is assigned, the ELF changes no status and responds with a NAK block.

COMMAND: Stop FTP Processing

BLOCK IDENTIFIER: 132 (decimal)

LINE TYPE: 0

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 wants to stop FTP processing now.

WHEN CAN COMMAND BE ISSUED: When FTP processing is active.

ELF PROCESSING REQUIREMENT: An FTP ABOR command is to be issued to remote hosts for all active FTP lines. All connections are broken with remote systems and no new connections are accepted. The FTP configuration parameters in the line 0 status table are cleared. See Section III.F.3.b.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: All status bytes are set to AVAILABLE status and the response is an ACK block.

COMMAND: Send file to another system.

BLOCK IDENTIFIER: 133 (decimal)

LINE TYPE: FTP

PARAMETERS: Parameter 1: Site Descriptor Table Ordinal
Parameter 2: Pathname Length (in bytes)
Parameter 3: Pathname (50 byte maximum)

PURPOSE OF COMMAND: The CDC 6600 wants to send a file to another system and wants the ELF to make the initial connection.

ELF PROCESSING REQUIREMENT: The ELF should set the status byte to WAITING FOR CTO status and respond with the FTP status table entry. The ELF should attempt to make connection to the desired host and send the FTP STOR command. The ELF should analyze the reply to the STOR command and tell the CDC 6600 whether to begin the transfer, delay the transfer, or that an error occurred. The details of the ELF processing requirement are given in the first five paragraphs of Section III.G.4.a.(1).

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

The status byte is changed to WAITING FOR CTO status and the response is the FTP status table entry.

The status byte for this line will later be changed to one of the following status values.

- CLEAR TO OUTPUT

- ACCESS NOT POSSIBLE NOW

- ERROR IN SOME PARAMETER

COMMAND: Retrieve file from another system

BLOCK IDENTIFIER: 134 (decimal)

LINE TYPE: FTP

PARAMETERS: Parameter 1: Site Descriptor Table Ordinal
Parameter 2: Parameter Length (in bytes)
Parameter 3: Pathname (50 byte maximum)

PURPOSE OF COMMAND: The CDC 6600 has detected a request to retrieve a file from another system. This command asks the ELF to make the initial processing.

WHEN CAN COMMAND BE ISSUED: When status byte contains NEW FTP CONNECTION status.

ELF PROCESSING REQUIREMENT: The ELF should set the status byte to WAITING FOR CTI status and respond with the FTP status table entry. The ELF should attempt to make connection to the desired host and send the FTP RETR command. The ELF should analyze the reply to RETR command and tell the CDC 6600 whether the transfer has begun, to try again later, or that an error has occurred. The details of the ELF processing requirement are given in the first five paragraphs of Section III.G.4.b.(1).

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

The status byte is changed to WAITING FOR CTI status and the response is the FTP status table entry.

The status byte for this line will later be changed to one of the following status values.

- CLEAR TO INPUT

- ACCESS NOT POSSIBLE NOW

- ERROR IN SOME PARAMETER

COMMAND: Input Auxiliary Data

BLOCK IDENTIFIER: 135 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 wishes to obtain information that the ELF has stored for him an auxiliary input. Presently, only pathnames and error messages are valid.

WHEN CAN COMMAND BE ISSUED: When FTP processing is active.

ELF PROCESSING REQUIREMENT: The ELF formats the data into an auxiliary input block. If no auxiliary information is being held for this line, an empty block is created.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: There are no status changes. The response is an auxiliary input block.

COMMAND: Access not possible now

BLOCK IDENTIFIER: 136 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 is telling the ELF that the file system request that was made for this line cannot be done at this time. There is not an error.

WHEN CAN COMMAND BE ISSUED: When the status byte contains STOR or RETR status.

ELF PROCESSING REQUIREMENT: The ELF should send a 451 failure reply to the FTP STOR or RETR command received from the remote host. This line should be made available again. See Section III.G.4.a.(2) and Section III.G.4.b.(2) for details.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The status byte should be changed to AVAILABLE status. The response is an ACK block.

COMMAND: Access not possible now

BLOCK IDENTIFIER: 136 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 is telling the ELF that the file system request that was made for this line cannot be done at this time. There is not an error.

WHEN CAN COMMAND BE ISSUED: When the status byte contains STOR or RETR status.

ELF PROCESSING REQUIREMENT: The ELF should send a 451 failure reply to the FTP STOR or RETR command received from the remote host. This line should be made available again. See Section III.G.4.a.(2) and Section III.G.4.b.(2) for details.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The status byte should be changed to AVAILABLE status. The response is an ACK block.

COMMAND: Error in some parameter

BLOCK IDENTIFIER: 137 (decimal)

LINE TYPE: FTP

PARAMETERS: Parameter 1: length, in bytes, of an error message. If zero, parameter two doesn't exist.
Parameter 2. Optional, display coded error message.

PURPOSE OF COMMAND: The CDC 6600 is telling the ELF that the file system request that was made for this line contained an error in the pathname.

WHEN CAN COMMAND BE ISSUED: When the status byte contains STOR or RETR status.

ELF PROCESSING REQUIREMENT: The ELF should send a 457 failure reply to the FTP STOR or RETR command received from the remote host. This line should be made available again. See Sections III.G.4.a.(2) and III.G.4.b.(2) for details.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The status byte should be changed to AVAILABLE status. The response is an ACK block.

COMMAND: Initial processing completed

BLOCK IDENTIFIER: 138 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 is telling the ELF that the file system request that was made on this line has been accepted and that the data transfer should begin.

WHEN CAN COMMAND BE ISSUED: When status byte contains STOR or RETR status.

ELF PROCESSING REQUIREMENT: The ELF should send a 250 successful reply to the FTP STOR or RETR command received from the remote host. The ELF should then allow the data transfer to begin. The details of the ELF processing are in Sections III.G.4.a.(2) and III.G.4.b.(2).

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If the status byte contained STOR, change the status to WAITING FOR CTI.

2. If the status byte contained RETR, change the status to WAITING FOR CTO. The response is the status table entry.

COMMAND: Transfer FTP data to ELF

BLOCK IDENTIFIER: 139 (decimal)

LINE TYPE: FTP

PARAMETERS: Parameter 1: Byte count of data bytes to be sent. This does not include the 4 byte block header or 2 byte data header.

PURPOSE OF COMMAND: This command gives warning to the ELF that the next block to be issued is an FTP data block for this line and gives the length of that block.

WHEN CAN COMMAND BE ISSUED: When the status byte contains CLEAR TO OUTPUT status.

ELF PROCESSING REQUIREMENT: Prepare to receive an FTP data block. See Section III.G.3.a.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If the ELF can accept the data now, it will change the status byte to WAITING FOR DATA status. The response is the status table entry.

2. If the ELF cannot accept the data block now, it will not change any status and the response will be a NAK block.

COMMAND: Transfer FTP data from ELF

BLOCK IDENTIFIER: 140 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 wishes to receive an FTP data block being held in the ELF.

WHEN CAN COMMAND BE ISSUED: when status byte contains CLEAR TO INPUT status.

ELF PROCESSING REQUIREMENT: See section III.G.3.b.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If the ELF can send the data block now, it will change the status to WAITING FOR XFER ACK status. The response will be the FTP data block.

2. If the ELF cannot send the data block now, it will not change any status, and the response will be a NAK block.

COMMAND: FTP Data Block

BLOCK IDENTIFIER: 141 (decimal)

LINE TYPE: FTP

PARAMETERS: A minimum of two parameter bytes. See Section III.G.2 for the format of this block.

PURPOSE OF COMMAND: This is not a command, but a block of data that the CDC 6600 wants the ELF to send to the remote host.

WHEN CAN COMMAND BE ISSUED: when status byte contains WAITING FOR DATA status.

ELF PROCESSING REQUIREMENT: The ELF should reformat the data block and send it to the remote host. See Section III.G.3.a for details of the interaction with the CDC 6600. If the data block descriptor has the EOF bit set, also see Section III.G.4.a.(1) and III.G.4.b.(2) for end of file processing procedures.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If the ELF received the data block correctly from the CDC 6600, and the data block descriptor was:

(a) Zero or EOR, then set the status to WAITING FOR CTO status; or,

(b) EOF, then set the status to WAITING FOR FPC status.

Response is the status table entry.

2. If the ELF received the data block incorrectly from the CDC 6600, set the status to CLEAR TO OUTPUT. Respond with an Error Response Block.

COMMAND: Good FTP Data Transfer

BLOCK IDENTIFIER: 142 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: This tells the ELF that it can discard the last data block sent to the CDC 6600 for this line.

WHEN CAN COMMAND BE ISSUED: when status byte contains WAITING FOR XFER ACK status.

ELF PROCESSING REQUIREMENT: Discard the last FTP data block sent to the CDC 6600 for this line. Accept more data from the remote system. See section III.G.3.b for details of the interaction with the CDC 6600. If the data block descriptor has the EOF bit set, also see sections III.G.4.a.(2) and III.G.4.b.(1) for end of file processing procedures.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If the data block descriptor contained zero or EOR, then set the status to WAITING FOR CTI status.

2. If the data block descriptor contained EOF, then set the status to WAITING FOR FPC status.

The response is the status table entry.

COMMAND: Final processing Completed

BLOCK IDENTIFIER: 143 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: At the end of a data transfer of a file, the file receiver tells the ELF that the file has been entered into the CDC 6600 file system.

WHEN CAN COMMAND BE ISSUED: When the status byte contains TRANSFER COMPLETE status.

ELF PROCESSING REQUIREMENT: The ELF processing requirement differs depending on if this is the end of a STOR sequence or a RETR sequence. See the last two paragraphs from Sections III.G.4.a.(2) and III.G.4.b.(1) for procedures to follow.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The status byte is changed to AVAILABLE status. The response is an ACK block.

COMMAND: Bye

BLOCK IDENTIFIER: 144 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 is closing the connection after sending an entire file that has been sent to the remote system and acknowledgement has been returned from the remote system.

WHEN CAN COMMAND BE ISSUED: When the status byte contains FINAL PROCESSING COMPLETE status.

ELF PROCESSING REQUIREMENT: The ELF should close all connections to the remote system for this line. This line should be made available.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The status byte should be changed to AVAILABLE status. The response is an ACK block.

COMMAND: Abort FTP Process

BLOCK IDENTIFIER: 145 (decimal)

LINE TYPE: FTP

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 cannot continue the process and must abort it.

WHEN CAN COMMAND BE ISSUED: Anytime the status byte does not contain AVAILABLE status.

ELF PROCESSING REQUIREMENT: An FTP ABOR command or an appropriate failure reply to an FTP STOR or RETR command is sent to the remote host. Clear all processing on this line.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: Set the status byte to AVAILABLE status. Send an ACK block as response.

## IV. Interactive Processing

### A. Objectives for Phase I Implementation

Our primary objective during Phase 1, as far as interactive processing is concerned, is to allow a terminal user connected to the ARPA network to access the timesharing system of the CDC 6600. During Phase 1, only a low speed asynchronous protocol is required.

There are two well-known low speed protocols 'teletype compatible' and 'IBM 2741 compatible'. A given timesharing system may accept either or both of these protocols. It is not our purpose to expand the capability of our current timesharing system. The CDC INTERCOM system does not currently support IBM 2741 terminals and will not support them over the ARPA network.

It is essential that our Phase 1 Interactive Protocol be implemented in a manner that will not interfere with our Phase 2 objectives. One Phase 2 goal that must be considered in the design of Phase 1 is the support of a negotiated TELNET option, for use between AFSCNET sites, to deliver lines of interactive data as one packet per line, rather than the character by character method currently used.

We have designed the CDC 6600 - ELF interface around this line by line approach to transferring data. This method has the additonal advantage of significantly reducing the number of interchanges between the CDC 6600 and the ELF.

### B. Protocols Required

The protocols required for interactive processing are listed in Section II. This specification requires no negotiated options added to the TELNET protocol. The standard representations and meanings of TELNET control functions are to be implemented. The ELF software will handle internally all of these control functions except the IP (Interrupt Process) and AO (Abort Output) control functions. On these, the ELF will set flag bits for the CDC 6600 and in addition, discard all queued data for that logical connection. The ELF will also implement the TELNET 'synch' signal - datamark convention so ASCII values can be sent to the CDC 6600. The ELF will send only the required ASCII control codes to the Network Virtual Terminal (NVT) printer: Null, Line Feed, and Carriage Return.

### C. Interactive Status Table

1. Structure

The Interactive Status Table consists of a continuous block of 12 bit bytes divided into fixed length entries. The number of entries in this table should be an assembly parameter with a maximum of 64. The original value should be 16. Each entry consists of two sections. The first section belongs to the CDC 6600 and its length is an assembly parameter whose range is 0-5. The original value should be 2 bytes.

2. Section One Usage

The two bytes in section one of the status table entry are named the status byte and the auxiliary status byte. The status byte is used by the ELF to control the sequence of events during ELF/CDC 6600 communications. The following is a list of the numeric values the ELF can place in the status byte of an interactive status table entry. Each number is biased by the decimal number 64, which is obtained by using the processing class value of 1 for interactive processing.

| Status | Numeric Value (Decimal) |
|---|---|
| Available | 64 |
| New Interactive Connection | 65 |
| Normal | 66 |
| Waiting for Xfer Ack | 67 |
| Waiting for Data | 68 |

The auxiliary status byte is used by the ELF to inform the CDC 6600 of the choice of actions it may take on a given logical connection. The format of the 12 bit auxiliary status byte is as follows:

Bit 0: CLEAR TO OUTPUT - if set, the CDC 6600 may send an interactive data block for this line.

Bit 1: CLEAR TO INPUT - if set, the CDC 6600 may input an interactive data block for this line.

Bit 2: TERMINAL READY - if set, the logical connection between the terminal and the ELF is intact.

Bit 3: ABORT OUTPUT INTERRUPT - if set, the TELNET control function AO has been received.

Bit 4: PROCESS INTERRUPT - if set, the TELNET control function IP has been received.

Bit 5: Unused

3. Section Two Usage

Section Two of the status table entry is strictly for the use and convenience of the CDC 6600. Upon request, the ELF must set this section of a particular entry to the values contained in the parameter section of the UPDATE STATUS ENTRY command.

4. Format of Interactive Status Block

The ELF must deliver to the CDC 6600, upon request, either the Interactive Status Table or a particular entry from the table. The form of the transfer block when one entry from the interactive status table is being sent to the CDC 6600 is as follows:

Check Sum

8 Byte Count (decimal)

Line Number of the Interactive Status Table Entry

578 Block Identifier (decimal)

Status Byte )

Auxiliary Status Byte ) Section 1

2 bytes )

For CDC 6600 use ) Section 2

When the entire interactive status table is sent to the CDC 6600 the form is as follows:

Check Sum

52 Byte Count (Decimal)

0 Line Number

577 Block Identifier (decimal)

( 16 contiguous status table entries, of 4 bytes

( each, ordered by interactive line number

D. Overall Control of Interactive Processing

1. Starting Interactive Processing

a. Description of Process

When the CDC 6600 wishes to begin interactive processing, it must issue an INITIATE INTERACTIVE PROCESSING command. If interactive processing is already in progress, an Error Response Block should be returned to the CDC 6600. The ELF should do any internal checking it desires, but should include verifying that access to the network is possible. The ELF should then configure the interactive lines and buffers using the parameters included with the INITIATE INTERACTIVE PROCESSING command and the algorithm described in the next section. If interactive processing cannot be initiated the ELF should respond with a NAK block. If the ELF can start interactive processing, the final values of the configuration parameters should be placed in the line 0 status block. The status byte of all configured interactive lines should be set to AVAILABLE status. The line 0 status block should be sent as the response to the CDC 6600 command.

b. Algorithm to Configure Interactive Buffers and Lines

This algorithm is used to determine the number of interactive lines to be used and the buffer size to be used for each line. The number of lines to be used during this interactive session is the minimum of:

The maximum number of interactive lines the ELF supports (an assembly parameter in the range 1 - 64 whose initial value is 16), and

The maximum number of lines the CDC 6600 wishes to use (first parameter on the INITIATE INTERACTIVE PROCESSING command).

The size of the buffer, in bytes, used for interactive data blocks during this session is the minimum of:

The maximum buffer size supported by the ELF (an assembly parameter in the range 20 - 150 whose initial value is 40), and

The maximum buffer size the CDC 6600 wishes to use (second parameter on the INITIATE INTERACTIVE PROCESSING command).

Should the total buffer space not be sufficient, the ELF should lower the number of lines to be used. Place the final values in the line 0 status block.

2. Control of Interactive Lines

a. Assignment of Interactive Lines

An interactive line is unassigned when the status byte contains AVAILABLE status. The CDC 6600 never has need to ask for an interactive line. When a terminal makes a connection to the ELF via the ARPA network, the ELF should choose an unassigned line and change its status to NEW INTERACTIVE CONNECTION.

b. Releasing of Interactive Lines

Once an interactive line is assigned by the ELF, it cannot be assigned again until a BYE command issued by the CDC 6600 releases that line. Upon receiving a BYE command, the ELF should set the status to AVAILABLE and respond with an ACK block. The ELF should close the logical connection to the terminal on that line.

c. Broken Connections

Should the logical connection be broken, the ELF should not release the line, but should clear the TERMINAL READY bit and, if necessary the CLEAR TO OUTPUT bit in the auxiliary status byte. The CDC 6600 will release the line sometime after detecting the TERMINAL READY bit cleared.

3. Dropping Interactive Processing

When the CDC 6600 is ready to stop the interactive processing session, it issues a STOP INTERACTIVE PROCESSING command on line 0. The ELF should terminate all connections to terminals and accept no more connections. All queued data should be discarded and the interactive parameters in the line 0 status block should be cleared. The ELF should respond with an ACK block.

E. Process Description

1. Processes Available

Phase I implementation is limited to the basic objective of allowing a terminal user who has access to the ARPA network to have access to the CDC 6600 timesharing system. The description found in the following sections is what the ELF must do during an interactive processing session to allow this access.

2. Format of Interactive Data Block

All information received from the terminal via the ARPA network that is not preceded by the TELNET control character IAC (Interpret As Command) is considered to be data that is to be passed to the CDC 6600. Data is transmitted between the ELF and the terminal as a string of ASCII characters as dictated by the TELNET protocol. However, the data blocks being transmitted between the ELF and the CDC 6600 must be in the following format convenient to the CDC 6600.

Blocks of interactive data being transmitted between the ELF and the CDC 6600 have 3 sections.

The first section is 4 bytes long and is the standard block header for all transfer blocks. The line number will be a non-zero interactive line. The block identifier will be either 578 (decimal) for ELF to CDC 6600 transfer or 71 (decimal) for CDC 6600 to ELF transfer.

The second section is the data header and consists of two bytes. The first byte is the data block descriptor and has the following format:

bit 0: CONVERSION If set, this block contains display coded characters; if clear, this block contains ASCII characters and is not to be converted or has not been converted between ASCII and display code.

bit 1: END OF LINE If set, this is the last portion of a line.

bit 2: START OF LINE If set, this is the first portion of a line

bits 3-11: Unused

For example, a display code line that fits entirely in one interactive data block would have all 3 bits set in the data block descriptor. The second byte is a byte count of section 3. This byte count plus six must equal the total byte count of the transfer block.

The third section is the data section. This section may not be empty, but it may not be longer than the maximum buffer size shown in the line 0 status block. If the data is in display code, as shown by bit 0 of data block descriptor being set, it will be packed with two characters per byte. The first character to be received or transmitted being the left character. If the block contains ASCII characters, they are packed into bits 0 - 6 of the byte and bits 7 - 11 are unused. Bit 0 is the least signficicant bit.

3. Data Transfer Procedures

a. Procedure to transfer interactive data from the CDC 6600 by the ELF

(1) Code conversion

Output lines can be passed from the CDC 6600 timesharing system to the ELF to be passed to the terminal during the time the logical line is assigned to that terminal. The format of the output lines received from the CDC 6600 are in the format described in the previous section.

If the CONVERSION bit is one in the data block descriptor, then the data is in display code and the ELF must convert it to ASCII code according to the Procedure for Converting Display Code to ASCII (Section II.H.2). A "new line sequence" is defined by an ASCII carriage return code, followed by an ASCII line feed code, followed by an appropriate number of ASCII null codes. If the START OF LINE bit is one in the data block descriptor, the ELF should send a new line sequence prior to sending the ASCII characters resulting from the conversion of the display code characters. If the END OF LINE bit is one in the data block descriptor, the ELF should send a new line sequence after sending the ASCII characters resulting from the conversion of the display code.

If the CONVERSION bit is zero in the data block descriptor, the ELF should transmit the ASCII characters in the interactive data block just as they are. The START OF LINE and END OF LINE bits have no meaning if the CONVERSION bit is zero.

(2) Transfer Procedure

The CDC 6600 can pass an interactive data block to the ELF when the status byte contains NORMAL status and the CLEAR TO OUTPUT bit is set in the auxiliary status byte.

*When the CDC 6600 is ready to output an interactive data block and the above conditions are satisfied, it issues a TRANSFER INTERACTIVE DATA TO ELF command. A parameter contains the byte count of the data bytes in Section 3 of the interactive data block. Upon receiving the command the ELF changes the status byte to WAITING FOR DATA status and sends the status table entry as a response.

The CDC 6600 sends the data block as the next command block.

If the ELF receives the data correctly, it should set the status byte to NORMAL status, clear the CLEAR TO OUTPUT bit in the auxiliary status byte, and send the status table entry as a response to the data block. The ELF should reformat the data using the procedures in the previous section and send the output data to the terminal. When the ELF has transmitted the data to the terminal and can accept another output block from the CDC 6600 it should set the CLEAR TO OUTPUT bit in the auxiliary status byte.

If the data transfer was incorrect, the ELF should set the status byte to NORMAL status and respond with an Error Response Block. The ELF should not clear the CLEAR TO OUTPUT bit in the auxiliary status byte and should ignore the data just received from the CDC 6600.

If the developers of the ELF software feel they do not need the TRANSFER INTERACTIVE DATA TO ELF command to warn them of an impending data block transfer, they can omit the exchange in the paragraph above marked with an *.

b. Procedure to Transfer Interactive Data from the ELF to the CDC 6600

(1) Code Conversion

A normal interactive data line received by the ELF from a terminal is a sequential set of ASCII characters that is terminated by a carriage return character. The ELF should convert these ASCII input characters using the Procedure for Converting ASCII to Display Code (Section II.H.3). One or more Interactive Data Blocks should be built according to directions in Section IV.E.2. The CONVERSION bit must be set in these blocks.

Another type of input data may reach the ELF enveloped within a TELNET "synch" signal and a TELNET datamark signal. This data is to be passed to the CDC 6600 unconverted. It must be built into one or more interactive data blocks with the CONVERSION bit set to zero.

(2) Transfer Procedure

The ELF receives input characters from the terminal and buffers them internally until an entire line has been received; that is, until a carriage return or datamark has been received. When the ELF has received an entire input line, it formats it into an interactive data block. It then sets the CLEAR TO INPUT bit in the auxiliary status byte to indicate to the CDC 6600 that an input block is ready.

When the CDC 6600 is ready to input a line and the CLEAR TO INPUT bit is set and the status byte contains NORMAL status, it should issue a TRANSFER INTERACTIVE DATA FROM ELF command. Upon receiving the command, the ELF should change the status byte to WAITING FOR XFER ACK status and respond to the command with an Interactive Data Block.

If the CDC 6600 receives the block correctly, it will send a GOOD INTERACTIVE DATA TRANSFER command. The ELF should discard the data just sent to the CDC 6600 and change the status byte to NORMAL status. The ELF should also clear the CLEAR TO INPUT bit until it has another interactive data block ready to send to the CDC 6600. The ELF should send the status table entry as a response to the command.

Description of an Interactive Processing Session

a. Initial Connection

An interactive session with a terminal is started when the ELF receives the first indication that a terminal is trying to connect to an interactive socket. When the initial connection is done, the ELF selects an unassigned interactive line and changes the status from AVAILABLE to NEW INTERACTIVE

CONNECTION. When the CDC 6600 detects this new connection, it will issue a START INTERACTIVE PROCESSING command. The ELF will change the status to NORMAL and set the TERMINAL READY and CLEAR TO OUTPUT bits in the auxiliary status byte. The ELF will send the status table entry as a response to the command. After this initial sequence to begin an interactive session, the actions of the ELF depend upon how the terminal user is using the CDC 6600 timesharing system. From now until the connection is broken, the ELF will use the auxiliary status byte to inform the CDC 6600 of what can be done, and the CDC 6600 will choose the action to be taken.

b. Receiving Input Data From a Terminal

When the ELF receives a normal input line or an input line bracketed between TELNET "synch" and TELNET datamark signals, it sets the CLEAR TO INPUT bit in the auxiliary status byte. Note that the bit is not set until an entire line has been received; that is, the carriage return code or datamark signals has been received. On normal input lines, only codes that convert to some display code are sent to the CDC 6600. Therefore, if a carriage return is received and no display code characters are present after the conversion, ignore the input line and do not set the CLEAR TO INPUT bit. The CDC 6600 will choose when it wishes to input the waiting data. The procedure to transfer interactive data from the ELF to the CDC 6600 (Section IX.E.3.b) will then be used.

c. Sending Output Data to a Terminal

When the ELF can accept an interactive data block from the CDC 6600 it will set the CLEAR TO OUTPUT bit. When the CDC 6600 is ready to send output for that terminal it will follow the procedure to transfer interactive data from the CDC 6600 to the ELF (Section IV.E.3.a). The ELF will then reformat and, if necessary, convert the data codes, and send the data to the terminal in the most efficient manner it can. When it has sent the data, it will set the CLEAR TO OUTPUT bit again.

d. Broken Connections

Should the connection to the terminal be broken, the ELF should clear the TERMINAL READY bit in the auxiliary status byte. The ELF should not attempt to send anymore data to that terminal and should clear the CLEAR TO OUTPUT bit if it is set. This line will have no activity on it until the CDC 6600 sends a BYE command to release it.

The terminal may attempt a reconnection but that will be handled on a separate line and the CDC 6600 will make the connection between the old and new lines and release the old line.

e. Interrupts Received From a Terminal.

The ELF also can receive TELNET control functions from the terminals. The EC (Erase Character) and EL (Erase Line) functions can be handled by the ELF since an entire line is being buffered by the ELF. The IP (Interrupt Process) and AO (Abort Output) functions need to be passed onto the CDC 6600 since the process runs in the CDC 6600 and it may be holding output as well as the ELF. When the IP function is received from the terminal, the ELF should discard all queued data to/from the terminal and set the PROCESS INTERRUPT bit in the auxiliary status byte. When the AO function is received from the terminal, the ELF should discard any queued output data for the terminal and set the ABORT OUTPUT INTERRUPT bit in the auxiliary status byte. When the CDC 6600 has detected an interrupt bit as set, it will take the action it wishes and then acknowledge the interrupt by issuing the CLEAR INTERRUPT command with the appropriate mask.

f. Stopping the Interactive Session

When the CDC 6600 is ready to release an interactive line it issues the BYE command. The ELF should set the status to AVAILABLE and respond with an ACK block. If the connection to the terminal still exists, the ELF should close the connection to that terminal and queued data sould be discarded.

f. List of Interactive-Oriented Commands Issued by the CDC 6600

The following pages describe the interactive-oriented commands and the conditions under which these commands can be issued by the CDC 6600. Each command is in the format of a standard transfer block (Section II.D.2). The descriptions contain the specific details about what line numbers are valid, what the parameters are and how many there sould be, under what status conditions is the command valid, and the processing requirement of the ELF. The last section is "Status Changes and ELF Responses to Command." Status changes only refer to section one of the interactive line status table entires, the section that can be changed by the ELF. "Responses" are the types of information the ELF is allowed to send to the CDC 6600 after receiving this command. Some response must be made to every command the CDC 6600 issues.

COMMAND: Initialize Interactive Processing

BLOCK IDENTIFIER: 65 (decimal)

LINE TYPE: 0

PARAMETERS: Parameter 1: Maximum number of interactive lines.
Parameter 2: Maximum size for interactive buffer (in bytes).

PURPOSE OF COMMAND: The CDC wishes to begin interactive processing.

WHEN CAN COMMAND BE ISSUED: When interactive processing is not active.

ELF PROCESSING REQUIREMENT: The ELF should verify that access to the network is possible. It must configure the number of interactive lines and the size of their buffers and place these values in the line 0 status table. See section IV.D.1.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If interactive processing is started, set the status byte of all configured interactive lines to AVAILABLE status. Resonse is the line 0 status table.

2. If FTP processing cannot be started, there are no status changes and the response is a NAK block.

COMMAND: Stop Interactive Processing

BLOCK IDENTIFIER: 66 (decimal)

LINE TYPE: 0

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 wants to stop interactive processing.

WHEN CAN COMMAND BE ISSUED: When interactive processing is active.

ELF PROCESSING REQUIREMENT: The ELF should terminate all connections to terminals and accept no more connections. All queued data should be discarded. The interactive parameters in the line 0 status table should be cleared.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The status bytes of all interactive lines are set to AVAILABLE status. The response is an ACK block.

COMMAND: Start Interactive Processing

BLOCK IDENTIFIER: 67 (decimal)

LINE TYPE: Interactive

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 is acknowledging and accepting a new terminal connection the ELF has received.

WHEN CAN COMMAND BE ISSUED: When the status byte contains NEW INTERACTIVE CONNECTION status.

ELF PROCESSING REQUIREMENT: Only the status changes described below. See Section IV.E.4.a.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: Change the status byte to NORMAL status and set the CLEAR TO OUTPUT and the TERMINAL READY bits in the auxiliary status byte. The response is the status table entry.

COMMAND: Transfer Interactive Data from ELF

BLOCK IDENTIFIER: 68 (decimal)

LINE TYPE: Interactive

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 wants the ELF to send it an input line queued in the ELF.

WHEN CAN COMMAND BE ISSUED: When the status byte contains NORMAL status and the CLEAR TO INPUT bit is set in the auxiliary status byte.

ELF PROCESSING REQUIREMENT: Send the input line to the CDC 6600. See Section IV.E.3.b.(2).

STATUS CHANGES AND ELF RESPONSES TO COMMAND: The status byte is changed to WAITING FOR XFER ACK. The response is the input line queued in the ELF.

COMMAND: Good Interactive Data Transfer

BLOCK IDENTIFIER: 69 (decimal)

LINE TYPE: Interactive

PARAMETERS: None

PURPOSE OF COMMAND: This tells the ELF it can discard the last interactive data block sent to the CDC 6600 for this line.

WHEN CAN COMMAND BE ISSUED: When status byte contains WAITING FOR XFER ACK status.

ELF PROCESSING REQUIREMENT: Discard the last interactive data block sent to the CDC 6600 for this line. See Section IV.E.3.b.(2).

STATUS CHANGES AND ELF RESPONSES TO COMMAND: Change the status byte to NORMAL status. Unless another interactive data block is now ready to be sent to the CDC 6600, clear the CLEAR TO INPUT bit in the auxiliary byte. Response is the status table entry.

COMMAND: Transfer Interactive Data to ELF

BLOCK IDENTIFIER: 70 (decimal)

LINE TYPE: Interactive

PARAMETERS: Parameter 1: Byte count of data bytes to be sent. This does not include the 4 byte block header on the 2 byte data header.

PURPOSE OF COMMAND: This command gives warning to the ELF that the next block to be issued is an interactive data block for this line and gives the length of that block.

WHEN CAN COMMAND BE ISSUED: When the status byte contains NORMAL status and the CLEAR TO OUTPUT bit is set in the auxiliary status byte.

ELF PROCESSING REQUIREMENT: Prepare to receive an interactive data block. See Section IV.E.3.a.(2).

STATUS CHANGES AND ELF RESPONSES TO COMMAND: Set the status byte to WAITING FOR DATA status. Response is the status table entry.

COMMAND: Interactive Data Block

BLOCK IDENTIFIER: 71 (decimal)

LINE TYPE: Interactive

PARAMETERS: A minimum of 3 parameter bytes. See Section IV.E.2 for the format of this block.

PURPOSE OF COMMAND: This is not a command, but all or part of an interactive output line that the CDC 6600 wants the ELF to send to a remote terminal.

WHEN CAN COMMAND BE ISSUED: When status byte contains WAITING FOR DATA status.

ELF PROCESSING REQUIREMENT: The ELF should reformat the data block and send it to the remote terminal. See Section IV.E.3.a for the detailed procedure.

STATUS CHANGES AND ELF RESPONSES TO COMMAND:

1. If the ELF received the data block correctly from the CDC 6600, set the status byte to NORMAL status and clear the CLEAR TO OUTPUT bit in the auxiliary status byte. Response is the status table entry.

2. If the ELF did not receive the data block correctly from the CDC 6600, set the status byte to NORMAL status. Do not clear the CLEAR TO OUTPUT bit in the auxiliary status byte. Response is an Error Response Block.

COMMAND: Clear Interrupt

BLOCK IDENTIFIER: 72 (decimal)

LINE TYPE: Interactive

PARAMETERS: Parameter 1: One byte mask used to clear the AO interrupt flag and/or the IP interrupt flag. Valid values are octal 10, 20, or 30.

PURPOSE OF COMMAND: The CDC 6600 is acknowledging the interrupts and asking the ELF to clear them.

WHEN CAN COMMAND BE ISSUED: Anytime that interactive processing is active.

ELF PROCESSING REQUIREMENT: Check the parameters for valid values. Take the logical product of the auxiliary status byte and the logical complement of the parameter mask. Put the result in the auxiliary status byte.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: There are no changes to the status byte. The auxiliary status byte has one or more interrupt flags set to zero. Response is the status table entry.

COMMAND: Bye

BLOCK IDENTIFIER: 73 (decimal)

LINE TYPE: Interactive

PARAMETERS: None

PURPOSE OF COMMAND: The CDC 6600 wishes to terminate an interactive processing session with a particular remote terminal.

WHEN CAN COMMAND BE ISSUED: When the status byte contains NORMAL or NEW INTERACTIVE CONNECTION status.

ELF PROCESSING REQUIREMENT: Close any existing connections with the terminal corresponding to this line. Discard all queued data for this line. See Section IV.D.2.b.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: Change the status byte to AVAILABLE status. Response is an ACK block.

COMMAND: Issue Go Ahead Signal

BLOCK IDENTIFIER: 74 (decimal)

LINE TYPE: Interactive

PARAMETERS: None

PURPOSE OF COMMAND: This commands tells the ELF to issue a TELNET GO AHEAD signal at the end of any queued output for the terminal corresponding to this line.

WHEN CAN COMMAND BE ISSUED: When status byte contains NORMAL status.

ELF PROCESSING REQUIREMENT: The ELF inserts a TELNET GO AHEAD signal at the end of any queued output for the terminal corresponding to this line.

STATUS CHANGES AND ELF RESPONSES TO COMMAND: No status changes. The response is an ACK block.

ADDENDUM TO SPECIFICATION

I. PURPOSE

This is an addendum to the specification to connect a CDC 6600 to a PDP-11 system, known as the ELF, in order to connect to the ARPA network. In the area of file transfer, the original specification was limited to the transfer of files between AFSC CDC 6600 sites. This addendum contains the necessary additional specifications to send or accept text files to/from any source on the network using the network default transfer standards. This addendum does not modify any information in the original specification, it merely adds another alternative.

II. COMBINATIONS OF TRANSFER PARAMETERS TO BE SUPPORTED

There are three combinations of transfer parameters that are supported under the original specification and this addendum. The original specification supported the following combination: a 12 bit byte size, IMAGE data representation type, RECORD structure, and BLOCK transmission mode. This addendum supports the following two combinations: an 8 bit byte size, ASCII, NON-PRINT data representation type, RECORD or FILE structure, and STREAM transmission mode.

III. END-OF-RECORD AND END-OF-FILE INDICATORS

The STREAM mode of data transfer employs different End-of-Record and End-of-File indicators than the BLOCK mode of transfer. Both sets of indicators are fully described in the standard network protocol documentation. The data block descriptors shown in the original specification also hold for data blocks when STREAM mode is used.

IV. CDC 6600 DEFINITION OF A LINE OF TEXT

A. Central Memory Word Size

The word size of the CDC 6600 central memory is 60 bits. This is logically divided into five 12-bit sections called bytes. This byte corresponds to the 12-bit word size of the CDC 6600 Peripheral Processor and the definition of byte in the original specification. One central memory word can contain ten 6-bit display code characters.

B. Line of Text Terminator

When the CDC 6600 is handling text files, a line of text is an integral number of central memory words. A line of text always ends at a word boundary. The end of a line of text is denoted by the last byte (bits 11-0) of a central memory word being equal to binary zero. The zero value in the last byte of a central memory word will be called the "line of text terminator."

C. Word Padding

If the valid characters in a line do not fill the last central memory word up to the line of text terminator in the last byte, the unused area in the central memory word is filled with binary zero. The worst case of this situation is when the number of 6-bit display code characters in a line is 10N+9. Then there is not room for the 12-bit line of text terminator in this central memory word. The last six bits of this central memory word and the first 48 bits of the next central memory word must be padded with zero. Since the line of text terminator is zero, this means the last central memory word of this line is zero.

D. Handling of Colons

Problems are rampant when the colon character is used. This is because the display code value for the colon is the 6-bit value zero, and the line of text terminator and word padding are also zero. It is impossible to have two colons in the last two character positions of a central memory word. This is because it would be interpreted as the line of text terminator. A special case occurs when one or more colons are the last valid characters in a line. In this case, the colons would be interpreted as word padding and not valid characters. The technique used to avoid this problem is to follow the colon or colons with a blank. This is done even if it forces the use of another central memory word to hold the line of text terminator.

V. DATA CONVERSION PROCEDURES

A. Requirements of the Data Conversion Procedure

This addendum gives the rules the ELF must follow when doing a generalized text file transfer between the local CDC 6600 and a remote host. The following sections describe the procedure needed to convert between CDC 6600 display code and ASCII code used to transmit characters over the network. The following requirements are placed on these processes:

These procedures must comply with the requirements of the original specification.

These procedures must implement the CDC 6600 standards for text files as given in Section IV.

These procedures must be invertible; that is, if a text file was sent to another AFSC CDC 6600 using this generalized text file transfer method, the file at the receiving end should be an exact copy of the original file.

B. Buffer Sizes

When the ELF sends a block of FTP data to the CDC 6600 it must send a block that is an integral number of buffer units, unless the EOR or EOF bit is set in the data block descriptor. A buffer unit is defined in the original specification and its size was set at 320 12-bit bytes. When dealing with text files, this buffer unit must be subdivided into 64 5-byte sections that correspond to the CDC 6600 central memory word.

An additional requirement in the case of text file transfers when the EOR or EOF bit is set in the data block descriptor is that the data block contain an integral number of words. This requirement will become evident in Section V.F.

C. Line of Text Terminator

1. Display Code Line of Text Terminator

The line of text terminator is used by the CDC 6600 to determine the end of a text line. The line of text terminator is defined by the ELF the same as it is defined by the CDC 6600 in Section IV.B; that is, if the last byte of a word is zero, that byte is a line of text terminator.

2. ASCII Line of Text Terminator

Under the ASCII data representation type, the two ASCII character sequence CRLF is the line of text terminator.

D. Code Conversion Table

The ELF should use the code conversion table in Section II.H.4 in the original specification. When converting from display code to ASCII code, if there are more than one ASCII values listed, the first value listed should be used. When converting from ASCII to display code, the values outside the range of hexadecimal 20 to 7E have no counterpart in display code and should be discarded. Only the CRLF sequence has meaning from the range of characters to be discarded.

E. Display Code to ASCII Conversion Procedures

When the ELF receives a block of data from the CDC 6600 that is for a text file transfer it should handle the data in 5 byte word groups. The ELF should first examine the present word to see if the last byte is zero; that is, does the word contain a line of text terminator.

If the word does not contain a line of text terminator, each character should be converted to ASCII using the conversion table and procedure referenced in Section V.D. The ASCII characters are then placed in a buffer to be transmitted to the remote host. If the last character in the word is zero, a special case may occur. This is explained in the last paragraph of this section.

If the word does contain a line of text terminator, the ELF should begin at the last character in the word and work backwards until it finds a non-zero character. When a non-zero character is found, the characters from the beginning of the word through the last non-zero character found are converted to ASCII, as above, and placed in a buffer to be transmitted to the remote host. An ASCII line of text terminator is then placed in the buffer.

It is possible that the entire word that contains the display code line of text terminator could be zero. In this case, the last character of the previous word must be checked for display code zero. If it is zero, this character is word padding and not a display colon; the ASCII colon that has been placed in the buffer to be transmitted must be removed. The ASCII line of text terminator should be placed in the buffer.

F. ASCII to Display Code Conversion Procedures

When the ELF receives a block of data from the remote host that is for a text file transfer, each character should be converted to display code using the conversion table and procedure referenced in Section V.D. The display code characters are placed in a buffer for later transmission to the CDC 6600. When the ASCII line of text terminator is detected, the ELF must set up a proper CDC 6600 text line.

If the last valid ASCII character that the ELF received before a line of text terminator was a colon, the ELF should insert an octal 55, a display code blank, in the buffer after the display code colon.

The ELF pads the remainder, if any, of the word with binary zeros. If the present word does not have 12 bits of binary zero in the last byte of the word, another word of binary zero must be placed in the buffer.

A special case that must be handled by the ELF software is when the ELF receives two successive ASCII colons from the network that will be placed in the last two character positions of a word. This would be interpreted as a line of text terminator by the CDC 6600. The choice must be made between preserving the line structure or changing a character. The line structure will be preserved. The ELF should change the colon in the last character position of the word to an octal 55, a display code blank.

There is no restriction that the last word of a buffer unit must contain a line of text terminator. The buffer unit size is just equal to the disk sector size on the CDC 6600 and has no relation to the size of a particular line.

RJC 19-MAY-75 06:07 32539

FOR OFFICIAL USE ONLY

PR-B-5 3143

(J32539) 19-MAY-75 06:07;;;; Title: Author(s): Roberta J, Carrier/RJC; Distribution: /RJC( [ INFO-ONLY ] ) ; Sub-Collections: NIC; Clerk: RJC; Origin: < CARRIER, SOWDRAFT/TFL.NLS;2, >, 9-MAY-75 12:20 RJC ;;;; ####;

REPORT ON TELECONFERENCING PANEL 1

REPORT 1a

The teleconferencing session of the Airlie house conference on telecommunications policy consisted of three presentations and one overview by a discussant. The chairman opened the meeting by noting that this particular panel was a logical wrap up to several other ones that had touched upon the concept of teleconferencing during the conference. These sessions included those on electronic newspapers, electronic mail, satellite and interactive cable t.v. telecommunications. These previous sessions reviewed both the technology and two types of impacts that can be expected through the use of teleconferencing systems. The chairman also noted that this discussion of teleconferencing was closely related to another area of telecommunications research: that of attempting to examine the potential impacts that may arise from the trade-offs between travel and communications in the future. 1a1

Roger Hough of the Stanford Research Insitute began the presentations with an overview of a number of voice and video teleconferencing systems that currently exist in Europe, Canada, Japan, Australia, and the United States. This review is based upon a current S.R.I. review of teleconferencing for the National Science Foundation. Hough's presentation sparked a number of questions from the audience regarding details on the various systems he reviewed. The basic similarity of many of the video systems was noted as well as the difficulty of attaining realistic cost/benefit trade-offs on these systems. On the other hand, the audio systems often do meet cost/benefit analyses and several have become used in operational environments in Canada and the U.K. Part of the discussion in this area of Hough's presentation centered around the value of speaker identification capabilities for audio systems. 1a2

Maxine Rockoff of H.E.W. continued the discussion of existing teleconferencing systems with a review of a series of experimental medical teleconferencing systems that have been tested in a variety of medical situations. She noted that the systems often optimize the time of the patient verses that of the medical workers. This is , of course, the opposite to today's conditions in the medical field in general.This could lead to resistance over the long term to the widespread implementation of these systems. 1a3

Rockoff discussed how medical teleconferencing systems are used to fulfill a variety of medical communications needs. Consultation between various types of physicians (generalists and specialists) and between other medical personnel and physicians have been augmented in several experiments funded by H.E.W. Supervision of various medical tasks by a remote professional is another application of medical teleconferencing. One case example presented was the preparation of prescriptions by an assistant and supervised by a pharmacist in a remote hospital. Telemedicine is a form of teleconferencing that directly involves the patient. Several experiments in this area have been undertaken. Other uses of teleconferencing involved the routine but important activities of administration, management, education, and training. Rockoff concluded by noting that while most of the systems reviewed in her presentation were visual in nature , an integrated approach to medical teleconferencing is required in the future so that the various medical needs for communications can be served by a variety of systems capabilities on a demand basis. 1a4

The formal presentations were concluded by a review of the computer conferencing field by Larry Day of Bell Canada. Day noted that some of the concepts associated with computer conferencing had been touched upon by the earlier panel on electronic mail. He defined computer conferencing as : "the use of shared computer files, remote terminal equipment, and telecommunications networks to facilitate group communications where face-to-face contact is either not possible or less desirable." He then outlined several common features associated with various computer conferencing systems. These features include the following capabilities: shared computer files, text editing, synchronous and asynchronous message handling, private messages, anonymous messages, a permanent conference record, and various specialized housekeeping features. Common applications of computer conferencing were then discussed: crisis management, remote joint authorship, scientific and technical information exchange, conflict resolution, and policy research. Several specific computer conferencing systems were reviewed and a few predictions on the future potential of computer conferencing were offered. 1a5

Ed Van Vleck provided an overview of teleconferencing from a historical and policy perspective as he discussed the three presentations. He noted that 15 years ago teleconferencing was just a buzz word used by a few communications professionals. The presentations had illustrated how far the field had come but also how far it still had to go. Most of the systems reviewed were experimental in nature and few met most forms of cost/benefit analysis. There were few long term operational systems and some that had existed in the past had not survived. Van Vleck noted that most of the policy implications associated with teleconferencing were being examined in the closely related field of potential travel/communications trade-offs . Many of these policy issues were also associated with the broader field of telecommunications rather than just teleconferencing. He briefly summarized some of the policy issues in the following categories: impacts on patterns of social interaction, the privacy issue, impacts on home life, fraud and disruption potential of teleconfereincing systems, government subsidy patterns between the transportation and communications industries, sector unemployment, and regulatory system impacts. 1a6

The discussion that followed ranged over a wide range of issues. Some questioners sought further information on various teleconferencing systems reviewed earlier. Others related various experiments and anecdotal experience associated with the attempts to use and evaluate teleconferencing systems. The policy issues associated with teleconferencing were also addressed. The chairman closed by noting that the large attendance and vigorous question period in itself indicated that teleconferencing was a subject of great interest to those associated with telecommunications policy research. 1a7

ABSTRACTS 1b

International Teleconferencing 1b1

Roger Hough 1b2

Teleconferencing is a topic that is often talked or written about in generalities. At the opposite end of the spectrum, the technical or operational aspects of particular systems are discussed or promoted. However, most of us do not have an awareness of the scope of teleconferencing as it is practiced today around the world. This paper reviews the results of a recent study sponsored by the National Science Foundation that attempted to determine the status of teleconferencing in the "real world". The study examines voice, visual, and computer teleconferencing on an international scale as it is practiced in business, government, and educational institutions. This paper will present a sampling of the usual and unusual applications and experiences with the various forms of teleconferencing in the above institutions. 1b3

Medical Teleconferencing 1b4

Maxine L. Rockoff 1b5

The utility of broad-band interactive two-way visual telecommunication has been assessed in a wide variety of health-care delivery settings in seven exploratory projects funded by the Bureau of Health Services Research, Department of Health, Education, and Welfare. These projects are briefly summarized and communications events are classified into five exhaustive but not mutually exclusive categories: consultation, supervision, direct patient care(telemedicine), administation and management, and education and training. Technical and human factors to be considered in the design of a visual telecommunication system are identified. Trade-offs available among them are discussed, such as resolution for remote controls and frame rate. The impact of visual telecommunication on the often competitive performance goals of the health-care system is discussed, e.g., random video consultations may simultaneously decrease physician productivity and patient inconvenience. The paper concludes by suggesting directions for future research and by noting that broad-band video is but one among many technological, organizational, manpower, and other options available to those who seek solutions to the problems of health-care delivery. 1b6

Computer Conferencing 1b7

Lawrence H. Day 1b8

This paper reviews the rapidly developing field of computer conferencing. It begins with a historical overview of the early activities at the Office of Emergency Preparedness (Executive Office of the President) and The Institute for the Future. The basic features that are available on most computer conferencing systems are described . This is followed by an examination of the specific features of the systems developed at OEP and IFTF. The relevant features of the Augmentation Research Center computer developed at the Stanford Research Institute are also reviewed. 1b9

The next section of the paper reviews the various types of applications for computer conferencing. These include: crisis management, remote joint authorship, information sharing, conflict resolution, taboo discussions, and policy research. The implications for the use of computer conferencing on an international scale are also discussed: international technology transfer, international collaboration, international policy determination, and applications for the Less Developed Countries . 1b10

The Third section of the paper illustrates how computer conferencing lends itself to post analysis of the dynamics of a particular conference. The paper concludes with an analysis of some of the problem areas that will have to be resolved before computer conferencing becomes a widespread tool. 1b11

Airlie House Report 

(J32540) 19-MAY-75 08:19;;;; Title: Author(s): Lawrence H. Day/LHD;
Distribution: /RWH( [ ACTION ] ) LHD( [ INFO-ONLY ] ) ; Sub-Collections:
NIC; Clerk: LHD; Origin: < DAY, AIRLIE.NLS;1, >, 19-MAY-75 08:11
LHD ;;;;####;

FORM FOR INPUTS TO VUGRAPHS

I sent this to RADC as an expedient. If it doesn't apply to you forget it!

FORM FOR INPUTS TO VUGRAPHS

TITLE: (same as name used elsewhere) 1

PROJ. ENGR: 2

OBJECTIVE: 3

USERS: 4

HW/SW: 5

FUNDING: (include source) 6

M/Y: (If more than one man include names) 7

CONTRACTOR: 8

CONTRACT NUMBER: 9

PROJECT START: 10

COMPLETION DATE: 11

MILESTONES: 12

PROBLEMS OR POINTS OF SPECIAL INTEREST: 13

FORM FOR INPUTS TO VUGRAPHS

(J32541) 19-MAY-75 09:44;;;; Title: Author(s): Edmund J. Kennedy/EJK; Distribution: /RADC( [ ACTION ] ) ; Sub-Collections: RADC; Clerk: EJK;

I Got It, Already!

Dear Frank:

Thank you for the response on the ELF password.

JJZ

I Got It, Already!

(J32542) 19-MAY-75 10:43;;;; Title: Author(s): John J. Zenor/JJZ;
Distribution: /FGB( [ ACTION ] ) ; Sub-Collections: NIC; Clerk: JJZ;

The NSF/OSIS EPC project for SRI is still alive

Via a flurry of calls over the past five days, we've determined that there is still life in the prospects of getting the NSF/OSIS EPC Project going at SRI. Here is the brief history of recent events: 1

At some point, like over a month ago, Tom Humphrey and Dave Brown came to the understanding that OSIS, having accepted the revised proposal, was countering with a plan to ask us to run the project at about half budget. 1a

The processing had gotten into some sort of loop back at NSF, and we had yet to hear from their contracts people to start the negotiations; the expectation here was that when we did, we'd learn about the extent of their proposed funding-level reduction. 1a1

At some later time, Stew Blake informed Tom that the Institute had to withdraw its offer to contribute the capital equipment to that project, as had been set up under the "cost sharing" that the program solicitation required for the EPC proposal. Since the proposal-offer period had elapsed, SRI was no longer legally obligated; we could simply "withdraw our offer to do the work." 1b

Tom and Dave Brown and Norm Nielsen and I concluded that these two cut-backs effectively knocked out our getting the EPC Project; we turned our energy towards organizing the DDPCS Community. 1c

Last week I (finally) reached Dr. Harold Bamford of NSF's OSIS, the man who formulated the EPC concept and with whom we've done all of the associated communication to date. I gave him the news about the Institute's not being able to come up with the cash. I talked perhaps 10 minutes and had to go to other things; because this hadn't been enough talking, I called back the next day. We talked for another ten minutes, and arranged for a longer talk soon. Saturday I talked with him for over an hour. Here is the essence of the new information: 2

He apparently hadn't planned to cut the project's funds -- what he did was to schedule the funding in two lots, \$150K from FY75 and \$145K from FY76 (by August) funding. 2a

Bamford mentioned that this contract had been put into a new contract-administrative category, called a "cooperative agreement," and that it was due to their own internal problems in getting this approach to work that the delay had occurred. NSF has decided to give up on that approach, and was turning to a more standard arrangement. 2b

He appeared to be very disappointed at losing an important piece of his plan. He urged us to consider any way that we could for altering our proposal in a way that could both meet the Institute's financial conditions and NSF's legal obligation (since their Program Solicitaion had explictly specified that the EPC Project would have to be a cost-shared one, there was no way that he could let us off the hook). 2c

He pointed out that the $120-or-so K was quite a steep bit of cost sharing, more than would be required to meet the spirit of the requirement; but, zero was too small. 2c1

He didn't say what their minimum would be, but it is clear that he'd like us to make a new offering. He pointed out that the costs being shared didn't have to appear as cash outlay -- any reasonable benefit given by SRI to the project could be counted, such as overhead labor, free use of proprietary things (e.g. perhaps the Institut's self-paid document-production study, if the results contribute significatly to the project, could provide some "cost-sharing" value?), etc. 2c2

He also mentioned that, if the total resources available to the Project were thus reduced by a lower contribution from SRI, they could consider putting in a bit more money if the successful project completion required it. 2c3

He further stated that, if we wanted to try altering our proposal, that he could re-program his funds in such a way as to hang on to the whole $295K into FY76 in case we could come to terms. 2d

I explained the multi-participant approach that we were explictly pursuing, and how we figured that there would be larger resources available in the technical documentation applications to pay for the expected large development costs of the full DDPC System. He perceives the logic, but said that they were committed to working with professional journals. 2e

We left it (I think) that he'd be quite willing to consider the pacing and staging of his journal-oriented EPC to maximize the payoff that could be achieved through cooperative sharing of developments and experience with any DDPCS community that we got going. 2e1

I think, in effect, that he'd implicitly go along with his EPC activity being a part of that Community. I'd very much like to see our re-formulation of an approach for their EPC take this into account. We did discuss some related possibilities; for instance, that OCR equipment could be tested at a later stage, etc., recognizing that the more critical parts of the experiment had to do with how the users took to the techniques, and whether when the resulting methodologies were worked out, the costs and benefits could reasonably be assessed. Note that the costs wouldn't have to be those associated with the particular means by which the supporting services were provided when the users' systems were experimented with, but could pertain to engineering evaluations of costs under whatever limited-purpose service-system design stipulations were desired. 2e2

Dave Brown and I talked briefly on the phone afterwards. We agreed that we'd ask Bamford to save our funding for early FY76, and to count on our trying to come back with a an altered proposal that could meet their cost-sharing requirements. We'll both be at the NCC for most of the week; we'll plan to pick up on a re-formulation study when we return. 3

NOTE: I would very much like to see this project become a real part of what to us at SRI is an emergent DDPCS Community. The opportunity seems real, and the payoff could be significant, both to NSF and to our Community. 4

The NSF/OSIS EPC project for SRI is still alive

(J32544) 19-MAY-75 12:29;;;; Title: Author(s): Douglas C. Engelbart/DCE; Distribution: /TLH( [ ACTION ] ) PWO( [ ACTION ] ) DVN( [ ACTION ] ) NDM( [ INFO-ONLY ] ) KLM( [ INFO-ONLY ] ) NRN( [ INFO-ONLY ] ) RLB2( [ INFO-ONLY ] ) EKM( [ INFO-ONLY ] ) RWW( [ INFO-ONLY ] ) JCN( [ INFO-ONLY ] ) RLL( [ INFO-ONLY ] ) DRB( [ INFO-ONLY ] ) DCE( [ INFO-ONLY ] ) BC( [ INFO-ONLY ] Bart: Note, a resurrected possibility) ; Sub-Collections: SRI-ARC; Clerk: DCE;

Output Processor Suggestions 1

will set all the margins. 1a

or will show a blank line between EACH STATEMENT. If you also want a blank line between each LINE of the statement, also insert 1b

Headers: The header you define will start to take effect "on the page following the one which includes this directive," according to the Output Processor Users Guide. Thus, if you put the header directive in Statement 0 or 1 which print on the first page, the first appearance of the header will be on the next page (page 2). 1c

Footers occur on the bottom of the page you set them, because they come "after" the place in the text you set them. The OP can't print a header until after it reads the statement with the directive (the next time is the next page), since it processes one statement at a time. 1d

THE FIX: Insert the header directive (and you can include the other initializing directives here too) at the end of the origin statement (A: 0+e), and also insert the directive to Paginate at the End of this Statement. Thus the next page will begin with statement one and have the header on it; the origin will be on a page by itself (which is usually allright with me since that technical garbage looks funny at the top of a report--I throw the origin page away). Now, if you are still using the normal footer which contains page numbers, you probably want your origin statement page to be numbered "0" rather than "1" so that the first full page will be numbered "1". If so, include in the origin statement the directive to set that Page Number to zero. 1e

Output Processor Suggestions

In summary, my suggestion is to Insert the following Text at the end of your origin statement (A: 0+e): 1f

[opt: ] 1f1

........

and, before giving the Output Terminal command, have your CM located at the beginning of the origin statement (Jump to Address A: 0 CR). 1g

For Terminet printout: 1h

Output Terminal OK (Send Form Feeds?) N (Simulate?) Y (Wait at page breaks?) N (Go?) N (Type CA when ready, CD to abort) 1h1

Now, before you type anything else, move your pager up manually until the print mechanism is one inch (6 lines) ABOVE the next perforation (You'll have to go down to the perforation and then back up one inch; you'll soon learn to line it up by the holes in the paper vis-a-vis the plastic flap). After you have set the paper, then type just the Carriage Return key to set the file going. The dotted line will print one inch later. 1i

Try all this and let me know. If you want faster responses to questions like this in the future, address your questions via SNDMSG to Feedback at Office-1. Feedback is set up to answer questions the next day (I don't mind, but sometimes am slow in answering my mail.). 1j

Output Processor Suggestions

(J32545) 19-MAY-75 13:19;;;; Title: Author(s): Jeanne M. Beck/JMB; Distribution: /FPA( [ ACTION ] ) FEEDBACK( [ INFO-ONLY ] just to let you in on what I answered him in case you get called into it--beck,inmes,0163) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: JMB;

Message from DAP on Memo program possibilities

19-MAY-75 0545-PDT POTTER: ETSMEMO (what else?)
Distribution: MEYER, potter
Received at: 19-MAY-75 05:45:19 1

Dean,
Sorry to take so long getting back to you this time, but the last week was really a mess. Anyway: it's almost perfect. The only problem is still in the subject block, to wit: it seems to put only one word of the subject on each line, regardless of whether or not there's space available for more. the subject itself can be allowed to gett quite close -- withing four spaces or even three -- of the "D" in "Date."
Aside from that it's great. Overwhelmed by altruism, though, I am moved to ask if you've thought about mechanisms by which alternate format(s) could be provided -- like a full block, e.g.: 1a

Memo for: Mr. Meyer
Mr. Norton
Ms. Roetter

cc: Mr. Engelbart

Subject: Memo formats
Date: 19 April 1975
From: David A. Potter 1a1

This is a sample of a full block memo format. It's not a format that's used at ETS, but I suspect that other members of the OFFICE-1 community might find it useful. Of course, it has the added advantage of being very easy to set up.

If you should decide to provide a full block format as an alternative to the standard ETS format, I'd suggest that you first check a standard secretarial handbook to get a really "correct" format...or check a standard secretary for guidance. 1b

Message from DAP on Memo program possibilities

(J32546) 19-MAY-75 13:30;;;; Title: Author(s): David A. Potter/DAP; Distribution: /FEEDBACK( [ INFO-ONLY ] ) ; Sub-Collections: NIC FEEDBACK; Clerk: NDM;

```
FILE cetsmemo % (CML.SAV,) TO (etsmemo.cml,) %                         1
  % DECLARATIONS %                                                     1a
    DECLARE PARSEFUNCTION                                              1a1
      answ,               % reads answer construct %                   1a1a
      answer,      % for questions - returns  0/1 %                    1a1b
      sp,          % reads next char, TRUE if space %                  1a1c
      readconfirm, % reads next char if ca %                           1a1d
      readbug,     % reads next char if BUG %                          1a1e
      readoption,  % TRUE if next char is optchar %                    1a1f
      readrepeat,  % TRUE if next char is repeat %                     1a1g
      lookansw,    % TRUE if next char is Y/CA %                       1a1h
      lookconfirm, % TRUE if next char is CA/REPEAT/INSERT %           1a1i
      lookbug,     % TRUE if next char is BUG %                        1a1j
      looknum,     % TRUE if next char is a number %                   1a1k
      clearname,   % clears the name area %                            1a1l
      notca;       % reads next char, TRUE if not CA char %            1a1m
    DECLARE COMMAND WORD                                               1a2
      "BRANCH" = 1 ,                                                   1a2a
      "GROUP" = 2 ,                                                    1a2b
      "PLEX" = 3 ,                                                     1a2c
      "STATEMENT" = 4 ,                                                1a2d
      "CHARACTER" = 5 ,                                                1a2e
      "CONTROLCHAR" = 6 ,                                              1a2f
      "INVISIBLE" = 7 ,                                                1a2g
      "LINK" = 8 ,                                                     1a2h
```

"DIRECTORY" = 9 , 1a2i

"PASSWORD" = 10 , 1a2j

"NUMBER" = 11 , 1a2k

"TEXT" = 12 , 1a2l

"VISIBLE" = 13 , 1a2m

"WORD" = 14 , 1a2n

"FILE" = 15 , 1a2o

"NEWFILELINK" = 16 , 1a2p

"OLDFILELINK" = 17 , 1a2q

"NAME" = 18 , 1a2r

"IDENT" = 19 , 1a2s

"IDENTLIST" = 20 , 1a2t

"EDGE" = 21 , 1a2u

"MARKER" = 22 , 1a2v

"NLS" = 23 , 1a2w

"ITEM" = 24 , 1a2x

"ITEMNOVS" = 25 , 1a2y

"SUCCESSOR" = 26 , 1a2z

"PREDECESSOR" = 27 , 1a2a@

"UP" = 28 , 1a2aa

"DOWN" = 29 , 1a2ab

"HEAD" = 30 , 1a2ac

"TAIL" = 31 , 1a2ad

"END" = 32 , 1a2ae

"BACK" = 33 , 1a2af

"NEXT" = 34 , 1a2ag

"ORIGIN" = 35 , 1a2ah

"FILERETURN" = 36 , 1a2ai

"RETURN" = 37 , 1a2aj

"FILENAME" = 38 , 1a2ak

"FIRSTNAME" = 39 , 1a2al

"NEXTNAME" = 40 , 1a2am

"EXTNAME" = 41 , 1a2an

"FIRSTCONTENT" = 42 , 1a2ao

"NEXTCONTENT" = 43 , 1a2ap

"FIRSTWORD" = 44 , 1a2aq

"NEXTWORD" = 45 , 1a2ar

"DETACHED" = 46 , 1a2as

"TTY" = 47 , 1a2at

"AUTO" = 48 , 1a2au

"CONTINUE" = 49 , 1a2av

"ON" = 50 , 1a2aw

"RECOVER" = 51 , 1a2ax

"SLINKER" = 52 , 1a2ay

"UPDATE" = 53 , 1a2az

"CLEAR" = 54 , 1a2b@

"IDENTS" = 55 , 1a2ba

"FILES" = 56 , 1a2bb

"DELETE" = 57 , 1a2bc

"DEFERRED" = 58 , 1a2bd

```
"IMMEDIATE" = 59 ,                                  1a2be
"NOT" = 60 ,                                        1a2bf
"PREVENT" = 61 ,                                    1a2bg
"RESET" = 62 ,                                      1a2bh
"ARCHIVE" = 63 ,                                    1a2bi
"SEQUENTIAL" = 64 ,                                 1a2bj
"TWO" = 65 ,                                        1a2bk
"JUSTIFIED" = 66 ,                                  1a2bl
"ASSEMBLER" = 67 ,                                  1a2bm
"BOTH" = 68 ,                                       1a2bn
"UNDELETE" = 69 ,                                   1a2bo
"FOR" = 70 ,                                        1a2bp
"STATUS" = 71 ,                                     1a2bq
"TAPE" = 72 ,                                       1a2br
"ACCOUNT" = 73 ,                                    1a2bs
"NO" = 74 ,                                         1a2bt
"VERSIONS" = 75 ,                                   1a2bu
"EXTENSION" = 76 ,                                  1a2bv
"DATE" = 77 ,                                       1a2bw
"CREATION" = 78 ,                                   1a2bx
"LAST" = 79 ,                                       1a2by
"FIRST" = 80 ,                                      1a2bz
"READ" = 81 ,                                       1a2c@
"WRITE" = 82 ,                                      1a2ca
"DUMP" = 83 ,                                       1a2cb
```

```
"EVERYTHING" = 84 ,                                                     1a2cc
"LENGTH" = 85 ,                                                         1a2cd
"MISCELLANEOUS" = 86 ,                                                  1a2ce
"ACCESSES" = 87 ,                                                       1a2cf
"PROTECT" = 88 ,                                                        1a2cg
"SIZE" = 89 ,                                                           1a2ch
"TIME" = 90 ,                                                           1a2ci
"VERBOSE" = 91 ,                                                        1a2cj
"SORT" = 92 ,                                                           1a2ck
"BYTESIZE" = 93 ,                                                       1a2cl
"ARCHIVED" = 94 ,                                                       1a2cm
"ALL" = 95 ,                                                            1a2cn
"MODIFICATIONS" = 96 ,                                                  1a2co
"UPPER" = 97 ,                                                          1a2cp
"LOWER" = 98 ,                                                          1a2cq
"MODE" = 99 ,                                                           1a2cr
"SENDMAIL" = 100 ,                                                      1a2cs
"BUSY" = 101 ,                                                          1a2ct
"QUICKPRINT" = 102 ,                                                    1a2cu
"JOURNAL" = 103 ,                                                       1a2cv
"PRINTER" = 104 ,                                                       1a2cw
"COM" = 105 ,                                                           1a2cx
"TERMINAL" = 106 ,                                                      1a2cy
"REMOTE" = 107 ,                                                        1a2cz
"REST" = 108 ,                                                          1a2d@
```

"CASE" = 109 , 1a2da

"CONTENT" = 110 , 1a2db

"TEMPORARY" = 111 , 1a2dc

"VIEWSPECS" = 112 , 1a2dd

"EXTERNAL" = 113 , 1a2de

"TO" = 114 , 1a2df

"PRIVATE" = 115 , 1a2dg

"PUBLIC" = 116 , 1a2dh

"TENEX" = 117 , 1a2di

"ALLOW" = 118 , 1a2dj

"EXECUTE" = 119 , 1a2dk

"APPEND" = 120 , 1a2dl

"LIST" = 121 , 1a2dm

"SET" = 122 , 1a2dn

"SELF" = 123 , 1a2do

"FORBID" = 124 , 1a2dp

"DISK" = 125 , 1a2dq

"DEFAULT" = 126 , 1a2dr

"OLD" = 127 , 1a2ds

"NEW" = 128 , 1a2dt

"COMPACT" = 129 , 1a2du

"RENAME" = 130 , 1a2dv

"ADD" = 131 , 1a2dw

"SUBTRACT" = 132 , 1a2dx

"MULTIPLY" = 133 , 1a2dy

```
"DIVIDE" = 134 ,                                    1a2dz
"RIGHT" = 135 ,                                     1a2e@
"LEFT" = 136 ,                                      1a2ea
"ACTION" = 137 ,                                    1a2eb
"AUTHORS" = 138 ,                                   1a2ec
"COMMENT" = 139 ,                                   1a2ed
"EXPEDITE" = 140 ,                                  1a2ee
"HARDCOPY" = 141 ,                                  1a2ef
"INFORMATION" = 142 ,                               1a2eg
"INSERT" = 143 ,                                    1a2eh
"KEYWORDS" = 144 ,                                  1a2ei
"OBSOLETES" = 145 ,                                 1a2ej
"RFC" = 146 ,                                       1a2ek
"SUBCOLLECTIONS" = 147 ,                            1a2el
"TITLE" = 148 ,                                     1a2em
"UNRECORDED" = 149 ,                                1a2en
"L10" = 150 ,                                       1a2eo
"PROCEDURE" = 151 ,                                 1a2ep
"SEQGENERATOR" = 152 ,                              1a2eq
"BUFFER" = 153 ,                                    1a2er
"NDDI" = 154 ,                                      1a2es
"PARSERULE" = 155 ,                                 1a2et
"CA" = 156 ,                                        1a2eu
"CD" = 157 ,                                        1a2ev
"RPT" = 158 ,                                       1a2ew
```

```
"BC" = 159 ,                                         1a2ex
"BW" = 160 ,                                         1a2ey
"BS" = 161 ,                                         1a2ez
"LITESC" = 162 ,                                     1a2f@
"IGNORE" = 163 ,                                     1a2fa
"SC" = 164 ,                                         1a2fb
"SW" = 165 ,                                         1a2fc
"TAB" = 166 ,                                        1a2fd
"IMLAC" = 167 ,                                      1a2fe
"TI" = 168 ,                                         1a2ff
"NVT" = 169 ,                                        1a2fg
"EXECUPORT" = 170 ,                                  1a2fh
"MENU" = 171 ,                                       1a2fi
"DNLS" = 172 ,                                       1a2fj
"TNLS" = 173 ,                                       1a2fk
"COMMAND" = 174 ,                                    1a2fl
"RULE" = 175 ,                                       1a2fm
"SUBSYSTEM" = 176 ,                                  1a2fn
"DISPLAY" = 177 ,                                    1a2fo
"FROZEN" = 178 ,                                     1a2fp
"HLPCOM" = 179 ,                                     1a2fq
"PROGRAM" = 180 ,                                    1a2fr
"TERSE" = 181 ,                                      1a2fs
"INDENTING" = 182 ,                                  1a2ft
"UNIVERSAL" = 183 ,                                  1a2fu
```

```
"ENTRY" = 184 ,                                   1a2fv
"INCLUDE" = 185 ,                                 1a2fw
"BOTTOM" = 186 ,                                  1a2fx
"PAGE" = 187 ,                                    1a2fy
"OFF" = 188 ,                                     1a2fz
"FULL" = 189 ,                                    1a2g@
"PARTIAL" = 190 ,                                 1a2ga
"ANTICIPATORY" = 191 ,                            1a2gb
"DEMAND" = 192 ,                                  1a2gc
"FIXED" = 193 ,                                   1a2gd
"CONTROL" = 194 ,                                 1a2ge
"CURRENTCONTEXT" = 195 ,                          1a2gf
"FEEDBACK" = 196 ,                                1a2gg
"HERALD" = 197 ,                                  1a2gh
"PRINTOPTIONS" = 198 ,                            1a2gi
"PROMPT" = 199 ,                                  1a2gj
"RECOGNITION" = 200 ,                             1a2gk
"STARTUP" = 201 ,                                 1a2gl
"LEVELADJUST" = 202 ,                             1a2gm
"REVERSE" = 203 ,                                 1a2gn
"TEST" = 204 ,                                    1a2go
"TASKER" = 205 ,                                  1a2gp
"LINEPROCESSOR" = 206 ,                           1a2gq
"CENTER" = 207 ,                                  1a2gr
"CNTLG" = 208 ,                                   1a2gs
```

```
      "ARC" = 209 ,                                                        1a2gt

      "COPIES" = 210 ,                                                     1a2gu

      "FORMATTED" = 211 ;                                                  1a2gv

  % COMMON RULES %                                                         1b

    % ENTITY DEFINITIONS %                                                 1b1

      editentity = textent / structure;                                    1b1a

    % TEXT ENTITY DEFINITIONS %                                            1b2

      textent = text1 / "TEXT" / "LINK";                                   1b2a

      text1 = "CHARACTER" / "WORD" / "VISIBLE" / "INVISIBLE" /
      "NUMBER";                                                            1b2b

    % STRUCTURE ENTITY DEFINITIONS %                                       1b3

      structure = "STATEMENT" / notstatement;                              1b3a

      notstatement = "GROUP" / "BRANCH" / "PLEX" ;                         1b3b

  SUBSYSTEM etsmemo KEYWORD "ETSMEMO"                                      1c

    COMMAND  linsform =  "INSERT" "FORMAT" <"in file at">                  1c1

      dest _ DSEL(#"STATEMENT")                                            1c1a

      CLEAR <"Sender's ident; type <OPTION> character for login
      ident">                                                              1c1b

        ( readoption()  sent _ logid() / sent _ LSEL(#"IDENT") )           1c1b1

        cksid(sent)                                                        1c1b2

      CLEAR <"Memorandum for <separated by carriage returns>:">
      param _ LSEL(#"TEXT")                                                1c1c

      CLEAR <"cc <separated by carriage returns>:"> param2 _
      LSEL(#"TEXT")                                                        1c1d

      CLEAR <"Subject:"> param3 _ LSEL(#"TEXT")                            1c1e

      CONFIRM                                                              1c1f

      xmemo(dest,param,param2,param3) ;                                    1c1g
```

```
      END.                                                          1c2
    FINISH                                                          1d
FILE letsmemo % (L10,) <meyer,etsmemo.subsys,> %                    2
    DECLARE STRING sid[20], sfname[200] ;                           2a
    (logid) PROCEDURE % put login ident in record %                 2b
      %FORMALS%                                                     2b1
        (result,    %result record%                                 2b1a
        parsemode);   %parsing, backup, cleanup%                    2b1b
      REF result;                                                   2b2
      LOCAL TEXT POINTER lptr1, lptr2 ;                             2b3
      CASE parsemode OF                                             2b4
        = parsing:                                                  2b4a
          BEGIN                                                     2b4a1
          FIND SF(*initsr*) ^lptr1 SE(lptr1) ^lptr2 > ;             2b4a2
          result    _ lptr1;   result[1] _ lptr1[1];                2b4a3
          result[2] _ lptr2;   result[3] _ lptr2[1];                2b4a4
          END;                                                      2b4a5
        ENDCASE;                                                    2b4b
      RETURN(&result);                                              2b5
      END.                                                          2b6
    (cksid) % check sender ident %                                  2c
      PROCEDURE                                                     2c1
        %FORMALS%                                                   2c1a
          (result,  %result record%                                 2c1a1
          parsemode,        %parsing, backup, cleanup%              2c1a2
```

```
        ident); %ident%                                         2c1a3
     REF result, ident;                                          2c1b
     LOCAL TEXT POINTER cptr1, cptr2 ;                           2c1c
     LOCAL STRING idstring[100], idinfor[2000] ;                 2c1d
   CASE parsemode OF                                             2c2
     = parsing:                                                  2c2a
        BEGIN                                                    2c2a1
        dismes (1,$"Checking ident...") ;                        2c2a2
        cptr1 _    ident;    cptr1[1] _ ident[1];                2c2a3
        cptr2 _ ident[2];    cptr2[1] _ ident[3];                2c2a4
        *idstring* _ cptr1 cptr2 ;                               2c2a5
        astruc ($idstring) ;                                     2c2a6
        *sid* _ *idstring* ;                                     2c2a7
        IF NOT ckident($idstring, $idinfor, 0) THEN              2c2a8
           BEGIN                                                 2c2a8a
           dismes(1,$"Invalid ident -- retype");                 2c2a8b
           RETURN(FALSE);                                        2c2a8c
           END;                                                  2c2a8d
        getifnf($idinfor, $sfname);                              2c2a9
        dismes (0) ;                                             2c2a10
        END;                                                     2c2a11
     ENDCASE;                                                    2c2b
   RETURN(&result);                                              2c3
   END.
                                                                 2c4
```

```
(xmemo) % format memo file %                                              2d
   PROCEDURE                                                              2d1
      %FORMALS%                                                           2d1a
         (result,  %result record%                                        2d1a1
         parsemode,        %parsing, backup, cleanup%                     2d1a2
         fileptr,  %pointer to file to be formatted%                      2d1a3
         reclist,  %Receivers%                                            2d1a4
         cclist,   %cc%                                                   2d1a5
         tit);      %Title%                                               2d1a6
      REF result, fileptr, reclist, cclist, tit;                          2d1b
      LOCAL TEXT POINTER sf, ptr1, ptr2, ptr3, ptr4;                      2d1c
      LOCAL STRING ccstr[400], titstr[1000], tempstr[1000] ;              2d1d
   CASE parsemode OF                                                      2d2
      = parsing:                                                          2d2a
         BEGIN                                                            2d2a1
         %set up for recreate display%                                    2d2a2
            dpset (dspallf, endfil, endfil, endfil) ;                     2d2a2a
         %put directives in origin%                                       2d2a3
            sf _ fileptr ;                                                2d2a3a
            sf.stpsid _ origin;                                           2d2a3b
            FIND SF(sf) [" ;;;;"/ENDCHR] ^sf;                             2d2a3c
               ST sf sf _ "             " ;                               2d2a
```

--------------------



```
%add to first statement %                                          2d2
   IF (sf := getsub(sf))=sf THEN RETURN(&result);                  2d2a
   sf[1] _ 1;                                                      2d2a
   %receivers%                                                     2d2a
      ptr1 _    reclist;  ptr1[1] _ reclist[1];                    2d2a4
      ptr2 _ reclist[2];  ptr2[1] _ reclist[3];                    2d2a4
   %cc%                                                            2d2a
      ptr3 _    cclist;  ptr3[1] _ cclist[1];                      2d2a4
      ptr4 _ cclist[2];  ptr4[1] _ cclist[3];                      2d2a4
      *ccstr* _ ptr3 ptr4 ;                                        2d2a4
      IF ccstr.L THEN *ccstr* _ "
      ", EOL, "                          cc:  ", *ccstr*,
      "" ;                                                         2d2a4
   %title%                                                         2d2a
      ptr3 _    tit;  ptr3[1] _ tit[1];                            2d2a4
      ptr4 _ tit[2];  ptr4[1] _ tit[3];                            2d2a4
      *titstr* _ ptr3 ptr4 ;                                       2d2a4
      IF titstr.L THEN *titstr* _ "Subject:  ",
      *titstr* ELSE *titstr* _ SP ;                                2d2a4
   %title, date, author%                                           2d2a
      FIND SF(*titstr*) $28(CH) < ([NP] $NP/TRUE) ^ptr3
      > $NP ^ptr4;                                                 2d2a4
      *tempstr* _ "
      ", SF(ptr3) ptr3, "  Date:  19 MAY 75", EOL ;                2d2a4
      ST ptr4 _ ptr4 SE(ptr4) ;                                    2d2a4
      IF FIND SF(*titstr*) 1$28(CH) < ([NP] $NP/TRUE)
      ^ptr3 > $NP ^ptr4 THEN                                       2d2a4
```

```
        BEGIN                                                    2d2a4f
        *tempstr* _ *tempstr*, SF(ptr3) ptr3 ;                   2d2a4f
        ST ptr4 _ ptr4 SE(ptr4) ;                                2d2a4f
        END;                                                     2d2a4f
      *tempstr* _ *tempstr*, "From:  ", *sfname*, "",
      EOL  ;                                                     2d2a4
      LOOP IF FIND SF(*titstr*) 1$2$8(CH) < ([NP]
      $NP/TRUE) ^ptr3 > $NP ^ptr4                                2d2a4
        THEN                                                     2d2a4f
          BEGIN                                                  2d2a4f6
          *tempstr* _ *tempstr*, SF(ptr3) ptr3, EOL ;
                                                                 2d2a4f6
          ST ptr4 _ ptr4 SE(ptr4) ;                              2d2a4f6
          IF titstr.L < 25 THEN                                  2d2a4f6
            BEGIN                                                2d2a4f6a
            *tempstr* _ *tempstr*, *titstr* ;                    2d2a4f6a
            EXIT LOOP ;                                          2d2a4f6a
            END ;                                                2d2a4f6a
          END                                                    2d2a4f6
        ELSE EXIT LOOP ;                                         2d2a4f
    ST sf sf _ "              Memorandum for:  ", +ptr1
    ptr2, "", *ccstr*, EOL, *tempstr*, "", EOL ;                 2d2a
  % put directives in last statement %                          2d2
```

```
            IF ( sf := getend(ptr3_getail(sf)) ) # sf THEN ST sf
            sf _ "" ;                                                   2d2a
        % insert clerk statement if necessary %                         2d2
            IF *initsr* # *sid* THEN                                    2d2a
              BEGIN                                                     2d2a6
              *tempstr* _ "

              ", *sid*, '/, -SF(*initsr*) SE(*initsr*) ;
              FIND SF(*tempstr*) ^ptr1 SE(*tempstr*) ^ptr2 > ;
              cinssta (ptr3, 0, $ptr1, $ptr2);
              END;
        END;
      ENDCASE;
    RETURN(&result);
    END.
  FINISH
```

ETS Memo Formatting Program

(J32547) 19-MAY-75 13:37;;;; Title: Author(s): N. Dean Meyer/NDM; Distribution: /DAP( [ INFO-ONLY ] ) JCN( [ INFO-ONLY ] ) FEED( [ INFO-ONLY ] ) ; Sub-Collections: SRI-ARC; Clerk: NDM; Origin: < MEYER, ETSMEMO.NLS;7, >, 9-MAY-75 14:22 NDM ;;;;####;

Teleconferencing and Computer Communication

Contact Report
Bert Liffman
General Conferencing Systems, Ltd.
May 15 1975 1

(416) 961-3680
General Conferencing Systems, Ltd.
4 Maitland St.
Toronto, Ontario
Canada M4Y 1C5 2

This contact report was made by Ra3y Panko (RA3Y) as part of an SRI project on teleconferencing. 3

General Conferencing Systems, Ltd. is the new name of Memo from Turner, which created General Conferencing System (GCS), the computer teleconferencing system on the I.P. Sharpe timesharing network in Canada and the United States. Historically, GCS was inspired by Murray Turoff's work, and Turoff is still a consultant to GCS, Ltd. 4

One organization is now using GCS. This is the Nonmedical Drugs Directorate. This is the Canadian Federal Government's agency for alcohol and drugs. The NMDD has 92 employees. Sixty-five percent are in Ottowa, the rest in field offices in the Maritime, Montreal, Ontario, Winnepeg, and Vancouver. The NMDD is a small organization that functions primarily on the basis of its ability to influence localite agencies. Communication has traditionally been a problem within the NMDD, because so many staff members are in field offices and are usually working outside the office, in the community. Also, the NMDD has been going through a reorgainzation based upon a matrix management scheme. 5

NMDD began using GCS in November 1974. There are now 6 terminals in the Ottowa office, 2 in each field office. These terminals are portable and are frequently taken along on trips. There are also two consultants in the United States, one in Washington D.C., the other in Newark New Jersey (Murray Turoff); both use the system. 6

The I.P. Sharpe system charges about $44 per connect hour, but in July the rate should fall to $25 per connect hour. The NMDD has averaged about 200 connect hours each month -- $8,800 per month at the current rate, $5,000 at the projected rate. For the traffic rates discussed below, this comes to about $3.00 per message at the current rate, $1.30 per message at the new rate. 7

The NMDD has conducted 10 conferences, all deaing with reorganization. Conferences averaged only about 83 public statements apiece. The average entry averaged 150 to 200 words. Many entries were position papers for discussion; this may inflate the average length. 8

The NMDD's real use of GCS has been private message transmission. GCS allows both private messages and public "statements" to be sent, the latter associated with a specific conference. In the six months of operation to date, about 20,000 messages have been sent among the 60 NMDD members who use the system. This averages about 50 messages, per person, per month. These figures do not include multiple copies. Messages average 100 to 150 words. 9

In the NMDD's use of GCS, we have a real-world organization using a system that offers both teleconferencing and mail box service. It may be significant that teleconferencing accounts for only 5 percent of the total system use. It may be that in computer-based human communications, conferences will serve the same purposes they do in normal organizational communication; they may be brief but important episodes in the total process. 10

GCS uses an IBM 360-75. It is programmed in the I.P. Sharpe version of APL. 11

Teleconferencing and Computer Communication

(J32549) 19-MAY-75 16:57;;;; Title: Author(s): Raymond R. Panko/RA3Y; Distribution: /KIRK( [ INFO-ONLY ] ) ; Sub-Collections: SRI-ARC; Clerk: RA3Y; Origin: < PANKO, GCS.NLS;1, >, 15-MAY-75 15:38 RA3Y ;;;;####;

MEJ 19-MAY-75 18:15 32558

Try Again - Number Foul-up

This is the second try after an initial foul-up; the SEND command fedback the error message "number reserved for someone else!". I have a sneaking hunch that is in no way near the truth! 1

Try Again - Number Foul-up

(J32558) 19-MAY-75 18:15;;;; Title: (Expedite) Title: Author(s): Mil E. Jernigan/MEJ; Distribution: /MEJ( [ ACTION ] ) ; Keywords: NSA Scenario; Sub-Collections: NSA NIC; Clerk: MEJ;

Etsmemo

Memorandum for: MR. MEYER
MR. TURNBULL

cc: ms. lockheed

Subject: ets memo formats

Date: 20 MAY 75
From: David A. Potter

This is the memo.

JMB 20-MAY-75 07:03 32562

Coffee: <25880,><25881,>

I think that the individuals who drink coffee should pay for the coffee; I don't think this should be an ARC expense (petty cash or whatever). 1

Coffee: <25880,><25881,>

(J32562) 20-MAY-75 07:03;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /SRI-ARC( [ INFO-ONLY ] ) ; Sub-Collections: SRI-ARC;
Clerk: JMB;

MOA with ESD for P5550 Security Task

This is the draft MOA I was suppposed to write a long while ago. Roc, Rzepka, and Mac have reviewed it. Please review it and comment so that I can forward to IS for review and comment.

MEMORANDUM OF AGREEMENT 1

between the 2

ROME AIR DEVELOPMENT CENTER 3

and the 4

ELECTRONIC SYSTEMS DIVISION 5

I. PURPOSE: The purpose this Memorandum of Agreement (MOA) is to define the relationship between the Rome Air Development Center (RADC) and the Electronics System Division (ESD) in regards to Project 5550 - Advanced Computer Technology, Task 08 - ADP System Security. 6

II. TERMS OF AGREEMENT: 7

A. Effective Date: This agreement is effective upon the approval by the Commander, Rome Air Development Center (RADC/CC) and the Electronic Systems Division, Deputy for Command and Management Systems (ESD/MC). It shall continue in effect throughout the period for which the task is a part of Project 5550 (which is expected to be to 30 June 1976) unless changed, superseded, or terminated. Requests for rescission, revision or change to this MOA must be mutually acceptable of the signatories and will be sent to RADC/IS for consolidation. Cancellation may be by either party upon written notification to the other party. The effective date appears at the top of the top right of this page. 7a

B. Offices of Prime Responsibility: The Rome Air Development Center, Information Science Division (RADC/IS) is the RADC OPR for this MOA. The Electronic Systems Division, Deputy forCommand and Management Systems, Information Systems Technology Applications Office (ESD/MCI)) is the ESD OPR for this MOA. points of contact are Mr. Roger B. Panara, RADC/ISIM and Major Roger Schell, ESD/MCI as focal points for communications and inquiries. 7b

C. Correspondence: All routine working type correspondence shall be addressed directly to the OPR within each organization. All non-routine correspondence (policy questions, position papers, controversial information, etc.) shall be addressed to RADC/CC for items sent to the Rome Air Development Center and to ESD/MC for items sent to the ESD. 7c

D. Funding: Funding for efforts to be conducted by ESD in support of this MOA will be forwarded from RADC by Procurement Directive. RADC/IS efforts to be conducted under this task will be funded directly from the Project 5550 budget authorization and be reported as work units under the task. Funding documents will be processed through normal channels. 7d

III. POLICY: The undersigned desire that the personnel of each organization work together efficiently, effectively, and productively in support of the program. It is intended that this agreement promote a spirit of cooperation and mutual understanding on the part of all RADC and ESD personnel. 8

IV. SCOPE OF RADC/IS RESPONSIBILITIES: RADC/IS, as Project 5550 Director will perform project duties as outlined in the Program Management Plan. Further, it will provide the following specialized support to ESD/MC: 9

A. Develop secure applications under the executive. 9a

B. Attend meetings and participate in tecnical evaluations, as required. 9b

C. Provide technical information to the contractor(s) as requested by ESD/MCI. 9c

D. Review and comment on computer security test plans, directives, procedures, schedules and test specifications as required. 9d

E. Provide technical assistance in support of design reviews. 9e

F. When it is not possible for RADC to accomplish upon request any support function required by this agreement, RADC will advise ESD/MCI immediately and priorities will be established. 9f

V. SCOPE OF ESD/MCI RESPONSIBILITIES: ESD/MCI has total technical responsibility and authority for all aspects of the ADP System Security task and will: 10

A. Keep RADC advised of all significant program meetings which involve the area of computer security and for which assistance is requested. 10a

B. Provide RADC with all necessary contractual data items and other data necessary for the proper performance of its functions as project director as outlined in the Program Management Plan. 10b

C. Provide higher headquarters with briefings on the task as requested by the Project Director. 10c

VI. COORDINATION: Both RADC/IS and ESD/MC have a responsibility to keep the other informed in a timely manner on program status, progress, and changes as they occur, particularly in those areas which may have a bearing or influence on the other party's exercise of its mission. The RADC/IS and ESD/MC will coordinate with each other on an "as required" basis. Visits or direct verbal communication to contractors of either organization will be coordinated in advance through the applicable OPR. 11

VII. SIGNATURE: This memorandum shall take effect when signed by both parties. 12

MOA with ESD for P5550 Security Task

(J32563) 20-MAY-75 07:35;;;; Title: Author(s): Roger B. Panara/RBP;
Distribution: /FJT( [ ACTION ] ) JLM( [ INFO-ONLY ] ) ; Sub-Collections:
RADC; Clerk: RBP; Origin: < PANARA, MOA.NLS;1, >, 20-MAY-75
07:26 RBP ;;;; ####;