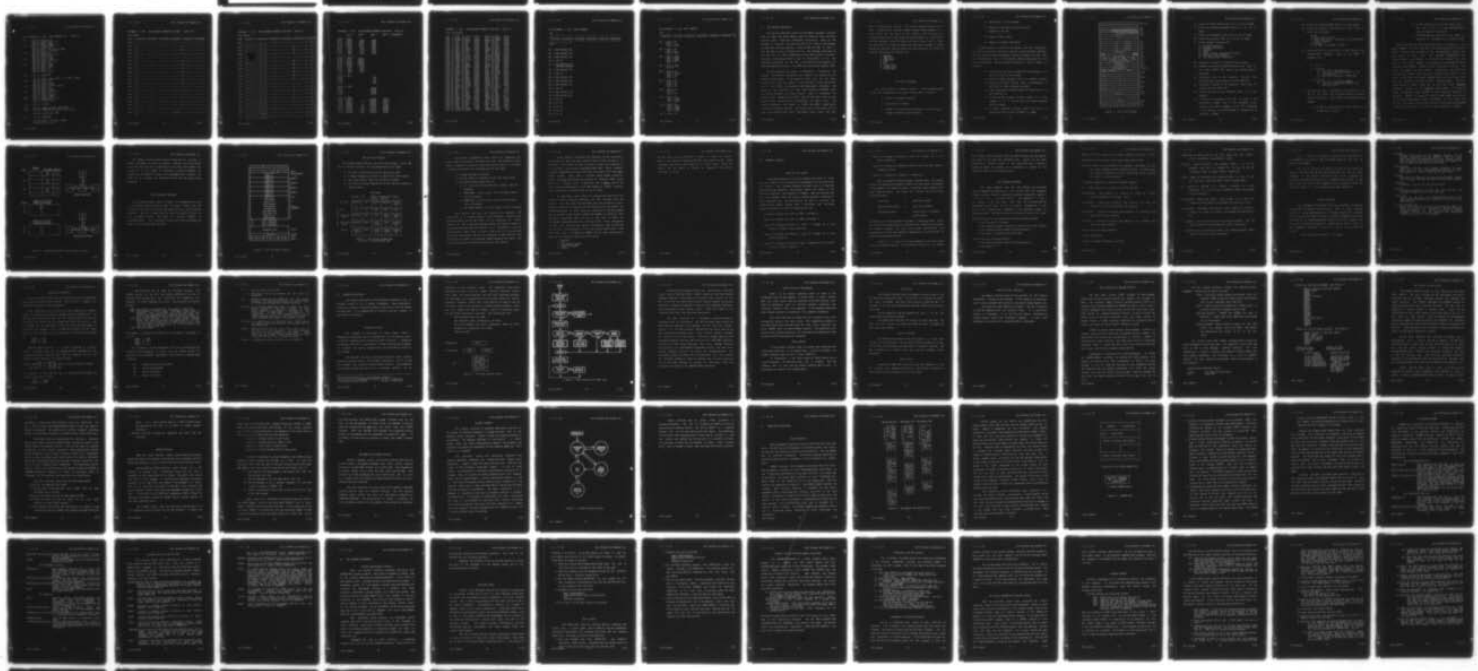
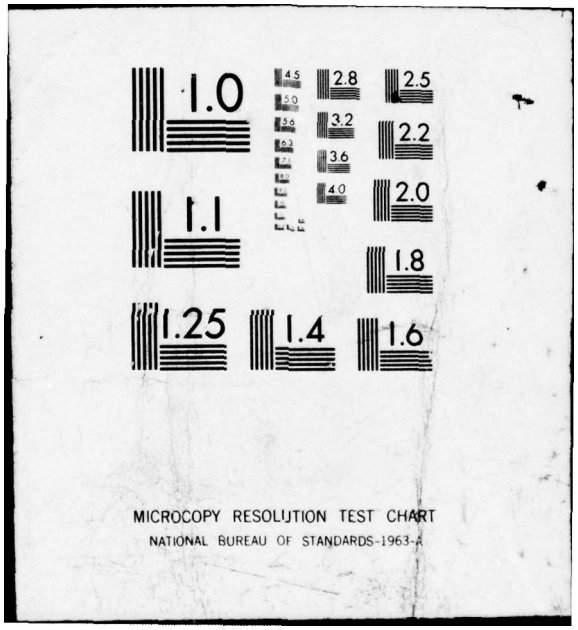
AD-A033 351 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MASS F/G 9/2
THE ARPANET NETWORK CONTROL CENTER PROGRAM.(U)
NOV 76 DAHC15-69-C-0179
UNCLASSIFIED TECHNICAL INFORMATION-90 NL

| OF |
AD
A033351



END

DATE
FILMED
2-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(15) DAHC15-69-C-0179,
F08606-73-C-0027

Bolt Beranek and Newman Inc.

(14) Technical Information - 90

Technical Information Report No. 90

(6)

THE ARPANET NETWORK CONTROL CENTER PROGRAM

(11) November 1976

(12) 87p.

060100

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

This report describes the NETWORK CONTROL CENTER (NCC) program used in the ARPA Network. The information in the report will explain the functions, structure, and major routines of the NCC program in detail, with diagrams included where helpful. Anyone who is interested in packet-switched networks can read and understand this INTRODUCTION. Network operators and programmers may be most interested in the sections on IMP MESSAGE PROCESSING and OPERATOR CONTROLS. The last sections are meant for programmers who need detailed knowledge of the program.

Background - HOSTS, IMPS, and the NCC

The ARPA Network provides a capability for geographically separated computers, called Hosts, to communicate with each other via common-carrier circuits. Each Host is connected to a small local computer called an Interface Message Processor (IMP); each IMP is connected to several other IMPs via wideband communications lines. The IMPs, most of which are virtually identical, are programmed to store and forward messages to their neighbor IMPs based on address information contained in each message.

In a typical network operation, a Host passes a message, including a destination address, to its local IMP. The message is

passed from IMP to IMP through the network until it finally arrives at the destination IMP, which in turn passes it along to the destination Host. An important aspect of this operation is that the path the message will traverse is not determined in advance; rather, an IMP forwards each message on the path it determines to be best, based on its current estimate of local network delay. Since the path choices are determined dynamically, IMPs can take account of circuit or computer loading (or failures) to insure prompt delivery of each message.

An interesting aspect of an IMP subnetwork (i.e., the set of IMPs and communication lines) is that it can be considered a distributed computation system. Each IMP performs its own tasks relatively independent of its neighbor IMPs; nevertheless, all IMPs are cooperating to achieve a single goal, reliable Host-Host communication. In some cases, for example the dynamic path selection mentioned above, each IMP cooperates with its neighbors in making reliable delay estimates for various path choices.

In any distributed computation system, it is likely to be difficult to detect component failures quickly; the difficulty is increased in the IMP subnetwork by the geographic separation of components. For this reason, it was decided to incorporate automatic reporting functions in the IMPs as an aid to failure diagnosis. Each IMP is programmed to examine itself and its environment periodically and to report the results of these

examinations to a central mediating agent. This agent has the function of collecting the (possibly conflicting) IMP reports, determining the most likely actual state of the network, and reporting on any significant change in the network. The mediating agent is the NCC computer - a Host computer which receives the IMP messages.

It is important to mention some jobs that the NCC Host is not designed to do. It does not initiate repair of broken hardware or software. It provides printouts of network changes to people who have the judgment and experience to decide what repairs to perform (if any) and in what order. The NCC Host, although aware of the network topology, does not take part in decisions about the paths and delays of the IMP subnetwork. The NCC Host computer must not be confused with a "network control center" manned by people providing centralized hardware and software support to meet the needs of the Host computers.

Finally, it should be noted that although the NCC computer is an important component of the network, its operation is not essential to that of the network. As with any other Host, it can fail without disturbing overall network integrity.

NCC Host Functions

The NCC Host has two general jobs: it is a tool for diagnosing network problems and it is a network data collector.

As a tool for diagnosing network problems, the primary NCC job is to use the periodic IMP messages (plus the internal clock timeouts) to discover changes. If the changes are major, the NCC Host will sound an alarm and flash a light, to insure that the change is noticed quickly. All changes, whether major or minor, are presented as a human-engineered printout, called the LOG. The LOG reports not only complete failures of a line or an IMP, but also information about unusual events which often precede complete failure (e.g., a noisy line may report a low error rate as some hardware component wears out). The IMP programs also detect many conditions which are not normal, but recoverable, and sends special "trap" messages that appear on the LOG.

The NCC Host is aware of the network topology. Unlike each IMP, which knows only how many "hops" are necessary to reach another IMP, the NCC Host reports about "lines" to the network support staff. Each IMP sends the NCC a snapshot of its environment. The NCC integrates the snapshots into a single picture of the network as a whole, and presents the larger view with lights, alarms, and a LOG printout. This knowledge of topology is occasionally of great importance; for instance, if the network ever partitions, isolating several sites, the NCC can compute the "frontier" and flash those

lights while turning off the lights for the "invisible" sites and lines.

As a network data collector, the NCC Host saves some of the raw IMP data in core memory. Host and line throughput data is collected from each IMP and summarized on a terminal hourly. The status of each IMP and line is saved every 15 minutes; a status summary printout is produced every 8 hours. The compacted encoding of the LOG printout builds up in memory. This data may be shipped to other Hosts upon request.

The data collection abilities work in conjunction with processes running on Tenex (F[illegible]-10) computers at other network Host sites. The full data collection is a distributed computation. Other network Hosts have reliable bulk storage and general-purpose languages - the NCC Host is a mini-computer without much storage or power. The two kinds of machines work together to get interesting data to the proper place for analysis. Typically, the NCC data is used for "batch" rather than "real-time" computation on other Hosts.

NCC Hardware and I/O Devices

The NCC Host looks like a Terminal IMP (TIP). It is a Honeywell 316 with a Multi-Line Controller (MLC) added. It has 28K words of core memory, an IMP interface, a light-display/alarm, and several terminals. It has a paper tape reader and a modem interface for program loading. It does not need special instructions (for instance, multiply and divide), disks, or other desirable options; the hardware is almost identical to the BBN-TESTIP. In case of serious hardware failure, the BBN-TESTIP may be substituted for the NCC Host with a minimum of effort. The I/O devices and their uses are described briefly below:

1) IMP Interface - The NCC's IMP interface is identical to the IMP's Host interface; the interfaces are connected by a cable which juggles the IMP and Host definitions appropriately. All network data, whether IMP or Host traffic, uses this path.

2) Primary terminal - This Model 35 Teletype has two major functions. It prints periodic reports of Host and line throughput and status. Secondly, it is used to change NCC parameters, and as a debugging aid, much as the IMP Teletype and DDT code are used for IMP control and debugging.

3) Lightbox & Alarm - A BBN-designed I/O device with 64 LED lights, 16 switches, and an audible buzzer; this is the primary warning of important network changes. Since there

is only one such device, the program can run without it, although the support staff would have to keep a careful eye on the LOG printout if the lightbox/alarm were broken.

4) LOG terminals - The LOG produces a hard copy record of network events. Normally two small lineprinters are used: a Centronics 306 is next to the NCC Host, and a quiet Inktronics is in the manned area of the Network Control Center. Both run at 1200 baud, and the Centronics needs padding characters after RETURN and LINEFEED.

5) MSG-DUMP terminal - One terminal is dedicated to printing an octal dump of any network message which comes from an IMP (as opposed to a Host) but is not a recognized status or throughput message. This facility is a flexible IMP debugging aid, particularly for viewing "bad" packets.

6) TT-EXER terminals - Other MLC ports may be used to exercise terminals by continuously printing a simple pattern. This is an aid to hardware maintenence of the LOG and MSG-DUMP devices.

Sample Printouts

The following few pages contain sample printouts. The first page shows the LOG printout similar to the lineprinter output in the Network Control Center. The remaining pages of printout come from the primary TTY, often referred to as the "summary TTY". The IMP and line status printouts are compressed, with the following 9 single characters used to indicate different states:

?	Unknown
+	Down in the plus direction (lines only)
-	Down in the minus direction (lines only)
*	Down
.	Up
>	Plus end looped (lines only)
<	Minus end looped (lines only)
Z	Both ends looped (lines only)
X	Status changed one or more times during the interval

The Host throughput data is divided into 8 categories as follows:

M->N Messages to Hosts at other nodes.
M<-N Messages from Hosts at other nodes.
P->N Packets to Hosts at other nodes.
P<-N Packets from Hosts at other nodes.
M->S Messages to Hosts at this node.
M<-S Messages from Hosts at this node.
P->S Packets to Hosts at this node.
P<-S Packets from Hosts at this node.

Every 8 hours, the IMP status, line status, Host throughput, and line throughput reports are printed, summarizing the period since midnight. Every hour, the Host and line throughput reports summarize the previous hour. The "quick summary" and "host summary" reports print only at the operator's command.

```
1100  NOVEMBER 13 1976   ARPA NETWORK LOG    PAGE 99

1100      IMP 60: *(102.,4,0)
         LINE 41: ERRORS PLUS 2/16
          IMP 58: HOST 1 DOWN
1101     LINE 41: ERRORS MINUS 1/17
          IMP 61: *(102.,0,0)
          IMP 58: HOST 1 UP, HOST 1 DOWN
          IMP 60: *(102.,1,177771)
1102      IMP 58: HOST 1 UP
         LINE 41: ERRORS MINUS 1/17
          IMP 61: *(102.,0,0)
1103      IMP 58: HOST 1 DOWN
          IMP 60: *(102.,177775,0)
          IMP 58: HOST 1 UP
1104      IMP 58: HOST 1 DOWN
          IMP 45: HOST 1 DOWN
          IMP 58: HOST 1 UP
          IMP 45: HOST 1 UP
          IMP 61: *(102.,0,0)

1105     LINE 41: DOWN ***

          IMP 58: HOST 1 DOWN, HOST 1 UP, HOST 1 DOWN
1106      IMP 58: HOST 1 UP
          IMP 61: *(102.,0,0)
          IMP 58: HOST 1 DOWN
1107      IMP 60: *(102.,177777,177776)
          IMP 58: HOST 1 UP
          IMP 58: HOST 1 DOWN
1108      IMP 39: TRAP: (7706,2,1)
          IMP  6: HOST 0 UP
          IMP 11: *(32.,104647,0)
1109      IMP 11: HOST 0 DOWN
1115     LINE 41: ERRORS PLUS 1/16
          IMP 41: *(63.,400,47)

1116     LINE 41: UP ***

1117      IMP 30: POWER FAILURE, RESTARTED
1118      IMP 30: OVERRIDE ON, TRAP: (23456,0,36)

         LINE 88: LOOPED PLUS ***

1120      IMP 63: DOWN ***

          IMP 58: HOST 1 UP, HOST 1 DOWN
          IMP 23: *(311.,11,0)
```

NOVEMBER 4 1976 ARPA NETWORK SUMMARY 0000-0800 PAGE 148

IMP STATUS

```
TIME  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890
0000  .......... .......... .......... .......... .......... ..........
      .?.
0030  .......... .......... .......... .......... X......... ..........
      .?.
0045  .......... .......... .......... .......... *......... ..........
      .?.
0100  .......... .......... .......... .......... X......... ..........
      .?.
0115  .......... .......... .......... .......... .......... ..........
      .?.
0130  .......... .......... .......... .......... X......... ..........
      .?.
0145  .......... .......... .......... .......... X......... ..........
      .?.
0200  .......... .......... .......... .......... *......... ..........
      .?.
0300  .......... .......... .......... .......... X......... ..........
      .?.
0315  .......... .......... .......... .......... X......... ..........
      .?.
0330  .......... .......... .......... .......... *......... ..........
      .?.
0345  .......... .......... .......... .......... X......... ..........
      .?.
0400  .......... .......... .......... .......... .......... ..........
      .?.
0500  .......... .......... .......... .......... X......... ..........
      .?.
0515  .......... .......... .......... .......... .......... ..........
      .?.
0530  .......... .......... .......... .......... X......... ..........
      .?.
0545  .......... .......... .......... .......... .......... ..........
      .?.
0600  .......... .......... .......... .......... X......... ..........
      .?.
0615  .......... .......... .......... .......... .......... ..........
      .?.
0745  .......... .......... .......... .....X.X.. .......... ..........
      .?.
```

NOVEMBER 4 1976 ARPA NETWORK SUMMARY 0000-0800 PAGE 150

LINE STATUS

```
TIME  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890
0000  .......... .......... .....?.... .......... .......... .?....*...
      .?........ ...?...<<.
0030  .......... .......... .....?.... .......... X......... .?....*...
      .?........ ...?...<<.
0045  .......... .......... .....?.... .......... *X........ .?....*...
      .?........ ...?...<<.
0100  .......... .......... .....?.... .......... XX........ .?....*...
      .?........ ...?...<<.
0115  .......... .......... .....?.... .......... .......... .?....*...
      .?........ ...?...<<.
0130  .......... .......... .....?.... .......... XX........ .?....*...
      .?........ ...?...<<.
0200  .......... .......... .....?.... .......... **........ .?....*...
      .?........ ...?...<<.
0300  .......... .......... .....?.... .......... XX........ .?....*...
      .?........ ...?...<<.
0315  .......... .......... .....?.... .......... XX........ .?....*...
      .?........ ...?...<<.
0330  .......... .......... .....?.... .......... **........ .?....*...
      .?........ ...?...<<.
0345  .......... .......... .....?.... .......... XX.......X .?....*...
      .?........ ...?...<<.
0400  .......... .......... .....?.... .......... .......... .?....*...
      .?........ ...?...<<.
0500  .......... .......... .....?.... .......... X......... .?....*...
      .?........ ...?...<<.
0515  .......... .......... .....?.... .......... X......... .?....*...
      .?........ ...?...<<.
0530  .......... .......... .....?.... .......... X......... .?....*...
      .?........ ...?...<<.
0545  .......... .......... .....?.... .......... X......... .?....*...
      .?........ ...?...<<.
0600  .......... .......... .....?.... .......... X......... .?....*...
      .?........ ...?...<<.
0615  .......... .......... .....?.... .......... X......... .?....*...
      .?........ ...?...<<.
0630  .......... .......... .....?.... .......... .......... .?....*...
      .?........ ...?...<<.
0700  .......... .......... .....?.... .......... .......... .?....*...
      .?........ ...?.X.X<.
0715  .......... .......... .....?.... .......... .......... .?....*...
      .?........ ...?...<<.
```

NOVEMBER 4 1976 ARPA NETWORK SUMMARY 0000-0800 PAGE 152

	HOST 0	HOST 1	HOST 2	HOST 3	THROUGHPUT
SITE 1 UCLA					
M->N	23778	7383	2489		
M<-N	26460	10963	3284		
P->N	23778	12117	2489		
P<-N	27937	10976	3284		
M->S	4667	1152	1757		
M<-S	1877	1137	4562		
P->S	4874	1152	2742		
P<-S	2862	1344	4562		
SITE 2 SRI-5					
M->N	61632	108480			
M<-N	67072	124160			
P->N	61632	125696			
P<-N	84864	132416			
M->S	2878	5200			
M<-S	4311	3763			
P->S	2877	6810			
P<-S	5870	3814			
SITE 3 NUC					
M->N			539		
M<-N			721		
P->N			539		
P<-N			956		
SITE 4 UTAH					
M->N			52928		
M<-N			36608		
P->N			52928		
P<-N			38336		
SITE 5 BBN					
M->N	134400		385856	13979	
M<-N	172160	18	335488	13911	
P->N	148672		385856	21815	
P<-N	177856	18	350016	14062	
M->S	129600	513	217152	31506	
M<-S	177344	481	147904	53056	
P->S	143168	513	217472	34624	
P<-S	177792	2036	162816	53120	

NOVEMBER 4 1976 ARPA NETWORK SUMMARY 1600-1700 PAGE 177

LINE THROUGHPUT

LINE	1:	PENT	TO EGLIN	22612	EGLIN	TO PENT	23768
LINE	2:	M-MAC	TO CCA	41216	CCA	TO M-MAC	38144
LINE	3:	M-MAC	TO WRPAT	23046	WRPAT	TO M-MAC	23166
LINE	4:	USC	TO DOCB	30635	DOCB	TO USC	30962
LINE	5:	UCLA	TO USC	13773	USC	TO UCLA	14089
LINE	6:	UCLA	TO NUC	16267	NUC	TO UCLA	16351
LINE	7:	UCLA	TO SCRL	14307	SCRL	TO UCLA	13909
LINE	8:	SRI-5	TO XEROX	46592	XEROX	TO SRI-5	46144
LINE	9:	BELV	TO ABER	17052	ABER	TO BELV	17080
LINE	10:	SRI-5	TO LLL	9630	LLL	TO SRI-5	10199
LINE	11:	AMEST	TO SRI-3	22871	SRI-3	TO AMEST	21604
LINE	12:	STAN	TO ISI22	85312	ISI22	TO STAN	86208
LINE	13:	HARV	TO NCC-T	55168	NCC-T	TO HARV	49344
LINE	14:	M-MAC	TO M-TIP	40384	M-TIP	TO M-MAC	33088
LINE	15:	UTAH	TO ILL	15332	ILL	TO UTAH	14837
LINE	16:	LINC	TO ROME	24887	ROME	TO LINC	16972
LINE	17:	CARN	TO ARGON	52352	ARGON	TO CARN	52352
LINE	18:	CARN	TO BELV	13848	BELV	TO CARN	12871
LINE	19:	HARV	TO NYU	28206	NYU	TO HARV	29017
LINE	20:	STAN	TO AMES	60352	AMES	TO STAN	60992
LINE	21:	MITRE	TO ARPA	19257	ARPA	TO MITRE	19130
LINE	22:	CARN	TO ROME	39680	ROME	TO CARN	40576
LINE	23:	NBS	TO NSA	12585	NSA	TO NBS	13212
LINE	24:	AMES	TO AMEST	39872	AMEST	TO AMES	39040
LINE	25:	ISI22	TO KIRT	79680	KIRT	TO ISI22	81280
LINE	27:	GWC	TO ARGON	29503	ARGON	TO GWC	29941
LINE	28:	DCEC	TO SDACP	38912	SDACP	TO DCEC	43584
LINE	29:	MITRE	TO SDACP	38464	SDACP	TO MITRE	33280
LINE	30:	BBN T	TO NCC-T	1837	NCC-T	TO BBN T	165
LINE	31:	BBN	TO CCA	43904	CCA	TO BBN	43072
LINE	32:	GWC	TO DOCB	32896	DOCB	TO GWC	32727
LINE	33:	PENT	TO ARPA	7842	ARPA	TO PENT	7731
LINE	34:	NBS	TO ABER	31350	ABER	TO NBS	29117
LINE	35:	XEROX	TO TYMSH	50240	TYMSH	TO XEROX	50752
LINE	36:	RAND	TO ISI52	38464	ISI52	TO RAND	38656
LINE	37:	FNWC	TO SCRL	13787	SCRL	TO FNWC	14464
LINE	38:	RAND	TO NELC	32768	NELC	TO RAND	31974
LINE	39:	AMES	TO HAW T	17047	HAW T	TO AMES	16946
LINE	40:	UTAH	TO LBL	27488	LBL	TO UTAH	27827
LINE	41:	SDACP	TO NORSR	15806	NORSR	TO SDACP	17167
LINE	42:	NORSR	TO ULOND	652	ULOND	TO NORSR	907

```
1502 NOVEMBER 10 1976  QUICK SUMMARY

 IMP
 .......*.. .......... .......... .......?.. .......... ..........
 .?.
 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890
 .......... .......... .....?...? .......... .......... .?........
 .?........ ...?...<<.
 LINE

IMP  1 MEM PROTECT OFF

IMP  2 MEM PROTECT OFF

IMP  3 MEM PROTECT OFF

IMP  4 TIP UP

IMP  5 MEM PROTECT OFF
       OVERRIDE ENABLED

IMP  6 MEM PROTECT OFF

IMP  7 TIP UP

IMP 10 MEM PROTECT OFF

IMP 11 MEM PROTECT OFF

IMP 12 MEM PROTECT OFF

IMP 13 TIP UP

IMP 14 MEM PROTECT OFF

IMP 15 MEM PROTECT OFF

IMP 16 TIP UP

IMP 17 TIP UP

IMP 18 TIP UP

IMP 19 TIP UP

IMP 20 TIP UP
```

```
1507 NOVEMBER 10 1976  HOST SUMMARY

 IMP
 .......*.. .......... .......... .......?.. .......... .......... .?.
 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 123

IMP  1 HOST 0 UP
       HOST 1 UP
       HOST 2 DOWN

IMP  2 HOST 0 UP
       HOST 1 UP
       HOST 2 DOWN

IMP  3 HOST 0 DOWN
       HOST 1 DOWN
       HOST 2 DOWN
       HOST 3 UP

IMP  4 HOST 0 DOWN
       TIP UP

IMP  5 HOST 0 UP
       HOST 1 TARDY
       HOST 2 UP
       HOST 3 UP

IMP  6 HOST 0 UP
       HOST 1 UP
       HOST 2 UP
       HOST 3 UP

IMP  7 HOST 0 UP
       TIP UP
       HOST 3 UP

IMP  9 HOST 0 UP
       HOST 1 TARDY
       HOST 3 DOWN

IMP 10 HOST 0 UP
       HOST 1 DOWN
       HOST 2 DOWN
       HOST 3 DOWN

IMP 11 HOST 0 UP
```

II. IMP MESSAGE PROCESSING

This section describes status and throughput messages received from all IMPs, the CLOCK timeout logic for IMPs and lines, and some concerns about reporting reasonable conclusions. When an IMP message is received, we know that IMP is UP. When an IMP message has not been received recently, it may be that the IMP is down or that the IMP is up but isolated from us. The topology table kept by the NCC can be used to determine if the IMP is down or isolated. Similarly, lines may go down or may be "invisible" to us. The topology maintained by the NCC is ordered by "line numbers" for human-engineered printouts - the IMPs know only about neighbor IMPs.

The NCC maintains two types of statistics: utilization and status. Utilization statistics are stored as a single word per statistic as follows: if the sign bit is not set, the low order fifteen bits are the value; if all the bits in the word are set (i.e., -1 or 177777), the statistic has overflowed; otherwise, the value of the statistic is (low order fifteen bits)*2**6. Each utilization statistic is stored thrice: once to accumulate the current hour (added into by BTR), once for the final total for the previous hour, and the third to accumulate the daily total. The status statistics are stored in IMPST and LINST. These tables consist of 96 6-word entries. The first entry is the status for the most recent quarter hour; each succeeding entry covers the next

prior fifteen-minute period. The entries beyond midnight are not used. In particular, the most current network status remains in a fixed place (at the head of the tables), and the location of midnight moves down in the tables as the day wears on. Each word of an entry contains a four-bit status for each of four lines or IMPs. The high order bit being set means that there was a change in status during the interval, and the three low order bits give the final (or current) status for the interval by:

0 UNKNOWN
1 DOWN PLUS
2 DOWN MINUS
3 DOWN
4 UP
5 LOOPED PLUS
6 LOOPED MINUS
7 LOOPED BOTH

IMP Status Messages

The Status Report is shown in Figure 1. The following events trigger the transmission of a report from an IMP to the NCC:

1) End of a 52-second interval

2) Trap location reached

3) Change in a sense switch setting or, for 516's only, change in memory protect setting

4) IMP restart (for any reason)

5) Change in the status of any statistics program in the IMP

6) Change in Host status

7) Change in neighbor IMP status

A Status Report is identified by coming from IMP statistics, has 304 as the first data word, and has a valid software checksum. The code which checks the message is mostly straight-line code with few subroutines; most of the symbolic address tags begin with the three letters BTR. The NCC program processes a Status Report as follows:

1) Declares the IMP which transmitted the message to be up in its table of IMP status
2) Prints a logger message if this is a change in status
3) Turns on the alarm display light corresponding to that IMP (if light displays are used)
4) Stops flashing the display light for that IMP (if it was flashing it)
5) Prints the IMP program version number if it has changed or if it is not one of the three expected version numbers
6) If a Host interface is being tested, the ratio of successful tests to test attempts is logged

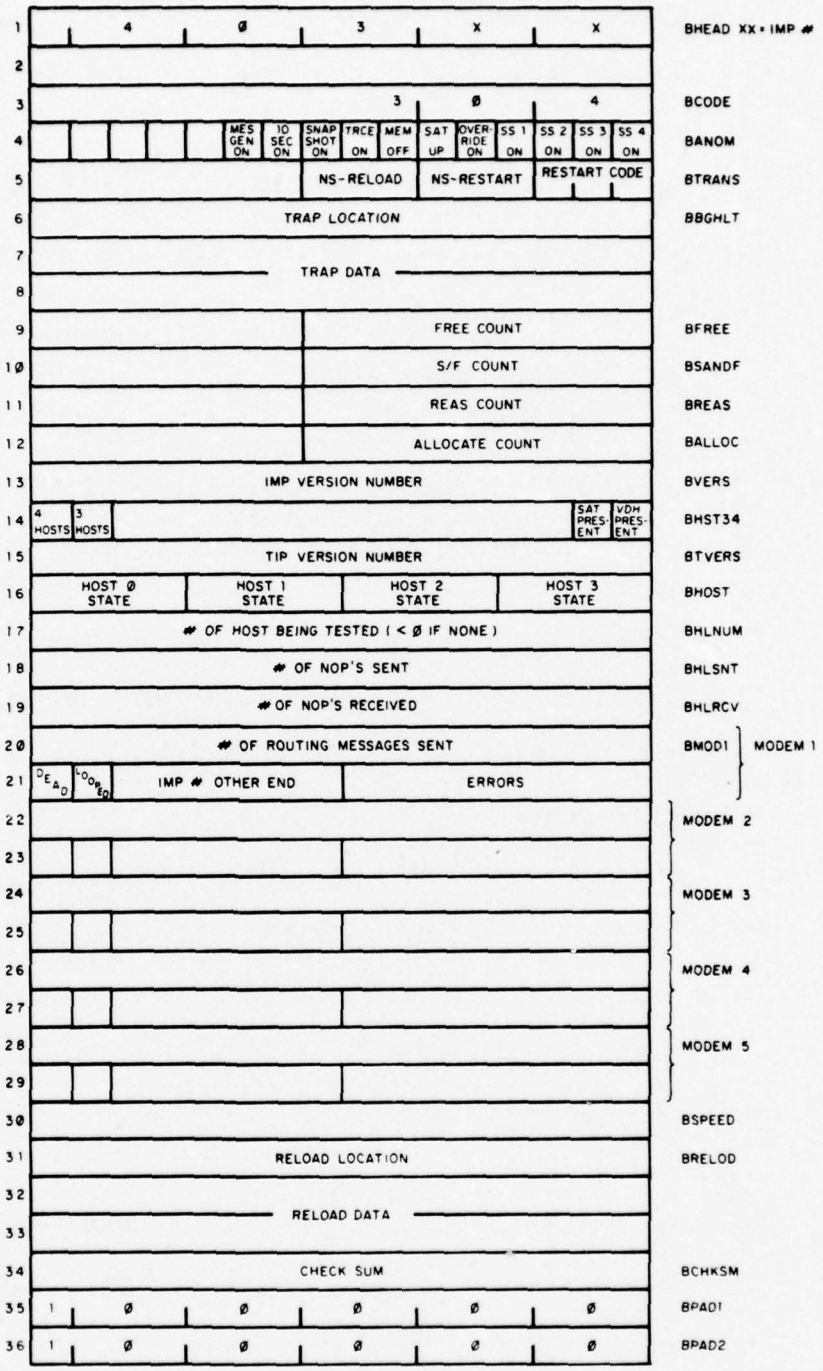


Figure 1: IMP Status Message

7) Prints the IMP's buffer count if it is out of range

8) Prints the store-and-forward count if it is out of range

9) Prints the reassembly count if it is out of range

10) Prints the allocate count if it is out of range

11) Prints changes in status for:

a) Message generator
b) Ten-sec. statistics
c) Snapshot
d) Trace
e) Memory protect (ignored for H316's)
f) The four sense switches
g) DDT enable flag ("Override").

12) Indicates a restart or reload if one occurred

13) Compares the reported HOST34 word with the value in the table HST34T and prints the reported word if different

14) Indicates a trap if one occurred. Specific trap printouts may be inhibited by the NCC operator

15) Indicates a "crash" if one occurred. These must be cleared with the IMP's DDT.

16) Prints the state of any connected Host if it has changed

17) If the site is defined in the NCC as being a TIP (pre-start TIPTAB), and if TIP subsystem is not running, that face is logged. If the TIP subsystem is not running, its version number is checked and, if incorrect, logged.

18) Prints the reported modem speed if it has changed

19) Records the status of one end of the five lines as one of the following:

a) Up
b) Down without errors
c) Down with errors, according to a DDT-settable error frequency
d) Looped
e) In limbo (no change in state)

20) Prints the ratio of line errors (in IMP "hello" and "I-heard-you" messages only) to the number of attempts, E/T if

a) $0 < E < T$

b) E=T and
(1) the line is declared UP
(2) the line is declared LOOPED at the reporting end
(3) the line is declared LOOPED BOTH

c) E=0 and
(1) the line is declared UNKNOWN
(2) the line is declared LOOPED at the other end only

21) For each line that is reported up, the number of the IMP at the other end is checked against the NCC's connectivity table. Three types of discrepancies are logged:

a) An IMP not contained in the table reports to the NCC with a line up

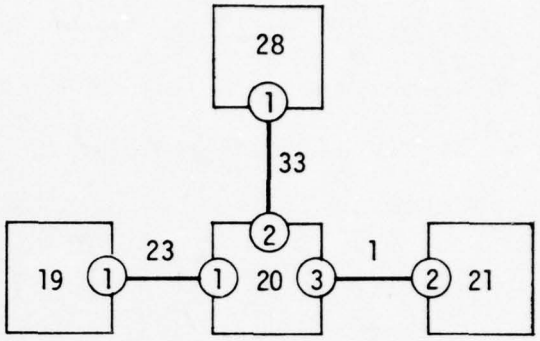
b) An IMP contained in the table reports that a new line is up (i.e., that IMP-modem pair does not appear in the table)

c) An IMP-modem pair contained in the table is reported as connected to the incorrect IMP.

While each IMP can only report about what it observes at its local modems, the NCC must make status determinations for entire lines; it must be able to match up the corresponding reports for both ends of each line. The program accomplishes line identification in two stages. In the first, the 'defined' line for each of the site's modems is determined by scanning the NCC's topology table, MODLNS. MODLNS consists of two-word entries. The table is indexed by line number. In each entry, the first word specifies the 'minus' end of the line, the second the 'plus.' Plus refers to the relatively higher numbered IMP of the two on the line. Both are stored as 8* (site number) + (modem number). The first stage results in BTRB1 being set up with the site's expected neighbors, indexed by the expected modem number. In the second stage, a permutation table, BTRTB2, is built using the neighbors as reported. BTRTB2 is indexed by 'actual' modem number and contains the corresponding expected number from BTRTB1. At this point, all of the line status and utilization data are processed using BTRTB2 to make the data refer to the correct line as defined in MODLNS.

MODLNS:

Line #		IMP NUMBER	MODEM NO.
1		20 21	3 2
.			
.			
23		19 20	1 1
.			
.			
33		20 28	2 1
n			



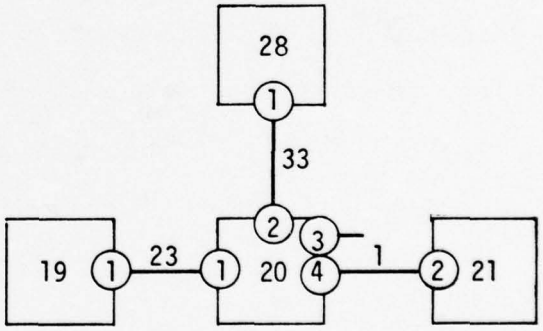
(Normal conditions)

BTRTB1 (For Site #20)

Modem #	Neighbor IMP Number
1	19
2	28
3	21
4	ϕ
5	ϕ

BTRTB2 (For Site #20 reported conditions)

1	1
2	2
3	ϕ
4	3
5	ϕ



(Reported conditions)

Figure 2: Reconciling Nominal and Reported Topology

For example, assume normal network conditions are recorded in MODLNS and BTRTB1, as shown in Figure 2. However, Site #20 reports to the NCC that Site #21 is reporting at the other end of Modem #4, instead of at the Modem #3 interface as shown in MODLNS. The program sets up BTRTB2 to ensure that throughput data reported for Modem #4 is recorded for Modem #3 for the Site #20 report being processed.

IMP Throughput Messages

The Throughput Report shown in Figure 3 is triggered by the end of a 52-second interval within an IMP. The report comes from IMP statistics, has 305 as the first data word, and has a valid software checksum. The NCC program uses the report to update its Host and line throughput tables. The code, at location STREPT, is straightforward. The utilization statistics from the message are added into the tables HTP1 and LTP1.

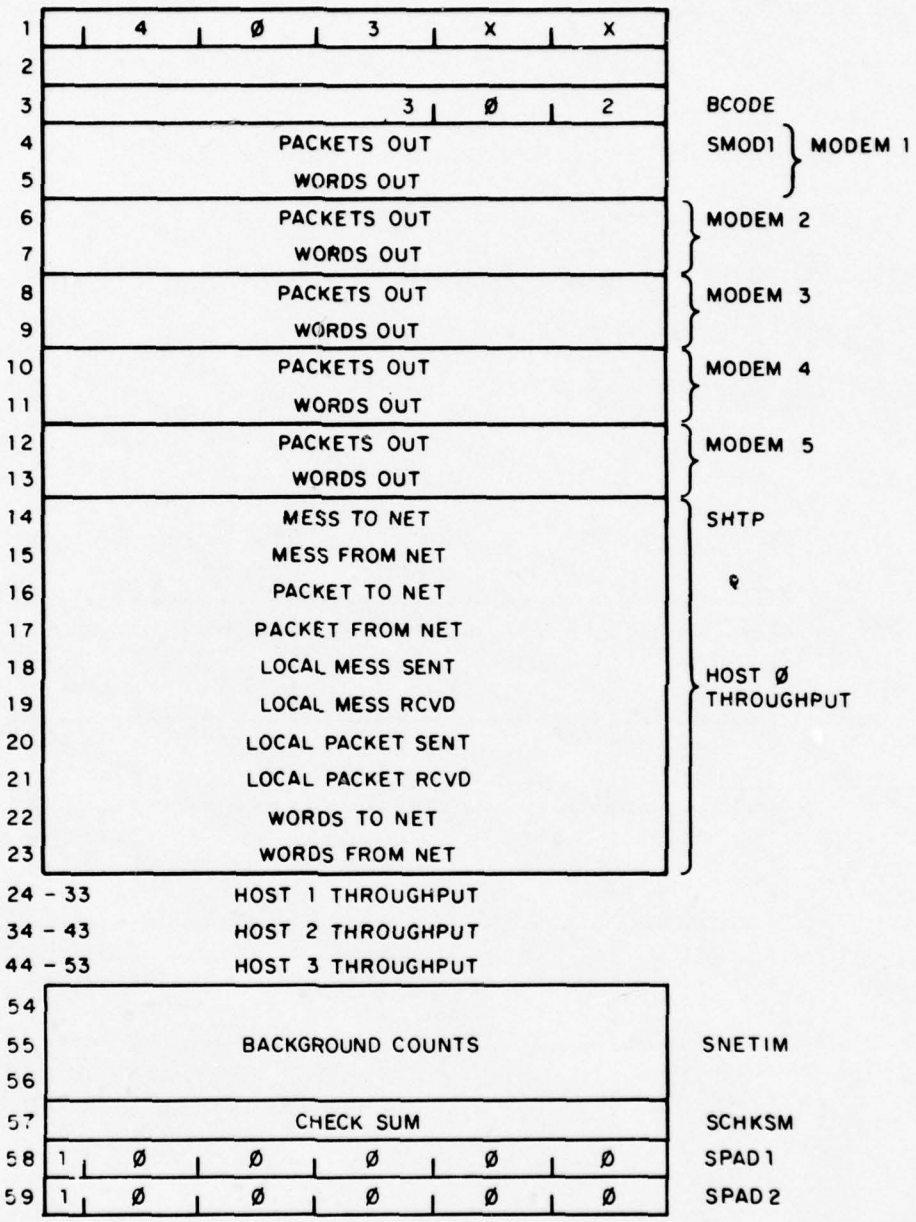


Figure 3: IMP Throughput Message

IMP and Line Timeouts

The program checks IMP and line status each minute. If an IMP has not reported recently, the following actions are taken:

1) The IMP is declared down in the IMP status table
2) A logger message is printed to indicate the IMP is down
3) The alarm display light for the IMP is flashed
4) All previous status indicators for that IMP are cleared to nominal values

PLUS END

MINUS END	No Info	Up	Down, no errors	Down with errors	Looped
No Info	Unknown ?	Up .	Unknown ?	Unknown ?	Looped plus >
Up	Up .	Up .	limbo	limbo	limbo
Down, no errors	Unknown ?	limbo	Down *	Down plus +	Looped plus >
Down with errors	Unknown ?	limbo	Down minus -	Down *	Looped plus >
Looped	Looped minus <	limbo	Looped minus <	Looped minus <	Looped both Z

Figure 4: Line Status Transitions
(Plus end = high-numbered IMP)

Line status is determined every minute by comparing the latest report from each of two IMPs on a line. The transition table shown in Figure 4 is used to choose the current state. If a change in status is detected, the following actions are taken:

1) A logger message is printed
2) The new status is recorded in the line status table
3) The alarm display lights are set
   a) UP - turn on both plus and minus lights, turn off flashing
   b) DOWN MINUS - turn on plus, turn off minus lights, turn on flashing
   c) DOWN PLUS - turn off plus, turn on minus lights, turn on flashing
   d) ALL OTHER STATES - turn off lights, turn on flashing

Each minute, IMP and line statuses are updated. IMP visibility is a central concept to this procedure. There are two possible reasons why an IMP might not be reporting in: first, because the IMP is actually malfunctioning, and second, because it has become isolated away from the NCC (i.e., 'invisible'). The NCC distinguishes these cases so that in the event of a failure which partitions the network, the NCC can determine the IMP involved, thus minimizing the number of irrelevant Logger messages and making such occurrences less of a deduction problem for the operator.

Using MODLNS to determine the topology, the NCC considers a line visible if there has been a report from at least one end of it 'recently' (see below); an IMP is visible if some line connected to it is visible. Only visible components have their status updated; invisible components are frozen in their last state until they again become visible. Thus, when a failure which causes a partition occurs, the first IMP beyond the frontier is declared DOWN (since there is a visible line to that site, but no usable line to the site), and simultaneously, all the IMPs beyond it become invisible (since there are no visible lines to any of them).

As mentioned, when there is a failure that partitions the network, the NCC correctly determines the IMP involved with the failure. The NCC must also determine the Line involved in such a failure. When the Line breaks (or the IMP dies--the two events are indistinguishable if they result in a partition), the NCC program has its 'modem' status tables (LINPLS and LINMIN, as filled by BTR) correctly indicating that one end of the line is DEAD, but erroneously indicating that the other end is UP. This information is left over from the last report, now obsolete, that the NCC had for the line. The approach is not to use a given report for a line more than once. In LINPLS and LINMIN, the low order fourteen bits give the last reported status for the line by:

0 UNKNOWN
1 UP
2 DOWN WITHOUT ERRORS
3 DOWN WITH ERRORS
4 LOOPED

The high order two bits serve as a counter to 'age' the report. When a new report is received, BTR sets the counter to two. CLOCK increments the counter (and clamps it at zero) before using the report, and the report is treated as 'Unknown' if the counter overflows (to zero).

III. OPERATOR CONTROLS

Light Box and Alarms

The program expects to use the lightbox and alarm to point out major events to the operators and programmers who are monitoring the network. The lightbox hardware includes a set of 16 switches which may be set manually. The program checks the state of the switches frequently, and responds to changes. When the switches are off, the program scans the several logical displays, choosing the "most important" one. The switches may be used to override the automatically chosen display with a particular one. The eight possible displays (in order of "importance") are:

No switch: Display the state of IMPs 1 through 64.

Switch 15: Display the state of IMPs 65 through 71.

Switch 14: Display the state of lines 1 through 64 in the "minus" or "high to low" direction.

Switch 13: Display the state of lines 65 through 95 in the "minus" direction.

Switch 12: Display the state of lines 1 through 64 in the "plus" or "low to high" direction.

Switch 11: Display the state of lines 65 through 95 in the "plus" direction.

Switch 10: Display the important service Hosts and other special "oddball" states.

Switch 5: Display all lights on, a lamp test.

When an important network element changes state, the program chooses the appropriate display, flashes the light which corresponds to the element, and sounds an alarm. A flashing light changes state every quarter-second. The alarm will produce one of the three following sounds:

Solid tone	--	IMP state change.
Quarter-second beep	--	Line state change.
Half-second beep	--	Service Host or "oddball" state change.

In addition, the alarm hardware includes a "watchdog-timer" delay. If the NCC Host program does not reference the lightbox/alarm device for several seconds, the alarm will sound continuously; this mechanism will notify the operators immediately of several kinds of possible program malfunctions.

Switch 16 is used to tell the program to stop the current alarm and flashing display. The transition from off to on signals

the program that the major event has been noticed, and the display can revert to the "next most important" one. Switch 16 has been installed at several places in the operations area, allowing a person who might be talking on a telephone to glance at the lightbox, then stop the annoying alarm.

DDT Debugging Commands

For those familiar with the IMP program, the following type-in commands are identical to the 516/316 IMP DDT commands: <break> <rubout> <return> <linefeed> <uparrow> <backarrow> <space> + - <0 to 9> / \ * , . = D E N S W and Z. The commands ! ? : " and ' are unique to the NCC Host. Note that the following IMP DDT characters are not used: $ B C H L O Q and T. Senseswitch 4 must be set for "destructive" commands to be executed, except as discussed below. The switch should be turned off most of the time.

Shift-control-P (or the break key) has the following effects:

1- any DDT output is stopped

2- the currently opened register is closed with no new contents

3- DDT forgets whatever number was being typed in

4- DDT types a carraige return-line feed

<rubout> has the following effects:

1- DDT forgets whatever number was being typed in

2- DDT types "# "

Period (.) has the value of the current register's address (15-bit)

<backarrow> has the value of the last thing typed by DDT

* has the value of the contents of the register addressed by what has been assembled as the current syllable. it always uses the current syllable as a 15-bit address and can be applied to itself or to any syllable.

, separates arguments to multiple argument commands

A1/ opens register at location A1 (15-bit address)

A1<backslash> open register at location A1 (used as a 9-bit relative address to the page . is on.

A1<linefeed> closes the currently open register (if any) and inserts A1 as its new contents (if supplied)

A1<uparrow> closes the currently open register (as linefeed) and opens the previous register

A1<return> closes the currently open register (as linefeed) and opens the next register

<space> and + both mean addition

- means subtraction

D means the number following is decimal

= types out the octal value of the last input eg: 3=3, 3+6=11, d11=13, "ab=40502, "ab+d10=40514, .=3033

A1,A2,A3Z clears ["zeroes"] core between limits - A1 is the constant core will be cleared to. A2 and A3 are the (inclusive) lower and upper limits, repectively.

A1,A2W dumps out ["writes"] core between limits - A1 and A2 are the (inclusive) lower and upper limits, respectively.

A1S starts up a program (i.e., causes a transfer to a core location). A1 is the (15-bit) address at which the program is to be started.

A1,A2,A3,A4E equals word search: under a mask of A1, DDT will search for words equal to A2 between the limits A3 to A4.

A1,A2,A3,A4N not-equals word search: under a mask of A1, DDT will search for words not equal to A2 between the limits A3 to A4.

" Assemble, into the two halves of the current syllable, the two ASCII values of the following two type-in characters.

' Assemble, into the current syllable, the (right-adjusted) ASCII value of the next type-in character.

For =, <return>, <linefeed>, <uparrow>, and <backslash> if the argument A1 is left out, the last number typed by DDT will be used as the argument.

For E, N, W, and Z, if either (or both) of the limits are left out, DDT will use the last limits specified for one of these commands. If the value is left out in an E, N, or Z command, the last value specified in any E, N, or Z command will be used. If the mask is left out of an E or N command it will be supplied from the last E or N done, or -1 if no mask has ever been specified.

DDT-NCC Commands

The debugging DDT commands have been expanded, introducing new commands for specific NCC functions. Most of these commands consist of using question-mark as an "escape character" to get away from the normal DDT type-in routine, a mnemonic alphabetic character (followed by the program generated typeout of the full command), and colon as a conformation character. RUBOUT may be used in place of the alphabetic character or colon to return to the normal DDT type-in.

! Clears the queue of events to be logged.

?Quickp:
Initiates the listing, on the summary Teletype, of the current anomalies in the network; that is, those conditions which are deviations from the nominal state. The character conventions for the Eight Hour Summary are used in reporting the IMP and line status.

?Host status:
Initiates a listing, on the summary Teletype, of which Hosts are up. The character conventions for the Eight Hour Summary are used in reporting the IMP status.

?Status:
Initiates the typeout of the Host and line status summary from midnight through the present fifteen minute period.

?Thrptsum:
Initiates a copy of the last hourly report.

?Daysum:
Initiates a typeout of the Host and line activity from midnight through the end of the last hour.

?Zero ctrs:
Clears the IMP down and RESTART/RELOAD counters. (The Tenex program NCCVER uses the counters to report on recent IMP restarts.)

Hitting the Break key
Any printout may be interrupted by pushing the BREAK key while the printout is in progress. DDT may be used to change parameters, etc. After 30 seconds of neither keyboard type-in nor DDT type-out, the interrupted printout will start at the beginning.

Modifying Parameters

Many of the NCC's activities are governed by various parameters which limit and define their domains. DDT is used to examine and change the settings of these parameters.

Line definitions are stored in a table beginning at location 21000. A line definition may be changed (without senseswitch 4) by putting the minus end of the line into location 2000, the plus end into 2001, and finally the line number into 2002. The end of a line is defined by the IMP number in bits 8-13 and the modem number in bits 14-16. For example, if line 30 had IMP 40 modem 3 at the minus end and IMP 63 modem 1 at the plus end, the line would be defined by:

```
2000/  0  503
2001/  0  771
2002/  0  D30
```

Machine types are stored in a table beginning at location 21630. A machine type may be changed (without senseswitch 4) by putting the value in location 2003 and the negative of the IMP number in 2002. There are four machine types:

```
          zero  --  516 IMP
greater than zero  --  TIP, the value is the TIP software version
            -1  --  316 IMP
  less than -1  --  Pluribus IMP
```

For example, if the NCC-TIP had a new patch which changed the version from 32702 to 32703:

```
2003/  0  32703
2002/  0  -D40
```

Idle MLC ports may be used for terminal testing. The TT-EXER portion of the MLC code expects parameters to be set in locations 2004 through 2007, and translates the parameters into "startup" or "stop" commands for a port. The locations are used as follows:

2004 Line width. For convenience, zero means 72 positions.
2005 Add together: terminal type, lower-case flag, type of printout. 00=> no padding; 10=> TTY33; 20=> TTY35; 30=> TI733; 40=> Centronics; 2=> terminal has lower case; 1=> want "QUICK BROWN FOX" rather than rotating line printout.
2006 Zero to stop testing. 1 to 17 to start normal testing, where the value is the baud rate selection,i.e., 2=>110, 4=>150, 5=>300, 7=>1200, etc. If 2006 is set to >17, bits 2-8 are used for the input OTA, bit 9 for LIU Xpatch, and bits 13-16 for output OTA.
2007 The port number.

For example, a rotating line printout on a 300 baud TI attached to port 6:

```
2004/  --  D80
2005/  --  30+2
2006/  --  5
2007/  --  6
```

The following are the parameters, the first column giving the location of the parameter, in octal, and the second giving the function of the parameter. Senseswitch 4 must be set to change these locations.

105 Minutes since the last hour.

106 Hours past midnight

107 Day of the month

110 Month of the year

111 Low two digits of the year.

--> When any of 105-111 are changed, the NCC should be restarted.

120 Nominal Primary version number for the IMP system. Locations 121 and 122 may contain the version number of IMP system undergoing pre-release testing.

123 thru 136 Clip level parameters used to check the IMP's buffer allocation. Each IMP includes, in every Status Message, the number of buffers in the "free-list", "store&forward", "reassembly", and "allocated". The NCC Host program will print a LOGGER message whenever an IMP reports that one of these categories has passed the parameter values.

21000+n+n Line definitions, two words per line. These should be changed by using locations 2000 through 2002, as explained above.

21630+n Machine type for site n: 0 for 516, -1 for 316, >0 for TIP, <-1 for Pluribus. For TIPs, the value stored is the TIP software version number. These words may be changed without senseswitch 4 by using locations 2003 and 2002, as explained above.

21740+n Configuration control word for site n (HOST34).

IV. PROGRAM DESCRIPTION

This chapter describes the NCC program organization, and is intended primarily for a system programmer. Where reasonable, reference to specific subroutine names in the code of the NCC system has been made. It is assumed that a listing of the NCC program is available for reference.

Program Structure

The program is structured in three logical levels - background, foreground, and I/O. Background, the idle machine state, is used as a mechanism for starting up some I/O events. Foreground processing, the bulk of the program, includes handling IMP messages. I/O routines are short, high-priority, un-interruptable strips of the program.

The Honeywell 316 has an interrupt structure which permits multiple interrupt levels. The hardware instruction set(1) allows both global and selective inhibit/enable control. The background state means running with most or all interrupts enabled, but not

---

(1)
For detailed descriptions, see Honeywell manuals:
Models 316 & 516 Programmers Ref Manual Doc.No. 70130072156
Intermodem Processor System Doc.No. 70130072261A

currently on any interrupt level. The foreground state means operating on either of the two lowest priority interrupt levels, with these two levels disabled, but with other interrupts allowed. I/O interrupts run to completion with the global inhibit in effect. Different hardware levels must work together for many of the NCC functions. Some of the software interfaces between hardware levels are very simple, yet some are complex. Most I/O interrupts signal the main TASK processing in some way. The interactions are:

IMP interface input and TASK.

IMP interface output, Background, and TASK.

MLC outin (LOGGER), MLC output, Background, TASK, and CLOCK.

Primary TTY, Background, and CLOCK.

TASK and CLOCK.

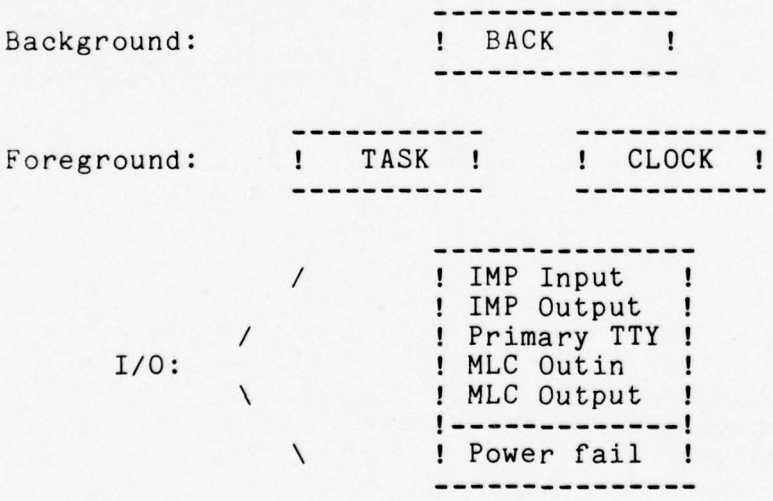


Figure 5: The Three Logical Levels

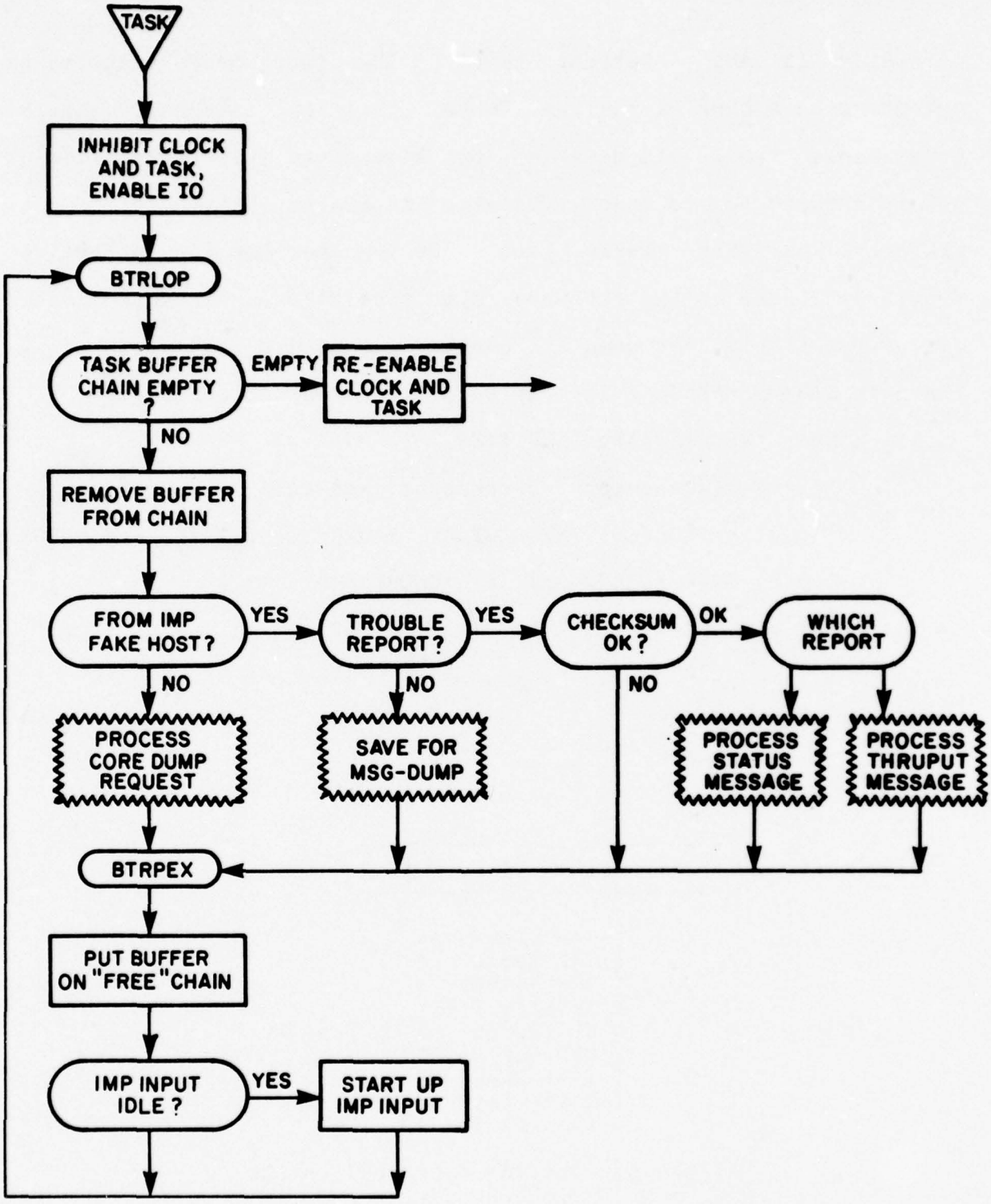


Figure 6: Major Portions of TASK Level

Foreground processing occurs on two low-priority interrupt levels called TASK and CLOCK. CLOCK, called by a 25.6 millisecond hardware interrupt, keeps track of the time of day, notices if any IMP or line has not been heard from for several minutes, and either performs or requests background to perform time-dependent functions. CLOCK is covered in more detail later. CLOCK and TASK do not interrupt each other, and can share subroutines.

The TASK interrupt is a "software" interrupt caused by execution of an I/O instruction. The IMP interface input causes the TASK interrupt whenever a full message arrives from the network. Figure 6 is the TASK overview, showing the four major sections of code within the wavy-edged boxes. Newly arrived network messages are routed to one of four places - IMP status message handling, IMP throughput message handling, IMP octal printout (MSG-DUMP) handling, and Host core dump request handling. The status message handling was covered in the "IMP MESSAGE PROCESSING" chapter. The other three message handling routines are short and straightforward; the throughput values are added into the proper Host and line tables; the MSG-DUMP message is copied to another area for later printout; and the core dump message is checked for legitimacy, then the parameters are saved in the NONPWC/NONPLO variables.

CLOCK Portion of Foreground

CLOCK is the second interrupt which is used in the foreground level of the program. It keeps track of time, notices IMPs and lines which have not reported to the NCC recently, sets flags for background level to initiate periodic printouts, and initiates periodic disk file updating. If the optional alarm and light display hardware is installed, it is updated frequently.

Since the clock can be locked out far in excess of its 25.6 millisecond period, the main entry, TOR, computes the number of intervals which have occurred since the last clock interrupt and calls TOWORK once for each elapsed interval. As explained in the section about program structure, I/O interrupts are enabled but TASK and extra CLOCK interrupts are inhibited.

Every Second

If the primary terminal timer is running (and inhibiting the output-only function), it is counted down. If zero is reached, the primary terminal state is set to "idle" (TTBSY=0).

If senseswitch 3 has been turned on, CLOCK checks for an inactive machine. The IMP-Host ready line is dropped. The LOG terminal must be idle and the primary terminal must be idle. If everything is inactive, the program halts.

Every Hour

The primary terminal hourly throughput printout flag is set for action by background level. If the hour is a multiple of eight, the total throughput and total status printout flags are set. If the hour is 24, the LOG terminal date and page number are set up for a new day.

If the "summary" time has reached the hour 1 or 25, the tables DHTP and DLTP are cleared.

The tables HTP1 and LTP1 are copied into HTP2 and LTP2. The HTP1 and LTP1 values are added into DHTP and DLTP. The HTP1 and LTP1 tables are zeroed.

Every 15 Minutes

The IMP and line status values are pushed on a stack that contains 96 such entries - one for each 15-minute segment in a day. In this way, status states are quantized to 15-minute periods, and any information on status since the previous midnight can be recovered.

Every Minute

Timeouts for IMPs and lines which have not reported to the NCC recently are checked every minute. The section of Chapter II called "IMP and Line Timeouts" covers this area.

Every 25.6 Ms. Interval

The memory locations which can be modified by DDT without senseswitch 4 being set are checked. If the operator is defining a line or other standard DDT operation, the actual change happens now.

The alarm and light display is updated. The set of switches is read and compared with the previous state. If a particular display is requested, it is put into the lights. Otherwise the program chooses among any "flashing" displays. If none needs flashing, the current set of reporting IMPs is displayed.

IMP Interface and Message Buffers

The IMP input routine (HIPR) attempts to keep messages coming into the NCC from the network and passing the messages to TASK. The variable HIBFP is used to indicate whether input is in progress; the variables BTRBEG and BTREND are used to form a list of buffers (messages) which have arrived but have not been acted upon. HIPR checks for hardware-detected errors; if none is found, the buffer is put on the end of the FIFO list; the input routine is marked "idle"; a TASK interrupt instruction is executed; and, if there is at least one "free" buffer, HSTINT is called.

HSTINT is used to start another input message. A buffer is removed from the "free" chain and its address stored in HIBFP. The DMC pointers are written and the interface is started. HSTINT is called from an input interrupt, from TASK level (with all interrupts disabled) after a buffer was put on the "free" chain and HIPR was zero (idle), and from the program initialization.

Background is responsible for sending messages. It checks the NONPWC/NONPLO variables for pending core dump requests. If a request (as stored by the TASK processing) is found, the reply message is set up. When the interface is idle (IHBUSY=0) and the IMP ready-line has not flapped, background will start the actual message and mark the output routine as busy. When the message has been sent, the output interrupt routine (IH) will mark the job as idle.

The NCC program maintains buffers for handling network messages. Buffers may be used as follows:

"free" - Available unused buffers are chained from the variable FREE through the first word of each buffer until a zero terminates this LIFO list.

Input - HIBFP contains the address of a buffer which is actively receiving a message from our IMP.

Awaiting attention - BTRBEG and BTREND are used to maintain a FIFO chain until TASK-level can dispose of all the buffers.

Being processed - When a buffer is taken off the TASK chain, its address is put into WPTR. IMP reports are digested and the buffer is put on the free chain.

All buffers have some common overhead which is used for buffer management rather than data. The remainder of the buffer may have an IMP or Host message. Wherever possible, the code referencing a buffer's contents uses symbolic offsets rather than fixed numeric references. The following is a list of symbolic offsets; for more detailed explanations, refer to the protocol or IMP documents.

Common Buffer Overhead Offsets

pb.chn	for free and Task chains
pb.s	final DMC

Offsets for IMP Status Message (see Figure 1)

```
BHEAD (2 words of Host-IMP leader)
BCODE (=304)
BANOM
BTRANS
BBGHLT (3 words)
BFREE
BSANDF
BREAS
BALLOC
BVERS
BHST34
BTVERS
BHOST
BHLNUM
BHLSNT
BHLRCV
BMOD1 (2*5 words)
BSPEED
BRELOD
BCHKSM
BPAD1
```

Offsets for IMP Throughput Message (see Figure 3)

```
BHEAD (2 words of Host-IMP leader)
BCODE (=305)
SMOD1 (2*5 words)
SHTP (4*10 words)
SNETIM (3 words)
SCHKSM
SPAD1
```

Offsets for Host Core-Dump Request	Offsets for Host Core-Dump Reply
CD.IMP (leader, 2 words)	copied from CD.IMP
CD.BS (bytesize)	copied from CD.BS
CD.BC (bytecount)	copied from CD.RBC
CD.ID (identifier)	copied from CD.ID
CD.RBC (returncount)	(unused)
CD.ADR (dumpaddress)	copied from CD.ADR
CD.WC (dumpwordcount)	copied from CD.WC [followed by the requested core dump.]

MLC Control and Utilities

This section describes most of the Multi-Line Controller (MLC) module. The code includes the MLC interrupts, the control structure, and the software interface between the "system" and "user" portions. The LOGGER, MSG-DUMP, and TT-EXER routines are "user" portions which rely on the MLC structure. The MLC code is organized to handle output to several ports - there are few restrictions on the non-MLC outside world which wants to use one or more ports for some purpose. There are several device-dependent routines to handle the annoying details of "padding" for terminals where a character may need several character times to accomplish its function. Many subroutines are re-entrant, using the printing port number to distinguish between the callers.

Of the three potential MLC interrupt routines, one is non-existent, one is trivial, and one is complex. The MLC input interrupt vector is never used, and the interrupt mask bit is never set. The MLC output routine, ML.OPI, is about half a page of straightforward code. The MLC outin interrupt is used to run much of the NCC printing code. (The MLC outin interrupt is actually caused by a 3.3 millisecond clock, but should not be confused with the 25.6 millisecond clock used for the IMP and line timeouts, etc.)

"User" routines which need to print on an MLC port are organized as follows: some Background code decides to print a string; it marks a Background/MLC flag as "busy"; it fakes an outin

interrupt; it uses the ML.OCW routine to print the characters, one at a time; it marks the Background/MLC flag as "idle" and jumps to ML.SLP to terminate. The simplicity of the MLC "system" interface allows new printing functions to be added without undue difficulty.

The ML.OCW routine is responsible for putting a character (and a port number) into the output buffer, saving the return in the ML.SPT table, and jumping to ML.SLP. This is the "user" to "system" transition in the control structure. Three subroutines, ML.FRK, ML.WF, and ML.HF, allow a "user" process to start up an independent fork to print on another port, and to wait for a fork to terminate. The ML.SVC subroutine allows the calling of another subroutine with the A, B, and X registers kept transparent. The subroutines PAD.33, PAD.35, PAD.TI, and CN.OUC are utility handlers which pad RETURN or FORMFEED characters properly for Model 33 Teletypes, Model 35 Teletypes, TI 733 terminals, and Centronics 306 lineprinters.

The outin interrupt control has the following pieces:

At ML.OII, the registers are saved.

If all the words in the previous outin buffer have not been processed, go to ML.XIT.

Make up a new buffer and set the DMC pointer words.

At ML.SLP, if there is a fork to start, do it. (I.e., give control to a new "user" routine.)

If the previous outin buffer has any words in it, remove a word (containing the port number) and dispatch through the ML.SPT

table. (I.e., give control back to a "user" routine which was waiting for its port to be able to accept another character.)

Finally, at ML.XIT, restore the registers and exit from the interrupt.

LOGGER Printing

TASK and CLOCK discover events which should be printed; background starts the terminal printing the first character; and the remaining characters are all printed from the outin interrupt level.

During TASK and CLOCK execution, each request for a LOG printout is encoded into a three-word block and put into a ringbuffer by the subroutines FLG or FLGCOD. The first word gives the time of the message, in minutes since midnight; the second contains, in its right half, the number of the message to be typed, and in its left half, the number of the IMP or Line involved; the third contains a word of data that some messages include with their text. If the LOG is idle (TTBSY1=0), background (TTG01) "fakes" an interrupt and starts running the LOGGER code as though already on the outin interrupt level.

The LOGGER module uses the three-word block pointed to by the ringbuffer service pointer for the next printout. It takes the

right half of the second word, indexes through the CODTXT or STRMSG table, and selects the ASCII text string. If the string contains any of the following characters, the parameter in the third word of the block will be substituted for the character.

Control A = Decimal print of left 8 bits.

Control B = Decimal print of right 8 bits.

Control C = Octal print of whole word.

Control E = Signed decimal print of whole word.

In order to compress the output somewhat, the LOGGER routine attempts to print several text strings on a line. It uses the following rule: only the text of the next string (without the time, IMP number, etc.) will be put on the end of the current line if:

1) The time for the new message is the same as the time for the current line, and
2) The new message is for the same IMP or line, and
3) The new message is not a 'special' message, i.e., one with a code in excess of STREET, and
4) There is sufficient room on the current line for the text of the new message.

These tests are performed by running through the code for the LOGGER twice. Prior to the first pass, SIMFLG is set to zero. SIMFLG inhibits printing during this pass. If any condition is not fulfilled, SIMFLG, in conjunction with the subroutine UNSIM, is used to end the simulation pass and begin typing on a fresh line. If all

the tests succeed, the LOGGER types a comma followed only by the text for the new message. In either event, the message is actually typed by re-executing the same code, the second time with SIMFLG non-zero. When the message has finished, the ringbuffer service pointer is incremented and the ringbuffer is tested for empty. If not empty, the next entry is started; if empty, the LOGGER is marked idle.

MSG-DUMP and TT-EXER Printing

Network messages which arrive from an IMP's fake Host but are not status or throughput messages have their entire contents printed on the MSG-DUMP terminal as an octal dump. Each message has a printed header which shows the time, date, and IMP name. This facility is an IMP debugging aid, and is most often used to print out a packet which has passed the modem checksum hardware but has a software-detected checksum.

MLC ports which are not being used by the LOGGER or MSG-DUMP functions may be used for terminal testing. The DDT operator commands which control the ports are described in Chapter III, "OPERATOR CONTROLS". The printout may be either a rotating (by line) alphabetic pattern or the standard "quick brown fox" sentence.

Primary Terminal

The control structure is somewhat complicated, due more to the number of functions assigned to a single terminal than to the program levels involved. Figure 7 shows five separate states for the handler. The terminal interface is half-duplex - it cannot handle input and output simutaneously; however, the hardware (and software) can detect the depression of the keyboard BREAK key while output is in progress.

The functions divide into interactive (keyboard with possible immediate response) and report-printing (output only) categories. Figure 7 shows the states and the flow. There are two "idle" (or "not currently printing") states - in one of these states, the initiation of printing is specifically inhibited for 30 seconds. Thus, if a person is doing interactive work (for example, using DDT to examine memory), control stays within states 1, 2, and 10. Only after 30 seconds without type-in and/or response will the terminal state reach 0, where pending TASK or time-of-day activated printing requests are considered. The state transition 0 to 4 happens at background level where a standard interrupt is "faked"; the transition 1 to 0 happens at CLOCK level; all other transitions are part of the standard I/O interrupt code. Finally, the dotted arrow shows the special case where the final character of a command has been typed-in, the next response will be a REPORT printout, and a 30-second delay would be annoying.

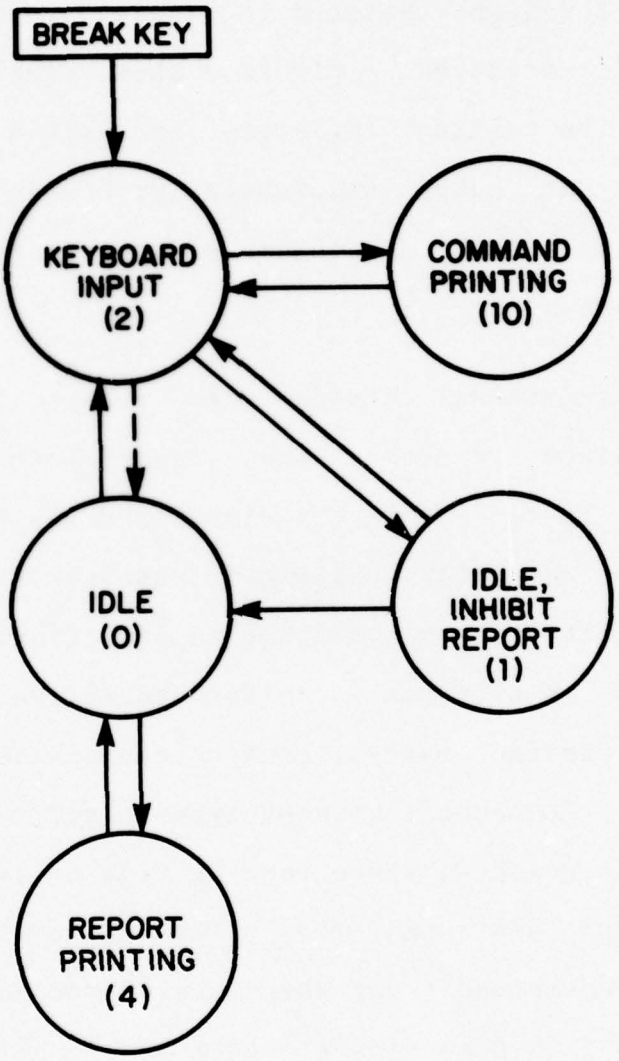


Figure 7: Primary Terminal States

Report printouts may be either CLOCK activated or operator-requested. The list of variables at RQUICK is a set of flag words for pending reports. The list at LQUICK is a dispatch address list for the code which prints the report. The background routine TTGO, after checking that a report may be allowed to come out, checks the RQUICK list for non-zero; if a non-zero is found, the interrupt is faked and the dispatch is made. Other routines which wish to request a report need only mark the correct variable.

V. TENEX DATA COLLECTORS

Data Gathering

Tenex programs collect four kinds of NCC Host data into disk files. The IMP and line throughput is collected hourly and daily. The IMP and line status is collected quarter-hourly. The raw LOGGER data is collected frequently. The current topology table may be collected quarter-hourly, although it is not collected on a regular basis.

NCCWCH collects the throughput and status data into files. The throughput is stored in a file "<NCC>mondd.DAT", where "mon" is the 3-character abbreviation of the month and "dd" is a 2-digit day of the month. The status data is stored in file "<NCC>monyr.ST", where "mon" is the month and "yr" is the 2-digit year. The file layouts are summarized in Figure 8. The NCCWCH program operates as follows: when initially started, it accesses the NCC Host's parameter table, discovering the location and size of the data tables; thereafter it checks the NCC Host's time and date for a new period; if the hourly throughput or quarter-hourly status has not been captured, it is retrieved and filed; and it "sleeps" for 4 minutes before looping. The NCCWCH program has defensive code to aid in recovering and/or restarting if error conditions are discovered.

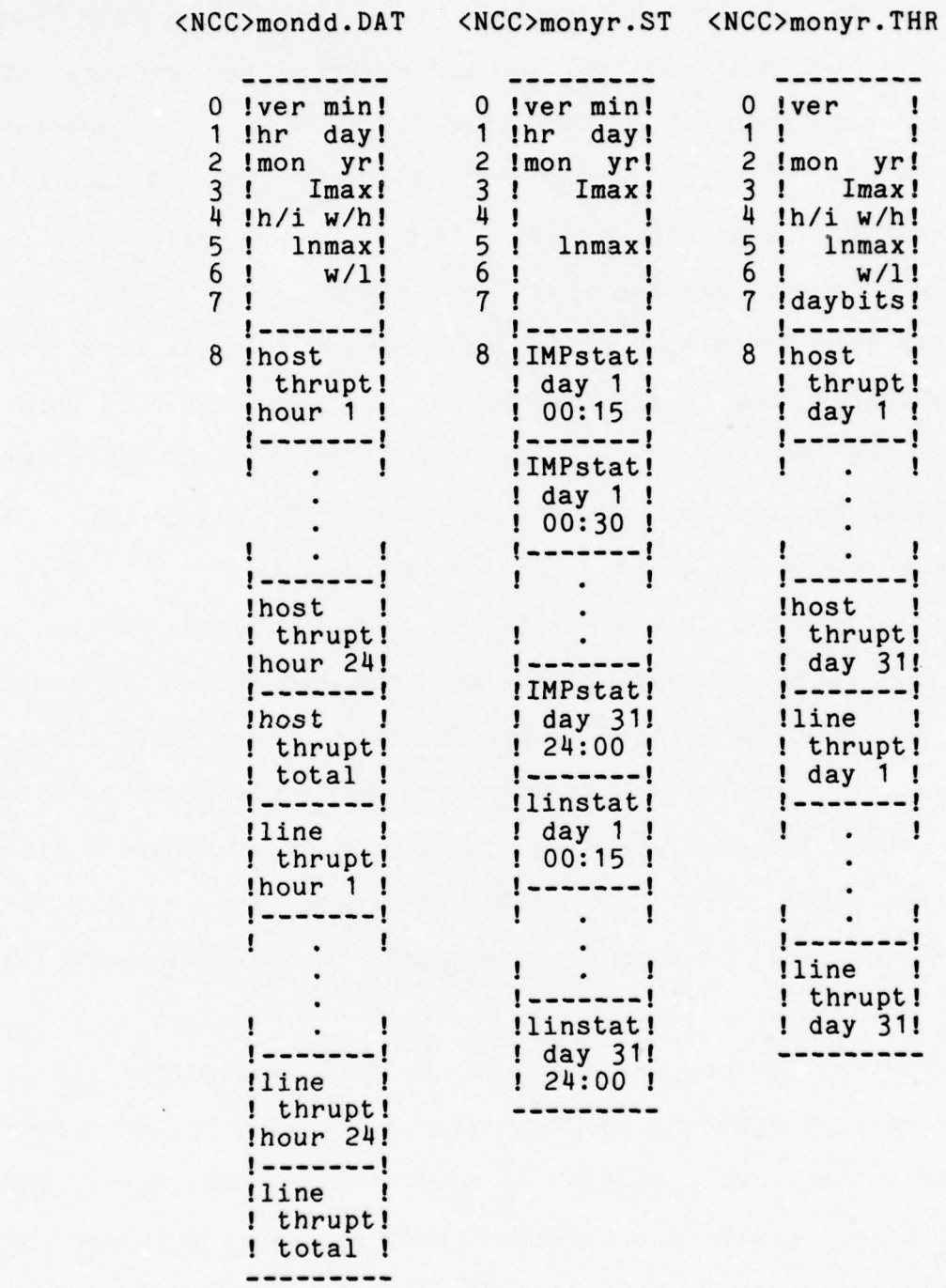
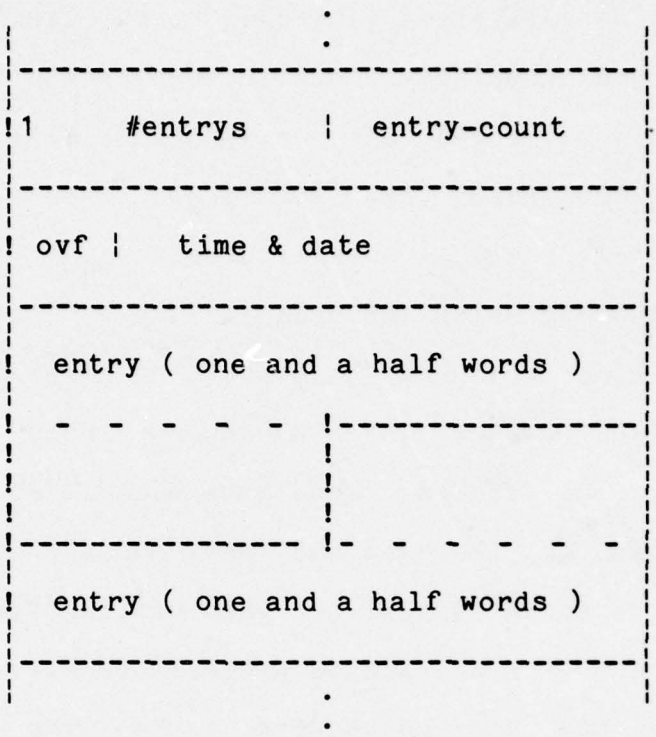


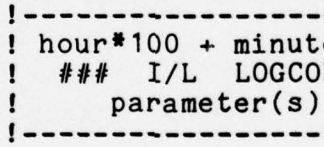
Figure 8: Throughput and Status Files

NCCLOG copies the NCC Host's raw LOGGER data into file "<NCC>L.mmdd", where "mm" and "dd" are the 2-digit month and day of month. Figure 9 summarizes the file structure. The program "sleeps" for 3 minutes, then examines the NCC Host's LOGGER buffer. It uses the fill and print pointers to the in-core ringbuffer to watch for new entries. When several are present, it copies the LOGGER entries into Tenex memory, formats them, and writes them into the file. Whenever data is appended to the file, two (36-bit) words of overhead are included which identify the block of data, the number of LOGGER entries, the NCC Host's time and date, etc. A 48-bit LOGGER entry is written into one and one-half Tenex words. If the number of entries in a block is odd, a half-word of fill is added to the block, resulting in all blocks ending on word boundaries. An entry, as written in the file, is three half-words; the top two bits of each half-word are zero; the first 16 bits contain hour times 100 plus minutes; the second 16 bits have 7 bits of IMP or line number, 1 flag bit (0=IMP, 1=line), and 8 bits of LOGGER code; the final 16 bits are a parameter whose interpretation depends upon the code.

The reliability of the distributed data collection is a problem for several reasons: occasionally, a large computer system with rotating memories will have extended down time; the Tenex and NCC Host protocol design and implementation may allow a message to be "lost", and a process to hang forever; and modifying the program in any single Host may cause problems in another Host. These problems have been attacked as follows:



A portion of the <NCC>L.mmdd file.



A single LOGGER entry.

Figure 9: LOGGER File

1. Using two Tenex systems (attached to two different IMPs) has resulted in acceptably reliable Tenex performance. The files collected are merged by the program FIXFIL whenever the "primary" collector has had trouble on a given day. More Tenex systems could be used if necessary.
2. A Tenex process which interrogates the NCC Host memory should have a timer running while awaiting a response. If the NCC Host or Tenex NCP or IMP subnetwork loses the inquiry or response, the process should be capable of restarting after about 2 minutes. Because the Tenex monitor does not allow a process to time itself out when waiting for a "raw message" input, the NCCWCH and NCCLOG program normally run as inferior forks to the program NCCFRK. When the fork is about to request input, it communicates via a shared page of memory with its superior. The NCCFRK program will wake up every few minutes and check for any forks which have been hung for too long. If one is found, the fork is killed and a new fork is created with a fresh copy of the appropriate program.
3. Programs which access NCC Host memory initially reference the "memory map" which resides at location 1007 of NCC Host core. If a new version of the NCC Host program were put up too quickly and it had the major core tables located at new locations, the data gatherers might collect bad data. A mechanism exists which can start and stop the NCCFRK inferior forks by communicating via the shared memory page. The FRKCON

program (from the NCCFRK.MAC source) should be used to stop the NCCWCH and NCCLOG programs and all Tenexs before a newly assembled NCC Host program is introduced.

Because our primary Tenex system does not have unlimited disk storage, file archiving is necessary. The daily throughput files (mondd.DAT) are the major offenders, using about 60 disk pages per day; they are copied to magnetic tape and deleted from the disk when they are 3 days old. However, each morning, the previous day's total throughput is copied from the "25th hour" of the "mondd.DAT" file into the appropriate day of the "monyr.THR" file by the NCCUPD program. The "monyr.THR" and "monyr.ST" files are archived after a month has elapsed. Thus, hourly throughput data is on-line for several days, and daily throughput and quarter-hourly status is on-line for at least 30 days, yet the total disk space is about 500 pages.

Finally, the PDP-1 is another data collector. From 1973 to 1975, while the Tenex "raw message" JSYS's were in a state of flux, the PDP-1 was the primary archiver of the NCC Host data. Currently the data collectors are not used, although the PDP-1 does access the NCC Host every few minutes and insure that the time and date of both machines are within a few minutes of each other.

The NETSUB package

NETSUB is a collection of subroutines designed to simplify two aspects of using the Tenex NCC Host data: referencing the data files mentioned in the previous section, and referencing the currently advertised ARPANET topology. Whenever an IMP or line is added or removed, three files are modified to reflect the change: NETSUB.MAC, NETSUB.REL, and MODLNS.TOP. All subroutines are compatible with the F40 Fortran run-time system; they may be converted to have F10 or BCPL compatible calling sequences. The following list includes the subroutine names, arguments, and brief explanations:

For <NCC>mondd.DAT file:

OPN(mo,day,yr)	Uses arguments "mo" and "day" to find the file and map it into the address space. Returns the year in the 3rd argument.
HOST(hr,imp,host,array)	Given the hour (1-24 or 25=total), the IMP# (1-127), and the Host (0-3), return the eight throughput values in the array.
LINE(hr,line,thr+,thr-)	Given the hour (1-25), the line# (1-199), return the two throughput values the the 3rd and 4th arguments. The 3rd argument has the value for the "plus" direction, namely the direction from the lower-number IMP toward the higher. The 4th is the "minus", or high toward low direction.
CLS	Remove the file pages from the address space and close the file.

For <NCC>monyr.THR file:

OPNTHR(mo,yr)	Uses arguments "mo" and "yr" to find the file and map it into the address space. If the given "yr" is zero, the most recent year will be chosen and returned in the 2nd argument.
GETHO(day,imp,host,array)	Given the day (1-31), the IMP# (1-127), and the Host (0-3), return the eight throughput values in the array.

GETLN(day,line,thr+,thr-) Given the day (1-31), the line# (1-199), return the two throughput values in the 3rd and 4th arguments.

CLSTHR Remove the file pages from the address space and close the file.

For <NCC>monyr.ST file:

OPNST(mo,yr) Uses arguments "mo" and "yr" to find the file and map it into the address space. If the given "yr" is zero, the most recent year will be chosen and returned in the 2nd argument.

GETIS(day,qtr-hr,imp,state) Given the day (1-31), the quarter-hour (1-96), and the IMP# (1-127), return the "status" in the 4th argument.

GETLS(day,qtr-hr,line,state) Given the day (1-31), the quarter-hour (1-96), and the line# (1-199), return the "status" in the 4th argument.

CLSST Remove the file pages from the address space and close the file.

For topology:

UPTOP Load the current advertised topology and site names from file MODLNS.TOP. If successful, print out a message; if unsuccessful, retain the topology from the original program load.

LNIMP(line,imp+,imp-) Given a line# (1-199), return the lower-numbered IMP in the 2nd argument, high IMP in the 3rd.

IMPLN(imp,n1,n2,n3,n4,n5) Given an IMP# (1-127), return either zero or a neighbor IMP# in arguments 2 through 6.

SNAME(imp,text) Given an IMP#, return a 5-character ASCII name.

SITNAM(imp,host,array) Given an IMP# and a Host# (0-3), return a 25-character ASCII Host name. The array will have trailing spaces added to fill out to 25 characters.

Programs which use the NCC data

This section gives brief descriptions of Tenex programs which reference the NCC Host data files. Most are written in a higher-level language, so they may be read and understood easily. Most use the NETSUB package. The machine language programs NCCWCH, NCCLOG, NCCUPD, and FIXFIL have been described earlier; NCCVER is covered in the next chapter.

NCCCHECK may be used to check the data collectors. It prompts the typist for a recent day, then checks the files mondd.DAT and monyr.ST for any missing intervals in the data. [source file is <COLE> CHECK.F4]

IMPST uses the monyr.ST file to find IMPs that have been down on a particular day. The algorithm used is primitive. [source <COLE> IMPST.F4]

SUMRY can duplicate the NCC Host summary printout of Host and/or line throughput from either the mondd.DAT or monyr.THR files. [source <COLE> SUMRY.F4]

LNTHR produces a monthly summary printout of line traffic. [source <COLE> LNTHR.F4]

HSTHR produces a monthly summary printout of host traffic. [source <COLE> HSTHR.F4]

AVGHT prints total network Host traffic for each day of a month. [source <COLE> AVGHT.F4]

SATLN prints the Atlantic satellite experiment traffic, either the hourly or the daily line throughput. [source <COLE> SATLN.F4 is a stripped down copy of SUMRY.F4]

IMPDOWNS prints an array of IMP downs for all sites for the last 3 weeks. The data is derived from the monyr.ST file. The default algorithm is the same as is used in the IMPST program, but others may be tried. [source <COLE> IMPDOWNS.BCP, BSUBS.MAC]

TRAF checks the validity of the mondd.DAT file in several ways, printing out any "overflows" or other oddities. Optionally, the day's total Host traffic may be added into

the file <NCC>TRAFFIC.BIN, which contains monthly total Host traffic for 12 months. [source <COLE> TRAF1.F4]

MAXTRAF references the TRAFFIC.BIN file, producing a list of the 10 busiest Hosts in a month. [source <COLE> MAXTRF.F4]

LOGPRT can produce an NCC Host LOGGER printout for a day from the <NCC>L.mmdd file. [source <COLE> LOGPRT.MAC]

LSTKIL is a large "batch" program which runs every night and produces about 30 separate files of information gleaned from the combined <NCC>L.mmdd files of both Tenexs. The output files are produced for different individuals who are interested in portions of the daily Network events. The files may be listed on the Division 6 lineprinter or sent as "mail" files. The L.mmdd file is sent both to the CCA Datacomputer for on-line storage (for about 2 months duration) and to the local magnetic tape archive. [source is several .MAC files in <NCCLOG>]

DCQUER is a program for retrieving LOGGER events from the CCA Datacomputer. Queries are typed-in using the data language. [source <NCCLOG> DCQUER.MAC]

INFLOG uses the current period of the <NCC>monyr.ST file to produce a current status display, formated for an Infoton storage tube display terminal. [source in <BARKER>]

MONTH uses a <NCC>monyr.ST file to estimate IMP down time, both MTBF and MTTR. [source in <BARKER>]

VI. NCC PROGRAM MAINTENANCE

General Maintenance Pitfalls

There are several potential problems associated with ARPANET NCC software support. This section attempts to expose some of the pitfalls awaiting the program maintainer. First, there are some general rules - perhaps obvious, but mentioned here anyway. A systematic method of file naming is helpful when maintaining both "service" and "debugging" versions of a program. The Tenex directory <NCC-PROG> contains almost all of the files needed. New program versions are named "NEW.*" until the program is actually installed in the ARPANET; the files are then renamed to "NCC.*". Version numbers on files correspond to the internal version number of the entire program. Use of the RUNFIL and DO Tenex subsystems make most of this trivia easy to remember; see files ARPA.ASSEMB, ARPA.ETC, ARPA.RELEASE, and ARPA.PATCH.

When modifying source modules, the programmer should remember that most of the modules may be used in NCC programs on other networks; conditional assembly switches should bracket code that is not desired generally; if extensive changes are made, all other NCC programs should be assembled and checked for simple page overflows, etc.

Debugging new code is often difficult. A "backroom" network will not have all the ARPANET resources. There will not be

an identical collection of terminals available. Thus, some of the software can not be tested directly.

Changing the software in the on-line NCC machine is tricky due both to the movement of core memory tables and to the distributed data collectors.

Invariant Traps

"Traps" in the Honeywell IMP and TIP consist of a mechanism for notifying the NCC Host of the P, A, and X register values when an unusual (but recoverable) event is detected. In a given program version, the P value is associated with the event, and the A and X are often parameters. Unfortunately, whenever the program is changed, many of the P values change also. "Invariant traps" are a mechanism for associating a small and easily-remembered number with an IMP or TIP event; operators and programmers can be relieved of the burden of learning new P values with every new IMP or TIP software release. The simple concept, unfortunately, involves many steps; some are "automated" into Tenex command files, but several steps remain.

The NCC program contains several translation tables which are used to convert a P value into a small fixed number. Each table has an IMP or TIP version number associated with it. The tables are

referred to as "slots". As of NCC version 131, there is room for six tables, each with up to 100 (octal) pairs of values. In detail, a new table is added as follows:

1. Check the current NCC program patch paper tape. The list of .BIN files used is attached to the front of the tape.
2. Check the computer-room NCC program listing for the currently active slots. Choose an unused slot.
3. Run the FASBOL program INVTRP.SAV. It will prompt you for some parameters, including the slot and the IMP/TIP.TXT file. It will produce a new .H16 file.
4. Assemble and list the trap file:

```
@RUN <IMPSYS>DAPX16
*filnam=NCC.SYM[1,1467],filnam.H16(LQ)
*^Z
@TIPCOPY filnam.LST
```

5. Go to step 3 in the next section on patching.

How to Patch

File PATnnn.H16, .BIN, and .LST are used for patching NCC version nnn. A patch paper tape is made up from the expendable initializer (NCCIN.BIN), all invariant trap files, and the program patch file. The steps involved in patching are:

1. Edit the changes into file PATnnn.H16. Use symbolic references where helpful. Remember that the intent should be clear to the person who must edit the sources later.

2. Assemble and list as follows:

```
@RUN <IMPSYS>DAPX16
*PATnnn=NCC.SYM,PATnnn.H16(LQ)
*^Z
@TIPCOPY PATnnn.LST
```

3. At a hardcopy terminal, prepare file PATNCC.PUN, punch two copies, and make a new IMAGE file. The simplest method is to edit file ARPA.PATCH to reference all .BIN trap files, then use RUNFIL.

4. Get the two paper tapes. Find an available backroom machine with a high-speed reader. Check that both tapes will load correctly. Write "NCC PATCHES" and attach the "punching" typescript to the first fold of one tape; label the other "BACKUP NCC PATCHES"; write the date on both tapes.

5. After checking with the operations staff, halt the NCC Host by turning on senseswitch 3 for a few seconds. Load a new tape into the NCC Host. Answer the "ZERO TABLES..." question with N and carefully enter the date and time. Replace the two old tapes with the new. Put the patch or trap listing into the computer-room NCC listing book.

NCCVER - Verify NCC Host memory from Tenex

File <IMPDMP>NCCVER.SAV is a Tenex program whose major purpose is to compare the NCC Host's memory with file <IMPDMP>IMAGE.NCC and print out the differences. The program needs Tenex's NETWIZ capability; users IMPDMP and NCCOPS have the capability on Tenex-C and Tenex-E. When started, the program prints some parameters about the current NCC version, prints any differences between the "official" and actual topology, and prints the prompt character <backarrow>. The program expects one of the following responses:

V - verify core with the image file and print any differences. The limits used are 60 and 77777, and words which are zero in the image are not printed.
D - print DOWNS and RESTARTS since the last ?Zeroctrs: command.
number<return> - print current contents of a single core location.
number,number<return> - print the current contents of an area of NCC core memory. The numbers are lower and upper limits, and must be within 750 words.
^Z - control-Z returns to the EXEC, after releasing the raw message handle.

The NCCVER program may be forced to send its output to a file rather than to the controlling terminal. Use the EXEC commands "GET <IMPDMP>NCCVER" and "REENTER", then supply a file name. The default file is the Dataproducts lineprinter ("NET:.NCC-TIP-400002"), and the V and D options are always executed.

Producing a new NCC version

The following few steps should help remind the programmer of the editing, assembling, listing, and punching stages of producing an NCC program. Most of the type-in has been automated into files.

1) Edit file ARPA.H16 to increment the octal value of .VERS. by one and to update the "last editor and date" line.
2) Edit source files. Make SRCCOM's.
3) As necessary, delete NEW.*;*, NEWIN.*;*, FOO.*;*, and *.TMP;* There should be more than 150 pages available.
4) RUNFIL of ARPA.ASSEMB
5) Check the FOO.ERR file. If there are errors that you want to fix in the sources immediately, then return to step 2.
6) Use DELVER, EXPUNGE, and DSKSTAT to remove the extraneous source files and check that there are more than 40 disk pages available.
7) Check the paper tape supply in the FACIT punch. You will need about two inches thickness.
8) RUN <IMPSYS>DOO with file ARPA.ETC
   Set the parameter Version to be the same as the value you gave to .VERS. in step 1.
   Check the typescript for unexpected printouts.
9) List file NEW.LST, perhaps with TIPCOPY's option NI.

Debugging in a Backroom Network

Set up a "backroom net", using as many resources as possible. The Prototype 516 machine should have the current IMP program. The Spare-TIP should get the NCC Host program. The two machines should be connected by the special IMP-to-Host cable, using Host interface # 0 on both machines. If the Prototype 516 does not have IMP #40 as its address, patch the IMP program to send its

trouble reports to the correct address, and patch the NCC program to know about its own Host address. Set up the NCC topology table (MODLNS) to show the actual configuration.

Set up the proper MLC ports with terminals. Try to borrow the extra Centronics printer from the Division-6 medical group. Use Teletypes, TIs, display terminals, or whatever you can scrounge. If you have modified the lightbox code, try to borrow the lightbox for a couple of hours. If you have modified the Host-Host core dump core, try to get the PDP-1 Host moved to the backroom net for a few hours.

The actual ARPANET NCC program release

Make an up-to-date patch tape, including the proper invariant traps in the correct NCC table addresses. Check all data tables that should be transparent to the program change, such as the throughput tables. Make up a special version of the SHUFFL.H16 program to help you rearrange memory contents. If you may have to recall the new NCC program, make up a SHUFFL program to move everything back. Just before the program switch, halt all Tenex data collectors with the FRKCON program. Halt the machine, shuffle core, and load the new program. If it flies, restart the data collectors. Check out the parts of the program that you were unable to test with the limited resources of the backroom net. Run the DOO file "ARPA.RELEASE" to clean up the support file renaming. Check

that NCCVER produces a good verify. Put all listings and tapes in the proper places. If any operator commands have changed, describe the changes in the *ARPANET LOG* and update the *computer room* copy of T.I.R.90.

Program Crashes

The NCC is designed to run continuously without any attention by the operator. In the event that something does happen to the program requiring intervention, there are several things the operator must know.

The NCC program has five entry points:

1000	Fresh start - all tables are cleared.
1001	Restart - tables are left intact
1002	Dump the NCC, and all its tables, through modem 1.
1003	Load the NCC, and all its tables, from modem 1.
1004	Stand Alone DDT - all NCC activities beyond running DDT are suspended. This is primarily used by the software staff for debugging.

When the program is loaded from paper tape, it will start a short dialogue at the primary TTY. It will ask "ZERO TABLES..." (and optionally "ARE U SURE..."), expecting a Y or N response; and will ask "DATE & TIME..." and expect a proper numeric (decimal) response of the form MM/DD/YY HH:MM<return>. This dialogue is merely an aid to setting up locations 105 through 111, and starting at 1001 or 1000; it may be manually bypassed when convenient.

The switches on the NCC machine proper, not to be confused with the switches on the Lights display, perform the following functions:

SS1 Disables the display console. To be used in the event that the NCC must run in a machine without this hardware. This should never occur operationally.

SS3 Halts the NCC Host. The programs trys to bring the interface ready-line down, print all LOGGER messages, and other clean-up work.

SS4 As with the IMP, this senseswitch enables the ability of DDT to modify memory. Locations 2000 through 2007 may be modified even if SS4 is off.

If the program stops running correctly, the usual procedure is to restart or reload the program. Use the listing of the latest version of the NCC program (currently 131) to help you. If the program stops running correctly several times within a few hours, the machine probably has hardware troubles. To get a healthy NCC machine again, you will have to do things like moving I/O cables inside the machines. If you need help, this is the time to get it. The general restart, reload or machine-swap procedure is:

1) Take readings. If the NCC Host software honcho (currently Jon Cole) is around, see if he is interested in looking for software problems. If he is not around or if he believes that the failure is hardware, continue with this procedure.

2) Start stand-alone DDT at 1004. If DDT doesn't run, go to step 6.

3) Examine locations 105 thru 111 for the time and date when the NCC stopped working. Put in the correct values (with SS4 on). Type 1001S to restart the NCC program.

4) Wait about a minute to see if the program remains healthy. If the program doesn't run at all, go to step 6.

5) Use NCCVER (on Tenex E or C) to verify the core against the image file. If NCCVER doesn't work correctly (and

Tenex seems healthy), go to step 6. Compare the printout with the sample printout in the front of the NCC listing. Check new printouts with the code. If there are several clobbered instructions, go to step 6. If there are one or two clobbered instructions, manually restart stand-alone DDT, fix them, and go back to step 3. If there are no clobbered instructions, you are done. HUZZAH! Fill out the log book, turn off SS4, etc.

6) Have you reloaded the paper tapes into this machine recently? If you have, go to the next step. If not, do it now. Load "NCC,ver". Load "NCC ver patches". Answer the "ZERO..." question with "N", type-in the data and time, and go back to step 4.

7) Find another available machine, preferably the TESTIP.

8) Connect the NCC machine to the TESTIP via a "silver box" at 50 KB. Use modem number 1 on both machines.

9) Load the paper tape "NCCLOD" into the TESTIP.

10) Stop the NCC machine. Master clear both machines. Then, within 5 seconds:
  10a) Start the TESTIP at 3000
  10b) Start the NCC machine at 1002

11) Within 30 seconds, the TESTIP should halt at 3020 and the NCC machine should halt at at 3035 or 3036. If they do not halt correctly, repeat the preceding two steps several times before proceeding to the next step.

12) Disconnect both TESTIP modems.

13) Power down the NCC-Host and the TESTIP.

14) Move the Host cable from NCC-Host, drawer C1, slot E31 to TESTIP, drawer C1, slot E31.

15) Optional: The light-box and buzzer hardware can be moved to the TESTIP or not, depending on how much of a hurry you are in and how long you expect the machine switch to last. The NCC-program will run without the light-box hardware (if senseswitch 1 is on.)

  15a) Remove the IOBUS cables from the NCC-Host, drawer C1, slots F11 and G11. Remove the terminator cards (CC691B) from the TESTIP drawer C1, slots F11 and G11. Install them in the NCC-Host where you took out the IOBUS cables.

15b) Install the light-box cables where you removed the terminator cards. The cable paddles labeled "B" must go in slot F11 and "C" must go in slot G11.

15c) Make sure the light-box AC power does not come from the NCC-Host. It should come either from an under-floor power box or from the back of the TESTIP.

16) See if the TESTIP ports 2, 3, and 4 have "T" LIU cards in them. If not, switch LIU cards around. Write down what cards you have moved, so that they can be returned to their original slots later.

17) Unplug the Centronics printer AC and EIA cords. Move the printer closer to the TESTIP. Plug the EIA cord into port 2 and the AC cord into the back of the TESTIP.

18) Move the Inktronics printer EIA extension cord from the NCC-Host to the TESTIP port 4. (It should reach if you get a few feet of slack from under the floor.)

19) Move the "MSG-DUMP" (or "checksum TTY") extension cord from the NCC-Host to the TESTIP port 3.

20) Optional: The standard TESTIP TTY can be used as the NCC-program's "Summary and DDT" TTY. If you prefer, you can find the NCC-Host TTY cable which goes to the gray box labeled "Summary" (on the floor behind the model 35 TTY) and switch cables between the TESTIP TTY and the NCC-Host TTY.

21) Power up the TESTIP. The Centronics should be on. Push the "select" switch on the Centronics. If you moved the light-box, the buzzer should sound when you push tne TESTIP MASTER-CLEAR. If you did not move the light-box, put senseswitch 1 on.

22) Was the transfer process (steps 9 - 11) successful? If yes, go back to step 2. If not, load the two paper tapes. Answer the "ZERO..." and "ARE U SURE..." questions with "Y" (or start the program at 1000), and go back to step 4.

APPENDIX: TENEX COMMAND FILES

The following pages contain the files referenced in the section on program maintenance. They are driven by the RUNFIL or DO subsystems, and feed the text to a Tenex EXEC running in an inferior fork. The files listed here should be used merely as sample files; the program maintainer must list and use the up-to-date files in directory <NCC-PROG>.

```
;  <NCC-PROG>ARPA.ASSEMB;1   Thu 27-May-76 11:00AM          PAGE 1

;
; RUNFIL or DO file.  part 1 assembles ARPANET NCC program
;
RUNSTAT
DAYTIME

RUN <IMPSYS>DAPX16
FOO=ARPA,P0-1,P2-3,TASK,CLOCK,STUFF,MLC,LOGGER(NLSQ)
^Z
RESET

TIPCOPY FOO.ERR,
TNCC-TIP
D2
HAZ2


DIR FOO.*,
SIZE

RUNSTAT
DSKSTAT
DAYTIME
;
; end of part 1.  ARPANET NCC is assembled.
; if errors, delete foo.*;* and expunge before editing sources again.
```

```
;  <NCC-PROG>ARPA.ETC;4   Fri 25-Jun-76 4:41PM                PAGE 1

; DO file.  part 2 continues the ARPANET NCC building process.
;
RUNSTAT
USESTAT
DAYTIME
RENAME FOO.LST$NEW.LST;%.Version.

RENAME FOO.BIN$NEW.BIN;%.Version.

;                           ; assemble initializer.
RUN <IMPSYS>DAPX16
FOO,FOO=FOO.SYM,PRENCC.H16(Q)
^Z
RENAME FOO.SYM$NEW.SYM;%.Version.

RENAME FOO.LST$NEWIN.LST;%.Version.

RENAME FOO.BIN$NEWIN.BIN;%.Version.

;                           ; make up the concordance..
RUN <IMPSYS>CONH16
NEW.CON;%.Version.
ARPA.H16
P0-1.H16
P2-3.H16
TASK.H16
CLOCK.H16
STUFF.H16
MLC.H16
LOGGER.H16

; hack.  (doctor up the error file with the source versions used.)
TECO
;Y$NEW.CON;%.Version.$
0JSLOGGER.^D1LZKBJRCONCORDANCE FOR^Dassembled from^DZJI
^D;Y$FOO.ERR$
;U$NEW.ERR;%.Version.

;H$
DELETE FOO.ERR;*
```

```
;  <NCC-PROG>ARPA.ETC;4   Fri 25-Jun-76 4:41PM                PAGE 1:1

;                              ; print listings.
LIST NEW.ERR

;;;COPY NEW.LST LPT:
;;;
RUN <IMP>LISTER
NEW.CON
LPT:

T8
D

RUN <IMP>LISTER
NEWIN.LST
LPT:

T8
D


;                              ; punch the paper tape.
RUN <IMPSYS>P316S
Y
37

NEW.BIN;%.Version.

RENAME [-$NEW.PUN;%.Version.

;;;COPY NEW.PUN PTP:,
;;;I
;;;
TIPCOPY NEW.PUN,
TNCC-TIP
EA

DIR NEW.*,NEWIN.*,
SIZE

RUNSTAT
USESTAT
DAYTIME
DSKSTAT
;
; end of part 2.  ARPANET NCC is punched and listed.
```

```
;  <NCC-PROG>ARPA.RELEASE;2   Fri 25-Jun-76 4:44PM          PAGE 1

; DO file.  on "release day", renames the ARPANET NCC files.
;
RENAME NEW.BIN;%.Version. NCC.BIN;%.Version.

RENAME NEW.ERR;%.Version. NCC.ERR;%.Version.

RENAME NEW.PUN;%.Version. NCC.PUN;%.Version.

RENAME NEW.SYM;%.Version. NCC.SYM;%.Version.

RENAME NEWIN.BIN;%.Version. NCCIN.BIN;%.Version.

RENAME NEWIN.LST;%.Version. NCCIN.LST;%.Version.

RENAME NEW.CON;%.Version. <NCC>NCC-CONCORDANCE.LST;%.Version.

RENAME NEW.LST;%.Version. <NCC>NCC-LISTING.LST.LST;%.Version.

;;;RENAME PATNEW.PUN PATNCC.PUN
;;;

;
; done.
```

```
;  <NCC-PROG>ARPA.PATCH;7   Tue 29-Jun-76 1:57PM          PAGE 1

;
; assemble ARPANET NCC patch file
;
;;;RUN <IMPSYS>DAPX16
;;;PAT131=NCC.SYM,PAT131.H16(LQ)
;;;^Z
;;;TIPCOPY PAT131.LST,
;;;TNCC-TIP
;;;AZ2
;;;

RUN <IMPSYS>P316S
Y
37
36000
NCCIN.BIN
<IMPSYS>INVTRP-3231.BIN
<IMPSYS>INVTRP-3232.BIN
<TIP>INV365.BIN
<TIP>INV371.BIN
PAT131.BIN$

RENAME [-P$PATNCC.PUN

TIPCOPY PATNCC.PUN$,
TNCC-TIP
AE

TIPCOPY PATNCC.PUN$,
TNCC-TIP
AE

RUN <IMPSYS>IMGBIN
NCC.BIN
<IMPSYS>INVTRP-3231.BIN
<IMPSYS>INVTRP-3232.BIN
<TIP>INV365.BIN
<TIP>INV371.BIN
PAT131.BIN$
FOO.IMAGE

GET FOO.IMAGE
DEL FOO.IMAGE;*
CONNECT IMPDMP
SAVE 0 777777 IMAGE.NCC$

; done with ARPANET NCC patching.
```