INTERNETWORKING
in the STELLA/II Project

A.B. Bonito
N. Celandroni
E. Ferro
M. Mannocci
E. Perotto

Report C83-26

Istituto CNUCE
PISA
November 1983

0. NOMENCLATURE

T.M.	=	TRANSPORT MANAGER
N.T.H.	=	NETWORK-TYPE HANDLER
L.H.	=	LINE HANDLER
N.H.	=	NETWORK-HANDLER
N.N.	=	NET NUMBER
LOC.HDR.	=	LOCAL HEADER
INT.HDR.	=	INTERNET HEADER
SAT.HDR.	=	SATELLITE HEADER
UNI.HDR.	=	UNIVERSE HEADER
G.TM.A.	=	GLOBAL T.M. ADDRESS
DRV	=	DRIVER
NGA	=	NEXT GATEWAY ADDRESS
C.R.	=	CAMBRIDGE RING

1. INTERNETWORKING STRUCTURE

The Internetworking architecture was already described in the paper STELLA/01/81. In the following a description is given of the INTERNET task which implements the internetworking functions.

INTERNET is the name of the internetworking task; it is written in the BCPL high-level language and includes routines and tables which are always valid regardless of the gateway on which it is running, and routines and tables which are strictly dependent on the LAN configuration of the gateway.

These "gateway-dependent" parts are in files whose names begin with INT and finish with the first 3 characters of the name of the city where the gateway resides.

INTERNET has on its top the Transport Managers and at its bottom the drivers which handle the physical line(s) of the various networks. Each of these processes may use different data format and protocols, so it is difficult to realize a <u>unique</u> task able of speaking different line-protocols and handling data with different formats. Moreover, the processing of the completed events must be FIFO. So INTERNET has been designed as a unique task but logically divided into 3 main parts:

- the "T.M. interface" entities. These are routines <u>specialized for every type of T.M.,</u> able of interpreting data coming from the associated T.M.

- the "line-handler" entities. These are routines <u>specialized for every driver (through which a LAN is connected),</u> able to communicate with the driver as well as to interpret data to/from a certain LAN.

- the "main body" or "brain" of the entire system which is <u>able to select the correct T.M. interface entity or the correct line-handler in a complete transparent way.</u>
  This part of software also performs functions strictly typical of the internetworking, like verifying if the destination net has been reched or not, if the target T.M. is local or remote (only in the case the destination net has been reached), to check and handle the TX/RX operation ends and so on. It is also able to handle the "options" specified in the Internet Header part of the data. In this first implementation only the ECHO/ECHO REPLY options are supported.

Every communication between INTERNET and the Transport Managers (via the "T.M.-interface" routines) and between INTERNET and the physical drivers (via the "line-handlers" routines) is made by using the ITCALL routine.

In the current implementation of INTERNET it is envisaged to include the following "type of T.M.":

```
 CERNET type
 TAPE   type (same application as in STELLA/I; not implemented)
 SSP    type (C.R. single-shot protocol)
 TALK   type (for internetworking messages exchange)
```

and the following line-handlers (with the corresponding LAN associated):

```
     SATELLITE LINE-HANDLER (Satellite network)
     LK        LINE-HANDLER (Cernet network)
     VU        LINE-HANDLER (Cambridge Ring network)
     X25       LINE-HANDLER (Euronet network, through
                             which the INET network,
                             for example, is connected)
```

INTERNET creates a big "IRG" region (in the GEN partition) where a default buffer pool is allocated. The buffers of this pool will be used if and only if a congestion state is reached in the data RX phase of any line-handler. In no case a booking of a buffer of the default pool is allowed (see the Getbuffer routine).

The general strategy is that the T.M.(s) must only send data to INTERNET for TX operations without any definition for the RX buffers. In fact a T.M. will be alerted by INTERNET when data are coming for it and the T.M. will map the buffer delivered to it by Internet.

The delivering of a buffer by a T.M. to INTERNET to be sent on a line must be interpreted by the T.M. as if the TX operation was successfully completed (remember that INTERNET works on a datagram base). TX errors must be recovered by the end-to-end protocol between T.M.

Each line-handler routine must define a buffer pool for data receiving from the handled physical line(s).

The satellite line-handler is a little bit different from the other line-handlers because it must create as many RX pools as the number of the "environments" defined in the gateway and must declare them in the PLIST as they will be automatically used by the STELLA(ST) driver on data receiving from satellite. This is because the satellite local header contains a byte (PROTOCOL byte) in which the nature of the data unit is specified (i.e. the "environment"). On the basis of the protocol byte, the ST driver is able to write the incoming data directly in a free buffer of the pool created for the specified "environment".

"Environment" or "type of T.M." has the same meaning.

All the RX pools created by the line-handlers plus the default pool are build inside the "IRG" region created by the "brain" of INTERNET. The use of the INTERNET default pool is the following: on data receiving from a line, if no free buffer is available in the relative RX pool, an attempt is made to get a free buffer from the default pool, just to try to save data instead of losing them. If no buffer is available even in the default pool, the data are irremediably lost. The end-to-end protocol between T.M.s will ask again for retransmission of the lost data.

The general structure of INTERNET can be so summarized:

```
                 +------+                 +------+
                 | T.M. |.................| T.M. |
                 |  A   |                 |  Z   |
                 +------+                 +------+
INTERNET +-------|------------------------|-----------+
         |       |                        |           |
         |  +----------+             +----------+     |
         |  |T.M.      |             |T.M.      |     |
         |  |Interface|.............|Interface|     |
         |  |A         |             |Z         |     |
         |  +----------+             +----------+     |
         |       |                        |           |
         | +------------------------------------+     |
         | |      MAIN BODY OF INTERNET         |     |
         | +------------------------------------+     |
         |       |            |              |        |
         | +----------+ +----------+   +----------+   |
         | | LINE     | | LINE     |   | LINE     |   |
         | | HANDLER  | | HANDLER  |...| HANDLER  |   |
         | | #1       | | #2       |   | #n       |   |
         | +----------+ +----------+   +----------+   |
         +-------|------------|--------------|--------+
                 |            |              |
                 +--+         +--+           +--+
                    |            |              |
                    |            |              |
                    V            V              V
```

The T.M. INTERFACES must know the structure of the data coming from the corresponding T.M.(s).

The LINE HANDLERS must be able to send/receive data coming from the physical lines which they handle on the basis of the protocol of the network they belong to.

In the STELLA-II implementation of the Internetworking, the T.M. INTERFACES and the LINE HANDLERS are build as <u>routines</u> inside the INTERNET task.

All the boxes shown in the previous figure have been thought as different processes which may or may not run on the same physical machine.
Internet is a process interfacing networks and applications; it performs the gateway functions. The applications of Internet are the Transport Managers (T.M.). The internet system makes all the possible switchings of the data between any couple of interfaced entities.
The T.M. interfaces and the subnetworks interfaces (line-handlers) are necessary to adapt already existing entities to the extabilished internetworking interface.

Data units, at the subnetwork level and at the transport manager level, may have one of the two formats shown in the following figure:

```
                              +--------------------------------------------
                              |  TRANSPORT         |
                              |   PROTOCOL         |          DATA
                              |   HEADER           |
                              +--------------------------------------------
                                          A
                                         | |
  I.H. not present                       | |        (normal data unit)
                                         | |
+--------------------------------------------------------------------------
|  SUBNETWORK  |
|  PROTOCOL    |
|  HEADER      |
+--------------------------------------------------------------------------
                                         | |
  I.H. present                           | |
                                         | |
                                          V
                              +-------------------------------------------
                              |  INTERNET      |  TRANSPORT   |
                              |  HEADER        |  PROTOCOL    |     DATA
                              |                |              |
                              +-------------------------------------------
```

The first header contained in the data frame is always the sunetwork protocol header in which a binary information specifying whether the internet header is present or not is contained. After crossing the interfaces in the direction towards the internetworking, data are always normalized to contain the internet header while the subnetwork protocol header is stripped off.

The task of the internetworking process is to perform the routing of the data. On the basis of the destination address, the internet process decides where to route the data unit. The routing is performed looking up into fixed routing tables.

When the target application or the target subnetwork is reached, the relative interface must strip off the internet header and the associated subnetwork protocol header is added before delivering the data unit to the target entity.
If the destination net is not yet reached, data are routed to the next network to cross for reaching the destination which is contained in the routing tables. The needed subnetwork protocol header is anyway added by the interface. This header will bring the information that the internet header is present. Data keep the internet header until the destination is reached.

2. TABLES used by INTERNET

2.1 LOCal processes GENeral TABle:

LOCGENTAB: 6 words *(number of local processes known by INTERNET)

1 word	1 word	1 word	2 words	1 word
PROCESS-ID	TYPE	NET NUMBER	LOCAL-ID FOR T.M.	OFFSET
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
-1	-1	-1	-1	.

PROCESS-ID : ID of a process.
If most significant bit is on, it is assumed to be a DRV; if off, it is assumed to be a task.

TYPE : T.M. (task or driver)
LINE (task or driver). The satellite is included here.

NET NUMBER : number of the network the process bolongs to.

LOCAL-ID FOR T.M. : 2 words containing the local-id of a T.M. It is the address inside the network it belongs to.

OFFSET : offset in NTRAVEC of the word containing the T.M.-interface routine addres (if TYPE=T.M.) or the line-handler routine address (if TYPE=LINE) able to handle data to/from the related PROCESS-ID.

This table must have one entry for every process of the STELLA-II system which can send/receive data which are analized by the INTERNET task.

The last row values are set to -1 as plug of the table.
The LOCGENTAB table is dependent on the gateway where INTERNET is running.

2.2 NETs TABLE:

NETTAB:

1 word	1 word	2 words	2 words
DESTINATION NET NUMBER	NEXT NET NUMBER	NEXT GATEWAY ADDRESS	CURRENT GATEWAY ADDRESS
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
-1	-1	-1 -1	-1 -1

DESTINATION NET NUMBER : number of the destination net

NEXT NET NUMBER : number of the next net which the packet must cross to reach the destination net.
DESTINATION NET = NEXT NET NUMBER means that the DESTINATION NET is reached.

NEXT GATEWAY ADDRESS : address of the "next gateway" as seen from the "next net".

CURRENT GATEWAY ADDRESS: address of the "current gateway" as seen from the "next net". If the destination net is reached, the second word of the current gateway address contains the offset in NTRAVEC of the routine handling the data belonging to the corresponding net.

The last row values are set to -1 as plug of the table.

Every NEXT NET NUMBER in NETTAB must have the line-handler routine address in the corresponding offset of the NTRAVEC vector.

The NETTAB table is <u>dependent</u> of the gateway where INTERNET is running.

2.3 Net-Type dependent Routine Address Vector

NTRAVEC

```
+--------+--------+--------+-----------------+--------+
| SPARE  | ADDR1  | ADDR2  | ............... | ADDRn  |
+--------+--------+--------+-----------------+--------+
Words  Ø        1        2                     n
```

This vector is as many words long as the number of line-handlers present in the configuration of the station plus the number of T.M. interface routines.

Word Ø of NTRAVEC is spare.

The first "NTLEN" words of NTRAVEC must contain the addresses of the line-handler routines. The following words must contain the addresses of the T.M. interface routines. All these routines will be invoked by INTERNET with different input parameters on the basis of the work they have to do.

Generally, the current routine address used is stored in the NTDRA global.

At the very beginning, the first NTLEN words of NTRAVEC contain the address of the default STOP routine. As soon as the START LINE(s) command is issued by the operator, the correct line-handler routine address(es) is got by the LINEHTAB vector and stored at the correct offset in NTRAVEC. This offset must be from 1 to NTLEN.

The NTRAVEC vector and its values are independent of the gateway where INTERNET is running.

2.4. LINE-Handler Table

LINEHTAB is a vector as many words long as the number of the line-handler routines.

In each word of this vector a line-handler routine address is stored.

Word Ø contains the STOP routine address.

On the basis of the "start line(s)" command, from LINEHTAB the correct line-handler routine address is got and copied in the corresponding position of NTRAVEC.

This vector is independent of the gateway where INTERNET is running.

2.5 Task Status Table

TSKSTATUSTAB

```
                   STATUS
                +---------+
entry:  Ø       | SPARE   |
                +---------+
        1       |ACTIVE/  |
                |ABORTED  |
                +---------+
        2       | SPARE   |
                +---------+
        3       | SPARE   |
                +---------+
        .       |ACTIVE/  |
                |ABORTED  |
                +---------+
        .       |    .    |
        .       |    .    |
        .       |    .    |
        .       |    .    |
                +---------+
                |ACTIVE/  |
        n       |ABORTED  |
                +---------+
```

This table is as many words long as the number of tasks present in the STELLA2 system.

Each word is accessed by using the PROCESSID value as word offset. It contains the status (aborted/active) related to that task.

As :   PROCESS-ID = Ø is FREEQUEUE
        PROCESS-ID = 2 is STELLAREQ
        PROCESS-ID = 3 is STELLADATA

they are never used as "fromprocess-id" (PROCESS-ID = 3 = STELLADATA is used when a block is enqueued, via ITCALL, to STELLADATA, but FROMPROCESS = 1 is used a block is enqueued to a process by STELLADATA), the entries Ø, 2 and 3 of TSKSTATUSTAB are spare.

STATUS is set up to ABORTED when Internet receives a block enqueued by himself. It means that it tried to enqueue via ITCALL a block to a certain process which was decided (by the ITCALL itself) to be aborted being its receiving queue too long.

STATUS is set up to ACTIVE when Internet receives a block enqueued by the Controller process, containing as first word the id of the process become active.

At the start time, all the words are initialized to "ACTIVE".

3. BLOCKS-FORMAT FOR INTERCOMMUNICATIONS

3.1) From T.M. to INTERNET

```
+---------------------+
| INTERNET <---- T.M. |
+---------------------+

     +-------------------------+
     |          LINK           |
     +-------------------------+
     |        BDB ADDR.        |
     +-------------------------+
     |          CODE           |
     +-------------------------+
     |   DATA LENGTH (bytes)   |
     +-------------------------+
     | OPTION/OPTION BDE ADDR. | (1)
     +-------------------------+
     |          SPARE          | may be used to pass the
     +-------------------------+ data word offset
     |      FROM PROCESS       |
     +-------------------------+
```

3.2) From LINE to INTERNET (also STELLA process is included here)

```
+---------------------+
| INTERNET <---- LINE | (also the STELLA driver is included
+---------------------+

     +-------------------------+
     |          LINK           |
     +-------------------------+
     |        BDB ADDR.        |
     +-------------------------+
     |        I/O CODE         |
     +-------------------------+
 (*) |   DATA LENGTH (bytes)   |
     +-------------------------+
     |          SPARE          |
     +-------------------------+
     |        FUNCTION         |
     +-------------------------+
     |      FROM LINE-ID       |
     +-------------------------+
```

(*) The DATA LENGTH value is expressed in <u>words</u> when FROM PROCESS = STELLA.

3.3) From INTERNET to T.M.

```
+----------------------+
| T.M. <---- INTERNET |
+----------------------+

        +--------------------------+
        |           LINK           |
        +--------------------------+
        |         BDB ADDR.        |
        +--------------------------+
        |    RX Operation CCDE     |
        +--------------------------+
        |   DATA LENGTH (bytes)    |
        +--------------------------+
        |       DATA OFFSET        |  (*)
        +--------------------------+
        |    FUNCTION = RXEND      |
        +--------------------------+
        |         INTERNET         |
        +--------------------------+
```

(*) this is the word-offset of the data inside the buffer.

3.4) From INTERNET to LINE ≠ STELLA

```
+----------------------+
| LINE <---- INTERNET |
+----------------------+

        +--------------------------+
        |           LINK           |
        +--------------------------+
        |         BDB ADDR.        |
        +--------------------------+
        |         RESERVED         |
        +--------------------------+
        |         RESERVED         |
        +--------------------------+
        |        TRANSPARENT       |
        +--------------------------+
        |         FUNCTION         |
        +--------------------------+
        |       FROM PROCESS       |  (2)
        +--------------------------+
```

3.5) From INTERNET to LINE = STELLA

```
+---------------------+
| STELLA <--- INTERNET|
+---------------------+

      +-------------------------+
      |          LINK           |
      +-------------------------+
      |        BDB ADDR.        |
      +-------------------------+
      |  DATA OFFSET (words)    |
      +-------------------------+
      |  DATA LENGTH (in words) |
      +-------------------------+
      |       TRANSPARENT       |
      +-------------------------+
      |  BACK-USER = INTERNET   |
      +-------------------------+
      |        INTERNET         |
      +-------------------------+
```

3.6) From the BUFfer MANAGER to INTERNET

```
+------------------------+
| INTERNET <---BUFMANAGER|
+------------------------+

      +-------------------------+
      |          LINK           |
      +-------------------------+
      |        PDB ADDR.        |
      +-------------------------+
      |        BDB ADDR.        |
      +-------------------------+
      |       TRANSPARENT       |
      +-------------------------+
      |          SPARE          |
      +-------------------------+
      |          SPARE          |
      +-------------------------+
      |       BUFMANAGER        |
      +-------------------------+
```

3.7) From INTERNET to INTERNET

```
+----------------------+
|INTERNET<----INTERNET|
+----------------------+

          +--------------+
          |     LINK     |
          +--------------+
          |ID OF PROCESS | a process is declared
          |  "ABORTED"   | "aborted"
          +--------------+
          |    SPARE     |
          +--------------+
          |    SPARE     |
          +--------------+
          |    SPARE     |
          +--------------+
          |    SPARE     |
          +--------------+
          |   INTERNET   |
          +--------------+
```

This block is enqueued to INTERNET by the ITCALL itself when INTERNET tries to enqueue a block to a process with a too long receiving queue. In this case the target process is declared aborted by the ITCALL and the calling task (Internet) is worned.

3.8) From CONTROLLER to INTERNET

```
+--------------------------+
|INTERNET <---- CONTROLLER|
+--------------------------+

          +--------------+
          |     LINK     |
          +--------------+
          |ID OF PROCESS | a process is declared
          |  "ACTIVE"    | "active"
          +--------------+
          |    SPARE     |
          +--------------+
          |    SPARE     |
          +--------------+
          |    SPARE     |
          +--------------+
          |    SPARE     |
          +--------------+
          |  CONTROLLER  |
          +--------------+
```

Note

1) This word contains either the option itself (if one word) or the BDB address of a buffer containing the options.
If no option needs BIT 15 on, we must force bit 15 to the following meaning:

BIT 15 ON : option itself
BIT 15 OFF : BDB address.

2) If LINE is a DRV, this block is enqueued by INTERNET to its receiving queue in the QLIST because the ITCALL is QIO type.
If LINE is a task this block is enqueued to the destination process. Therefore, in the first case, the last word contains the LINE-ID; in the second case it contains INTERNET as from process.

4. Most Important GLOBALS used:

LOCIDVEC : contains the "local-id" of a process. It is a 3 words vector

USERBUF : virtual address of the user buffer as received by INTERNET. It will be modified by the line-handler routines or by the T.M. interface routines.

SAVEUSBUF : original virtual address of the user buffer as received by INTERNET. It is never modified.

DATALENGTH: Length of the sent/received data. It is expressed in bytes.

BDBADDRESS: current BDB address.

NTDRA : contains the address of the current line-handler or T.M. interface routine invoked.

NTENTRY : entry in NETTAB

LGTENTRY : entry in LOCGENTAB

5. KEYWORDS to call the line-handler routines or the T.M. interface routines.

NOTES: The BCPL global USERBUF points to the beginning of the data.

a) SET.INTHDR

The routine must decide if the internet header is already present in the data buffer or not. If not, the local header must be stripped off and the Internet header build.
Datalength must be updated. On exit (if all OK), USERBUF points in any case to the INT.HDR.

Output :	IH.ALREADY.EXIST	the internet header is already present.
	< 0	some errors occur during the address translation to build the INT.HDR. In this case, the buffer will be released.
	> 0	all OK. The INT.HDR has been added.

b) LCC.HDR

The routine must strip off the INT.HDR and build the local header. Datalength must be updated. On exit, if all OK, USERBUF will point to the local header.

Output : >Ø address translation is OK

<Ø some errors occurred during the addresses translation. In this case the buffer will be released.

c) NEXTNET.LOCHDR

The routine must build the local header of the net the data are going to cross and add it on the top of the INT.HDR.

DATALENGTH must be updated and, on exit, USERBUF must point on the top of the added local header.

Output : >Ø addresses translation is OK and the local header has been added.

<Ø errors occurred during the addresses translation. In this case the buffer will be released.

d) BUF.GET

The routine booked a free buffer of its receiving pool. The BDB address of the available buffer is returned in the second word of the RESULT vector.

e) READ.LINE

An RX operation was completed and a new RX operation must be issued on the line.

f) INI.LINE

The line-handler routine is requested to create the RX pool(s) for its line(s) and to perform all the operations necessary to initialize the physical line(s). Generally, a pending RX operation is issued at the end of the initialization.

g) STOP.LINE

The line-handler routine is requested to perform all the operations necessary to stop the physical line(s).
The blocks used to create the RX pool(s) and the BDBs are returned to the free-block queue.

h) SEND.DATA

The routine must send in the appropriate way the data on the (selected) line.

k) YOUR.CODE

It is responsability of the routine to handle this case. The routine is invoked with this keyword by the "brain of INTERNET" when the I/C function code of the completed I/O operation is not supported (normally there is an error in the function code). In other words, it is a default case.

l) DASM.ABORTED

This keyword is valid only for the satellite line handler. See 8.2.

6. INTERNET HEADER structure in the first implementation of the internetworking:

```
(+ significant)  15              8|7                 Ø (-significant)
 bit            +-----------------------------------+
                |      DATA    UNIT    LENGTH       | word: Ø
                +-----------------+-----------------+
                | VERSION         |       IHL       |  "    1
                +-----------------+-----------------+
                | USER PROTOCOL   | QUALITY OF SERVICE|     2
                +-----------------+-----------------+
                |             IDENTIFIER            |       3
                +-----------------------------------+
                |___                                |       4
                |___      SOURCE   ADRESS           |       5
                |                                   |       6
                +-----------------------------------+
                |___                                |       7
                |___    DESTINATION  ADRESS         |       8
                |                                   |       9
                +-----------------------------------+
                |               SPARE               |      10
                +-----------------------------------+
                |            OPTION MASK            |      11
                +-----------------------------------+
                |          HEADER CHECKSUM          |      12
                +-----------------------------------+
```

where:

DATA UNIT LENGTH : data length + internet header length (in bytes)

VERSION : version number of the INTERNET implementation. Current version number is Ø1.

IHL : internet header length. It must be expressed in bytes. Current length is 26 bytes.

USER PROTOCOL : "transport manager type". Envisadged types are:
Cernet =1; Tape =2; SSP =3; Talk =4.

QUALITY OF SERVICE : Not implemented. SPARE byte.

IDENTIFIER : Not implemented. SPARE word.

SOURCE ADDRESS : First word is for the source network number.
The following two words describe the source local address.

DESTINATION ADDRESS: First word is for the destination network number.
The following two words describe the destination local address.

OPTION MASK : Mask of the options present in the internet-header. In the first version of INT.HDR. only the ECHO/ECHO REPLY options are implemented. Thus, the content of this word may be:

#X0001 --> ECHO
#X0002 --> ECHO REPLY
0 --> NO OPTION PRESENT

HEADER CHECKSUM : Not implemented in the first version of INT.HDR.

7. GATEWAY-DEPENDENT TABLES

Chapter 7 is dedicated to the description of the tables used by the Internet "brain" part for the PISA gateway and the GENEVA gateway. Common tables are specifically indicated.

7.1 NETS CONFIGURATION (the same for every gateway)

NET #1	:	SATELLITE NET
NET #2	:	EURONET (I)
NET #3	:	PISA-CERNET NET (I)
NET #4	:	PISA-CAMBRIDGE RING NET (I)
NET #5	:	UNIVERSE NET (U.K.)
NET #6	:	CERN-CERNET NET (C.H.)
NET #7	:	ISPRA-CERNET (I)
NET #8	:	DUBLIN-NET (IRELAND)
NET #9	:	FRASCATI-NET (I)
NET #10	:	CERN-CAMBRIDGE RING NET (C.H.)
NET #11	:	PISA TALK NET
NET #12	:	GENEVA TALK NET
NET #13	:	FRASCATI TALK NET
NET #14	:	DUBLIN TALK NET
NET #15	:	ISPRA TALK NET
NET #16	:	RUTHERFORD TALK NET

7.2 LOCGENTAB for PISA configuration

PROC.-ID	PROC.-TYPE	NET-NUMBER (1)	LOCAL-ID-FOR T.M.	OFFSET (2)
CERNET	T.M.	3	0, #x1301	CER.IFOFF
TAPE	T.M.	1	-1, -1 (3)	TAP.IFOFF
SSPROC	T.M.	4	64, 1	SSP.IFOFF
TALKPROC	T.M.	11	1, 1	TAL.IFOFF
STELLABURST	LINE	1	0, 0	SATOFF
X25HAND	LINE	2	0, 0	X25OFF
LK-DRV-INPUT	LINE	3	0, 0	LKOFF
LK-DRV-OUTPUT	LINE	3	0, 0	LKOFF
VU-DRV-INPUT	LINE	4	0, 0	VUOFF
VU-DRV-OUTPUT	LINE	4	0, 0	VUOFF
-1	-1	-1	-1, -1	-1
1W	1W	1W	2W	1W

(1) Number of the network to which the process belongs. Each Net-number must be concorded among all the users of the Internetworking. See table 7.1.

(2) Offset in NTRAVEC to find the corresponding line-handler routine address or T.M.-interface routine address. See values in 7.5.

(3) Tape-to-tape transfer as in STELLA-I is not implemented.

7.2.1 LOCGENTAB for GENEVA configuration

PROC.-ID	PROC.-TYPE	NET-NUMBER (1)	LOCAL-ID-FOR T.M.	OFFSET (2)
CERNET	T.M.	6	0, #x320A	CER.IFOFF
TAPE	T.M.	1	-1, -1 (3)	TAP.IFOFF
SSPROC	T.M.	10	64, 1	SSP.IFOFF
TALKPROC	T.M.	12	1, 1	TAL.IFOFF
STELLABURST	LINE	1	0, 0	SATOFF
LK-DRV	LINE	6	0, 0	LKOFF
X25HAND	LINE	2	0, 0	X25OFF
VU-DRV-INPUT	LINE	10	0, 0	VUOFF
VU-DRV-OUTPUT	LINE	10	0, 0	VUOFF
-1	-1	-1	-1, -1	-1
1W	1W	1W	2W	1W

7.2.2 LOCGENTAB for ISPRA configuration

PROC.-ID	PROC.-TYPE	NET-NUMBER (1)	LOCAL-ID-FOR T.M.		OFFSET (2)
CERNET	T.M.	7	0, #X1701		CER.IFOFF
TAPE	T.M.	1	-1, -1	(3)	TAP.IFOFF
TALKPROC	T.M.	15	1, 1		TAL.IFOFF
LK-DRV	LINE	7	0, 0		LKOFF
X25HAND	LINE	2	0, 0		X25OFF
-1	-1	-1	-1, -1		-1
1W	1W	1W	2W		1W

7.3 <u>NETTAB for PISA configuration</u>

DEST. NET	NEXT NET	NEXT GATEWAY ADDRESS (4) (5)	CURRENT GATEWAY ADDRESS (4) (5)	
1	1	0, 0	0, SATOFF	DESTINATION NET REACHED
2	2	0, 0	0, X25OFF	
3	3	0, 0	0, LKOFF	
4	4	0, 0	0, VUOFF	
5	1	0, #X0011	0, #X0022	
6	1	0, #X0001	0, #X0022	
7	2	0, #X800A	0, #X800B	
8	2	0, #X8003	0, #X800B	
9	1	0, #X0023	0, #X0022	
10	1	0, #X0001	0, #X0022	
11	11	0, 0	0, TAL.IFOFF	REACH
12	1	0, #X0001	0, #X0022	
13	1	0, #X0023	0, #X0022	
14	2	0, #X8003	0, #X800B	
15	2	0, #X800A	0, #X800B	
16	1	0, #X0011	0, #X0022	
-1	-1	-1, -1	-1, -1	
1W	1W	2W	2W	

(4) Satellite gateway addresses

```
#X01  = GENEVA satellite gateway address (CH)
#X11  = RUTHERFORD satellite gateway address (UK)
#X22  = PISA satellite gateway address (I)
#X23  = FRASCATI satellite gateway address (I)
```

(5) EURONET Logical Units

```
#X800A  = Logical Unit to be used to reach the ISPRA
          gateway via EURONET;
#X8003  = Logical Unit to be used to reach the DUBLIN
          gateway via EURONET;
#X800B  = Logical Unit to be used for the PISA Gateway
          via EURONET.
```

7.3.1 NETTAB for GENEVA configuration

DEST. NET	NEXT NET	NEXT GATEWAY ADDRESS (4)	CURRENT GATEWAY ADDRESS (4)	
1	1	0, 0	0, SATOFF	(**)
2	1	0, #X0022	0, #X0001	
3	1	0, #X0022	0, #X0001	
4	1	0, #X0022	0, #X0001	
5	1	0, #X0011	0, #X0001	
6	6	0, 0	0, LKOFF	(**)
7	1	0, #X0022	0, #X0001	
8	1	0, #X0022	0, #X0001	
9	1	0, #X0023	0, #X0001	
10	10	0, 0	0, VUOFF	(**)
11	1	0, #X0022	0, #X0001	
12	12	0, 0	0, TAL.IFOFF	(**)
13	1	0, #X0023	0, #X0001	
14	1	0, #X0022	0, #X0001	
15	1	0, #X0022	0, #X0001	
16	1	0, #X0011	0, #X0001	
-1	-1	-1, -1	-1, -1	
1W	1W	2W	2W	

(**) Destination net reached.

7.3.2 NETTAB for ISPRA configuration

DEST. NET	NEXT NET	NEXT GATEWAY ADDRESS (5)	CURRENT GATEWAY ADDRESS (5)	
2	2	0, 0	0, X250FF	(**)
3	2	0, #X800B	0, #X800A	
4	2	0, #X800B	0, #X800A	
5	2	0, #X800B	0, #X800A	
6	2	0, #X800B	0, #X800A	
7	7	0, 0	0, LKCFF	(**)
8	2	0, #X8003	0, #X800A	
9	2	0, #X800B	0, #X800A	
10	2	0, #X800B	0, #X800A	
11	2	0, #X800B	0, #X800A	
12	2	0, #X800B	0, #X800A	
13	2	0, #X800B	0, #X800A	
14	2	0, #X8003	0, #X800A	
15	15	0, 0	0, TAL.IFOFF	(**)
16	2	0, #X800B	0, #X800A	
-1	-1	-1, -1	-1, -1	
1W	1W	2W	2W	

7.4 NTRAVEC

```
+--------+----------+--------+--------+--------+
|STOP    |SATELLITE|X25      |LK      |CR      |
|ROUTINE|LINEH     |LINEH   |LINEH   |LINEH   |
|ADDR.   |ROUTINE   |ROUTINE|ROUTINE|ROUTINE|
|        |ADDRESS   |ADDRESS|ADDRESS|ADDRESS|
+--------+----------+--------+--------+--------+
  0        SATOFF     X25OFF   LKOFF   VUOFF

+----------+----------+----------+----------+
|CER.TM.   |TAPE.TM.  | SSP.TM   |TALK.TM   |
|INTERFACE|INTERFACE|INTERFACE|INTERFACE|
|ROUTINE   |ROUTINE   |ROUTINE   |ROUTINE   |
|ADDRESS   |ADDRESS   |ADDRESS   |ADDRESS   |
+----------+----------+----------+----------+
CER.IFOFF  TAP.IFOFF SSP.IFOFF TAL.IFOFF
```

SATOFF	= 1	contains the SATELLITE line handler routine address
X25OFF	= 2	contains the X25 line handler routine address
LKOFF	= 3	contains the LKDRV line handler routine address (CERNET)
VUOFF	= 4	contains the VUDRV line handler routine address (CAMBRIDGE RING)
CER.IFOFF	= 5	contains the CERNET T.M.-interface routine address
TAP.IFOFF	= 6	contains the TAPE T.M.-interface routine address.
SSP.IFOFF	= 7	contains the SSP T.M.-interface routine address.
TAL.IFOFF	= 8	contains the TALK T.M.-interface routine address.

The correct line-handler routine address is copied from the analojous LINEHTAB as soon as the line-handler is requested to start.

8. INTERNET: SATELLITE LINE HANDLER STRUCTURE

```
              +------+      +------+      +------+
              |T.M.  |      |T.M.  |      |T.M.  |
              +------+      +------+      +------+
                 |             |             |
+--------------------------------------------------------+
|                  "BRAIN" OF INTERNET                   |
|                                                        |
|                                 +----------------------|
|                         +--->|  SATELLITE LINE-        |
|                         |    |  HANDLER ROUTINE        |
+------------------------|------------------------------+
             ITCALL to   |                |  ITCALL from INTERNET
             INTERNET    |                |
                         |    +-----------V-----------+
                         |    |  STELLA PROCÉSS       |
                         +----|   (ST DRV)            |
                              +-----------------------+
                                          |                LDC
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                          |
                              +-----------------------+
                              |  COMMUNICATION        |
                              |   INTERFACE           |
                              |    MODULE             |
                              +-----------------------+
                                          |
                                          |
                                          +--+
                                             |
                                             |
                                             V
                                     OTS/SATELLITE
```

8.1 Preliminary Remarks

The satellite network is made up of the CTS satellite and of the following components:

--the LDC (Link Driving Computer);
--the CIM (Communication Interface Module) between the LDC and the LINKABIT. The CIM performs CPU intensive and time critical functions for the LDC and also permits the use of disparate LDC types at different sites. Moreover, multiple HDLC frames can be packed into a single window;
--the modem/codec between 70 MHz two phase modulate and 1 MHZ digital signals with half rate coding and VITERBI decoding;
--the antenna (3 meter) and the R.F. system for transmission and reception at 14 and 11 GHz respectively, with low level interface at 70 MHz.

The real transmission speed on satellite is 1 Mbit/sec. The satellite network can function simultaneously both as a transit network with attached internetwork gateways and as a network per se supporting directly applications of differing capabilities and requirements. The satellite network is also requested to accomodate packets with a wide range of lengths, allowing a short packet containing a message with a small number of characters to coexist efficiently with a packet containing a long message or perhaps several host-multiplexed messages. To achieve these aims, a simmetric, dynamic-allocation TDMA channel access scheme was developed. It is implemented by the DASM process (see CNUCE Internel Report N.C83-21).

The satellite network is interfaced by Internet via the satellite line handler routine.

The satellite line-handler functions are performed by the SAT.ROUTINE of the INTERNET task. The address of this routine, when the satellite line is started, is stored in the NTRAVEC vector at the SATOFF offset.

The ST driver is seen by INTERNET as a task; so the communication between them is via the ITCALL routine used for inter-task-communication. The ST driver is able to receive data for every T.M. whose RX PDB address is stored at the correct offset in the PLIST common area (see CNUCE Internal Report N. C83-22).

The format of the satellite local header is the following:

```
     +----------------------+
 1   | SOURCE   |   DEST    | Ø
     +----------------------+
 3   | CONTROL  | PROTOCOL  | 2      byte offsets
     +----------------------+
```

where PROTOCOL is subdivided in such a way that the 3 less significant bits indicate the "environment", i.e. the type of the incoming/outgoing transport data unit.

8.2 SATELLITE LINE HANDLER BEHAVIOUR

In the following the actions performed by the satellite line handler routine are described. The different behaviour is on the basis of the input "keyword" parameter.

a) SET.INTHDR

Certainly incoming data have on top the satellite local header and contain also the INT.HDR.

- the satellite local header is stripped off
- USERBUF points on the INT.HDR
- DATALENGTH is decreased.
- the value IH.ALREADY.EXISTENT is returned.

b) LOC.HDR

- no operation because the satellite is seen as a "transport" network, i.e. data with a particular "satellite" format do not exist.

c) NEXTNET.LCCHDR

The satellite local header must be added on the top of the outgoing data.

- the satellite local header is added
- USERBUF points to the satellite local header
- DATALENGTH is increased.

d) BUF.GOT

- no operation because no GETBUF routine is invoked by the satellite line-handler for an RX buffer acquisition. The ST driver performs it automatically.

e) READ.LINE

- no operation (automatically performed by the STDRV driver)

f) INI.LINE

- RX pools are created (1 for every defined "environment"), initialized for the ST driver and declared in the correct positions of the PLIST.
- a request for an initial large-data slot (20 msec) is enqueued to DASM.

g) STOP.LINE

- the created PDB(s) and BDB(s) are returned to the free block queue
- a request to relinguish the large-data slot is enqueued to DASM.

h) SEND.DATA

- an ITCALL is performed to STELLADATA and the format of the block enqueued to STELLADATA is as in 3.5 where the TRANSPARENT word is = 0.
- a check is made on the number of the blocks enqueued to STELLADATA. If it is greater than a maximum, a THROUPUT request is enqueued to DASM.

i) YOUR.CODE

- print the invalid function code of the completed I/O operation.

j) DASM.ABORTED

- the DASM process is terminated. The satellite line cannot longer be used.
- the created PDB(s) and BDN(s) are returned to the free block queue.

b) STOP.LINE

. the X25HAND task is requested to perform the necessary operations to close all the opened virtual circuits. Via ITCALL, a block having the following format is enqueued to the X25HAND task:

```
+----------+
|   LINK   |
+----------+
|    0     |
+----------+
|    0     |
+----------+
|    0     |
+----------+
|    0     |
+----------+
|STOP.LINE |
+----------+
| INTERNET |
+----------+
```

. it is assumed that the closing procedure is successfully completed and a value > 0 is returned.

c) READ.LINE

. no operation because a new pending read operation is automatically issued by the X25HAND task on completion of a read operation.

d) SEND.DATA

. the X25HAND is requested to send the data on the specified Logical Unit (associated to a virtual circuit). Via ITCALL, a block having the following format is enqueued to the X25HAND task:

```
+--------------------+
|       LINK         |
+--------------------+
|   BDB ADDRESS      |
+--------------------+
| DATA OFFSET (word) |
+--------------------+
| DATALENGTH (byte)  |
+--------------------+
|  LOGICAL UNIT      |
+--------------------+
|   SEND.DATA        |
+--------------------+
|    INTERNET        |
+--------------------+
```

where:

BDB ADDRESS — is the address of the BDB describing the buffer which contains the data to be sent;

DATA OFFSET — is the word-offset of the beginning of the data inside the buffer;

DATALENGTH — is the length in bytes of the data to be sent;

LOGICAL UNIT — is the Logical Unit Number which must be used by X25HAND to send the data. Different Logical Units correspond to different destinations.

SEND.DATA — Internetworking code.

e) NEXTNET.LOCHDR

. no operation because the X25 network is always a "transport" network. It means that does not exist an X25-local header to be added on the top of the internet header.

f) LOC.HDR

. this case will never be entered because the X25 line is only a "transport" network.

g) SET.INTHDR

. the INT.HDR is always present when the routine is invoked at this entry point because data sent via the X25 network must always have already the internet header;
. USERBUF have only to be pointed on the top of the internet header, skipping the initial spare space of the buffer.
. the value IH.ALREADY.EXISTENT is returned.

h) BUF.GCT

. this case will never be entered because the X25 RX buffer pool is completely handled by the X25HAND task.

i) YOUR.CODE

. a check is made to verify if the function code of the enqueued block is equal to INI.FAILED or to STOP.FAILED. In both cases a message is printed on the operator console and the STOP.ROUTINE address is copied in NTRAVEC at the offset corresponding to the X25 line handler routine.
Otherwise the received invalid function code is printed on the terminal.

A very simple tecnique is used to map the addresses into the destination port word of the packet: the network and port number range is restricted (up to 15).
Therefore the address becomes a word with three compressed fields

```
    Bit  15      12 11              4 3       0
         +-----------------------------------+
         | Net    |       Station     | Port  |
         +-----------------------------------+
```

that are expanded into the larger fields (1 word each) of the internet address.

10.2 CAMBRIDGE RING LINE HANDLER BEHAVIOUR

In the following the actions performed by the cambridge ring line handler routine are described. The different behaviour is on the basis of the input "keyword" parameter.

a) SET.INTHDR

Incoming data have on top the SSP header and the internet header should be added. Two different cases may arise:

- packet coming from a local TM
  - the internet destination address is built expanding the destination word (got from the spare word of fig. 3.1)
  - the internet source address is built using local network and station numbers and the reply port number (word 1 of the packet)
- packet from another station on the ring
  - the internet destination address is built expanding the destination port word (known by the driver when the read was completed and passed using the spare word of fig. 3.2)
  - the internet source address is built using local network number, source station number (passed by the driver in the high byte of I/O CODE fig.3.2) and reply port number (word 1 of the packet)
  - the value IH.ADDED is returned.

b) LCC.HDR

- The internet header is stripped off and the reply port word of the packet is filled with the compressed internet source address.

c) NEXTNET.LCCHDR

This case is never entered because the ring is not considered a "transport" network.

d) INI.LINE

- The receive buffer pool is created and initialized for the ST driver.
- The VUDRV is initialized and a pending read operation is issued; it will be waiting for a packet from any station and for any destination port.

e) STOP.LINE

- The created buffer pool is released
- the pending read is killed.

f) READ.LINE

- A new pending read on the ring is issued.

g) BUF.GOT

This case is entered if there was no buffer available to perform the read operation; when a buffer becomes available the line-handler is awaken and therefore a new read is issued.

h) SEND.DATA

- An ITCALL is performed to VUDRV specifying the destination station and port numbers.

i) YOUR.CODE

This case should never be entered. Just in case:

- print the invalid function code of the completed I/O operation.

11. INTERNET: LK LINE HANDLER STRUCTURE (CERNET)

```
                                              +------+  +-------+
                                              | XFL  |  |FILEMA|
                                              +------+  +-------+
                                                 |         |  via NWDRV
                                                +V---------V+
           +------+     +------+                |  CERNET   |
           |T.M.  |     |T.M.  |                |   T.M.    |
           +------+     +------+                +-----------+
              |            |                        |
+-----------------------------------------------------------+
|                                        |     CERNET        |
|                                        | T.M. INTERFACE    |
|                                        |     ROUTINE       |
|                                        +-------------------|
|   "BRAIN" OF INTERNET                                      |
|                                         +------------------|
|                                         | LK LINE-HANDLER  |
|                                         |     ROUTINE      |
+-----------------------------------------------------------+
                                                A
                                                |  ITCALL
                                                V
                                         +------------------+
                                         |      LK DRV      |
                                         +------------------+
                                                |
                                                |
                                                +--+
                                                   |
                                                   |
                                                   V                 LDC
   ---------------------------------------------------------------------------
```

11.1 Preliminary remarks

CERNET is a packet local network with 15 switching nodes built up at CERN over the last five years, which allows nearly 100 small and large computers of different makes to intercommunicate at speed of up to several hundred Kb/s, with very low error rates. At CNUCE a small-size CERNET has been installed and used between Pisa and S.Piero a Grado. The packet switching network may occasionally lose a packet; the T.M. end-to-end protocol provide detection of and recovery.

Before two processes can start exchange messages, they must extabilish a "logical link" or "connection" between themselves. This link is used for message addressing, flow control and error control. Once it exists, the logical link is full-duplex, i.e. messages may flow simultaneously in both directions.

The "packet" is the unit of information handled by the packet switching network. It has the absolute maximum size of 1023 words (16 bits) and is composed of a fixed format header and a variable-sized data part. One or more packets carry one "message". A message is a bit string that a transmitter process wants to convey to a receiver process, using the network. There is no intrinsic upper bound on the length of a message.

The Internet process works on the Cernet data at the packet level.

The Cernet packet format is the following:

```
                +--------------------------+   <---
Word   0        |  Packet size (in words)  |      |
                +--------------------------+      |
  "    1        |     Destination T.M.     |      |
                +--------------------------+   CERNET LOCAL HEADER
       2        |        Source T.M.       |      |  (routing header)
                +--------------------------+      |
       3        |    Routing facilities    |      |
                +--------------------------+   <---
       4        |___                    ___|      |
       5        |___                    ___|      |
       6        |___                    ___|   PROTOCOL HEADER
       7        |___                    ___|      |
       8        |___                    ___|      |
       9        |                          |      |
                +--------------------------+   <---
      10        |                          |      |
                +--------------------------+      |
      11        |                          |      |
                +--------------------------+      |
      12        |      Message header      |   DATA AREA
                +--------------------------+      |
      13        |                          |      |
                |                          |      .
                       Message body               .
                |                          |      .
       n        |                          |      |
                +--------------------------+   <---
```

Each Cernet address is 16 bits long so structured:

```
+---------+---------+---------+---------+
|    R    |    S    |    I    |    T    |
+---------+---------+---------+---------+
```

where:

R = region number
S = subscriber
I = indirect subscriber
T = transport manager number

All the above fields are 4 bits long.

11.2 The Cernet address-mapping problem

The method used for mapping the LOCAL addresses into INTERNET addresses and viceversa is quite dependent on the installation site. The main difference is in dealing between already existing network installations and new ones.
In any case the three words in the Internet Header reserved for the source (/destination) address field have the following structure:

```
+---------------------------------------------------+
| NET NUMBER    | LOCAL  INTERNET  ADDRESS          |
+---------------------------------------------------+
 16 bits (1 w)          32 bits (2 w)
```

In any considered network, to each entity known by the Internet process a local internet address is assigned.

--<u>CERNET installation at CERN</u>

13 machines of the CERNET network are considered as belonging to the internet environment and the relative local internet addresses are assigned from 1 to 13. The local addresses are inserted in the LOCAL TABLE; their positions in the table are the relative LOCAL INTERNET ADDRESSES.

The positions are assigned giving the lower numbers to the more frequently used entities. Scanning the LOCAL TABLE, the source address is translated from "local cernet" to "local internet"; the destination local address is easily obtained from the same table considering the destination "local internet" address as offset in the table.

AS only 5 addresses were available to see the rest of the internetworking world from the Cernet network at Cern, it was needed to organize another address mapping table, the INTERNET TABLE, in a flat way. This table has two fields for each of the five entries:

```
+-------------------+-------------------+
| LOCAL ADDRESS     | FULL INTERN. ADDR.|
+---------------------------------------+
      16 bits             16 bits
```

The Local Address is a Cernet type address 16 bits long (1 word) and the Full Internet Address is compressed in 16 bits with the following meaning:

--high byte: represents the net number
--low byte : indicates the local internet address.

The INTERNET TABLE allows the translation from LOCAL into LOCAL INTERNET of the destination addresses and the translation of the source addresses from LOCAL INTERNET into LOCAL.

The search is made scanning the whole table. Translating the destination address from LOCAL into INTERNET, the search is performed on the INTERNET TABLE first and on the LOCAL TABLE successively. In this second case, the NET number is the CERNET NET number at Cern. On translating the source address from INTERNET INTO LOCAL, a check is made first if the NET number is relative to the CERNET at Cern. In this case the search is performed on the LOCAL TABLE, otherwise it is performed on the INTEBNET TABLE after compressing the Full Internet Address.

** LOCAL TABLE for CERNET at CERN **

OFFSET *	CERNET ADDR.	MACHINE NAME	DESCRIPTION	MACHINE TYPE
1	#X5000	IBM	ibm file manager	IBM 3081
2	#X320A	STELLA2	stella2 inter. node	PDP 11/40
3	#XB100	PDNA7A	framm data acquis.	PDP 11/34
4	#XB200	PDNA7B	framm data monitor	PDP 11/34
5	#X6900	PDINFN	infn/cernet/dec gatew	PDP 11/40
6	#X8000	PDNA31	fantechi	PDP 11/??
7	#XE200	PDR210	isr R210	PDP 11/60
8	#XB400	PDCPG4	CPG4 camac support	PDP 11/34
9	#X6300	NDSTELLA	stella NORD 10	NORD-10
10	#X3123	OMCPG1	program development	PDP 11/60
11	#X322E	OMCPG4	pdp online support	PDP 11/34
12	#X4000	CDC	cdc file manager	CYBER 170
13	#XCA00	VXALEPH	aleph	VAX 11/750

(*) this offset constitutes the LOCAL INTERNET ADDRESS.

** INTERNET TABLE for CERNET at CERN **

LOCAL ADDR. (exadecimal)	FULL INTERNET net number	ADDRESS • loc. int. address	MACHINE DESCRIPTION AND TYPE
3219	03 •	01	Cnuce internet node PDP11/70
320B	03 •	02	S. Piero-Infn PDP11/60
320C	09 •	01	Frascati-Infn PDP11/34
320D	07 •	01	ISPRA-Euratom PDP11/44
320A	06 •	02	Stella2 inter.node PDP11/40

--PISA-ISPRA-FRASCATI cases

In these cases the Cernet network installation are completely new, so it was possible to give a structure to the addresses already swited to the internetworking. The LOCAL ADDRESSES were obviously kept 16 bits long. All the addresses having the most significative 4 bits equal to a gateway address are used to map the rest of the internetworking world. Thus, indicating with letters the four bits field the address structure is so represented:

G N L L

where:

G	=	gateway number (4 bits)
N	=	network number (4 bits)
LL	=	local internetwork address (8 bits).

The sharing of the 12 bits between the last two fields could be easily rewewed in the proportions; nevertheless 4 bits were enough to include all the network numbers in the current implementation.
No tables are requested in these cases for the address mapping; addresses only must be expanded or compressed depending upon the type of translation.

11.3 LK LINE HANDLER BEHAVIOUR (CERNET HANDLER)

In the following the actions performed by the cernet line handler routine are described. The different behaviour is on the basis of the input "keyword" parameter.

a) SET.INTHDR

Incoming data have on top the cernet local header and the internet header should be added except the case in which the destination cernet address is a special one (#XFF) that means that the internet header is already present and follows the cernet local header. If the internet header must be added, it is built in such a way that the last 4 words overlap the cernet local header. The address mapping rules previously described are used to create the internet addresses.

b) LCC.HDR

The internet header is stripped off and the cernet local header is rebuilt using the mapping rules previously described.

c) NEXTNET.LCCHDR

This case is never used in the current implementation because CERNET was never used as "transport" network for data not belonging to the cernet environment. In any case, the cernet local header must be added on the top of the internet header using as source and destination addresses the current gateway address and next gateway address respectively associated in NETTAB to the "next net" to be crossed.

d) INI.LINE

- The receive buffer pool is created and initialized for the ST driver.
- The LKDRV is initialized and a pending read operation is issued.

e) STOP.LINE

- The created buffer pool is released
- the pending read is killed.

f) READ.LINE

- A new pending read operation on the LK driver is issued.

g) BUF.GOT

This case is entered if there was no buffer available to perform the read operation; when a buffer becomes available the line-handler is awaken and therefore a pending read operation is issued.

h) SEND.DATA

- An ITCALL is performed to LKDRV specifying in the optional QIO parameters the "unsave mode" parameter.

i) YOUR.CODE

This case should never be entered. Just in case:

- print the invalid function code of the completed I/O operation.

12. Transport Manager Interface Routine Behaviour

12.1 The Cernet T.M. interface routine is a subset of the LK line-handler routine.

12.2 The Tape T.M. interface routine has not been written in the current implementation because the tape-to-tape application is not yet implemented.

12.3 The SSP T.M. interface routine is a subset of the VU line-handler routine (Cambridge Ring).

12.4 The TALK T.M. interface routine has the following behaviour:

a) CASE SET.INTHDR:
as the internet header is already present in the data passed by the TALK process, the only actions to do are:

-USERBUF must be pointed to the internet header

-the version number must be written in the internet header.

b) CASE LOC.HDR:
-the two words of LOCIDVEC are filled with the value 1.

These are the only two cases in which this T.M. interface routine is invoked.

13. X25HAND TASK BEHAVIOUR

13.1 General Considerations

The EURONET packet switching network is used to interconnect the INET network (an X25 local network installed in ISPRA at the EURATOM) and the LDC at DUBLIN to the pisa LDC installed at CNUCE. The INET gateway and the PISA LDC are connected to the Euronet node in Rome by means of two 9.6 Kb/s leased lines respectively.

An X25 network may have:

- -subscribers with no internet software installed; they are not taken in consideration at all.
- -subscribers with the internet software installed; they are considered.

A Virtual Circuit (V.C.) for the internet traffic is extabilished between two subscribers having a special password (the source node name); between two subscribers one V.C. at maximum can be extabilished where all the traffic is multiplexed. A logical unit number identifies a virtual circuit. In the current implementation of the X25HAND task, logical unit numbers are assigned to the following virtual circuits:

V.C. between PISA and ISPRA and viceversa
V.C. between PISA and GENEVA and viceversa
(not yet connected)
V.C. between PISA and DUBLIN and viceversa.

In the X25 protocol, a Connect Request (C.R.) is issued in order to extabilish a V.C. connecting two DCEs (stations) for data exchanging. If the other DCE of the V.C. sends back a Connect Accept (C.A.), the virtual circuit is extabilished and data can be sent/received. Otherwise a disconnect is sent back by the X25 network itself.
A V.C. is extabilished when a packet has to be sent on the network. If the X25HAND task accepts a C.R. (sending back a C.A.), a virtual circuit is set up with the partner.
When the internet system is shutted-down, the X25HAND task is advised to close all the opened virtual circuits; the other ends of the V.C. will be notified by the network about the closing.

In conclusion:

-C.R.	it is performed before sending data on a V.C. that is still closed
-C.A.	it is issued when a C.R. is received
-DISCONNECT	it is performed when the system must be shutted down or automatically by the network on an opened virtual circuit.

13.2 Program Description

The X25HAND task has been written in BCPL and all the calls to the X25 directives are performed via ITCALL. The X25HAND task creates the ...XRG region in which the receiving pool is allocated. From this pool, buffers are got to receive data on a virtual circuit.
The X25HAND task is always in wait state from the INTERNET task and from the X25 network. All the possible virtual circuits are in "disconnected" state.

----<u>Activated by INTERNET:</u>

a) INI.LINE command:

The X25HAND task opens the network data queue.

b) STOP.LINE command:

All the opened virtual circuits are closed.

c) SEND.DATA command:

The X25HAND task is requested to send data on a specific virtual circuit. If the V.C. is already connected, data are immediately sent. If the V.C. is disconnected, the connect request is sent and the virtual circuit state is changed from "disconnected" to "wait-for-ack"; the data to be sent are enqueued for later sending. If the state of the virtual circuit is already "wait-for ack", the data to be sent are enqueued for later sending.

----<u>Activated by X25 network:</u>

a) IO.XCR function:

Receive operation end. Data are received by X25HAND task and, if the I/O operation was succesfully completed, the received data are passed to the INTERNET task via ITCALL. Otherwise the receiving buffer is released. In any case a new pending read operation is issued on the virtual circuit on which the RX operation was completed.

b) IO.SND function:

Transmit operation end. The data buffer related to the TX operation completed is released.

c) IC.XCN function:

Connect request operation completed; a virtual circuit has been opened. The state of the corresponding virtual circuit is set to "connected"; a pending read operation on that virtual circuit is issued. Moreover, all eventually enqueued packets on the just connected virtual circuit are sent.

d) IC.XAC function:

Connect accept operation completed. The state of the corresponding virtual circuit is set to "connected" . A pending receive data operation is issued on that virtual circuit.

e) IO.XRJ/IO.XDC function:

Reject/disconnect operation completed. No more operations issued.

f) IO.GND function:

Get network data queue operation completed. A command has been received. Type of commands can be:

--CONREQ

Connection request command received. If the sender name is unknown by the X25HAND task, a message is printed on the operator console and a Reject Connection is sent. If known, a Connection Accept is sent.

--DISCONNECTION/RESTART

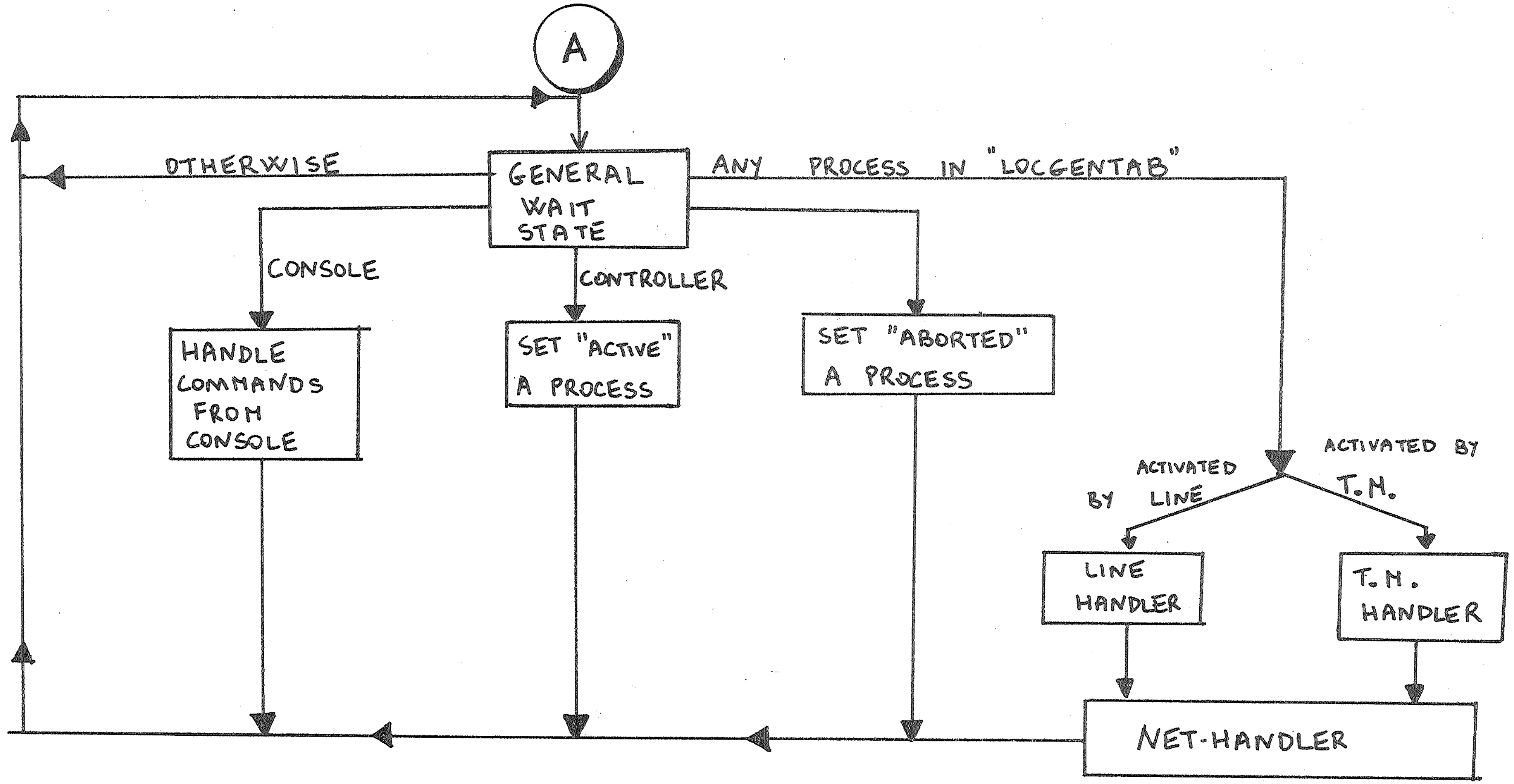
Disconnect/restart command received. The X25HAND task sets one virtual circuit/all virtual circuit(s) state(s) to "disconnect". The disconnect operation is sent on the virtual circuit/all virtual circuits.

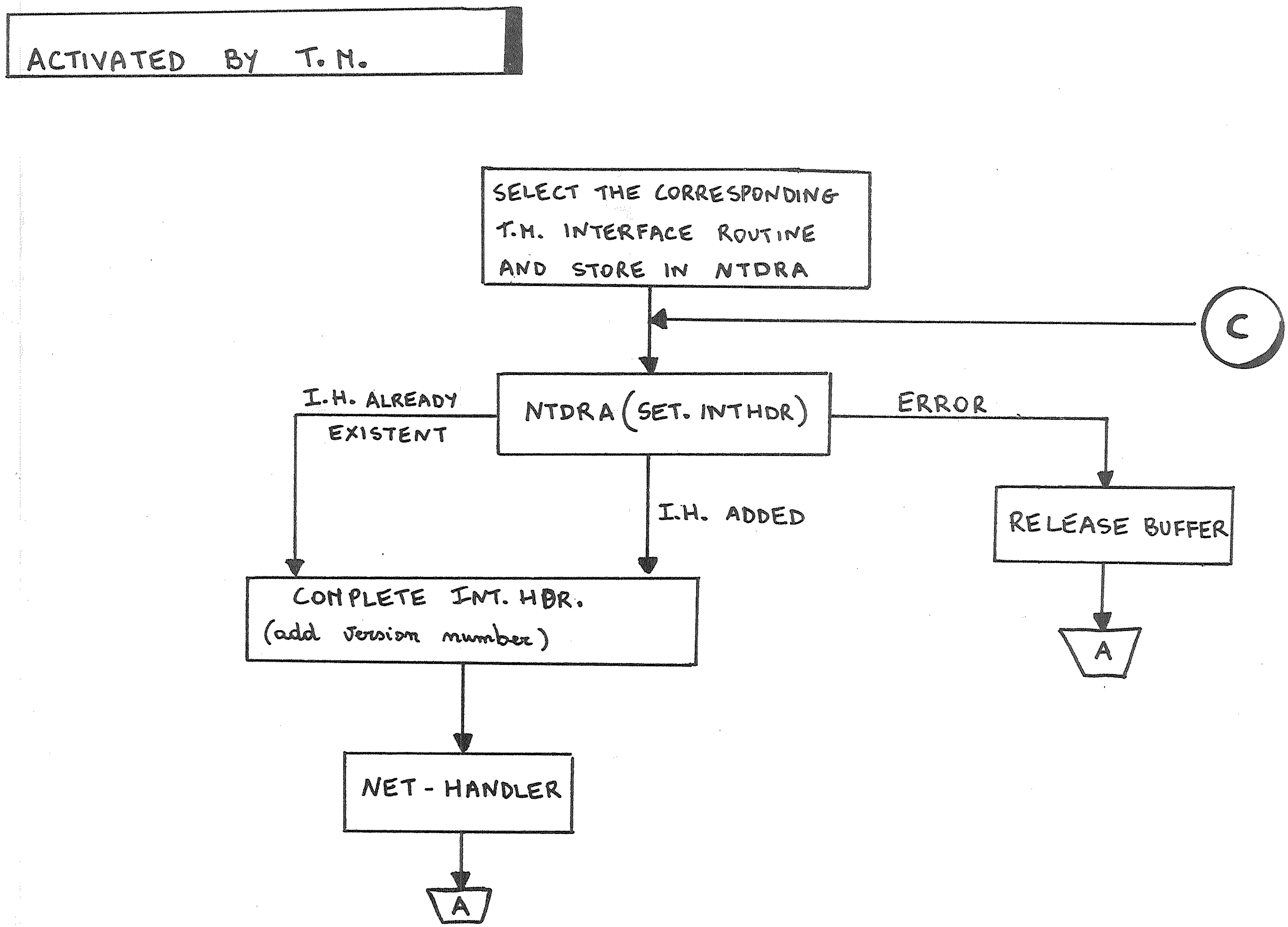
INTERNET

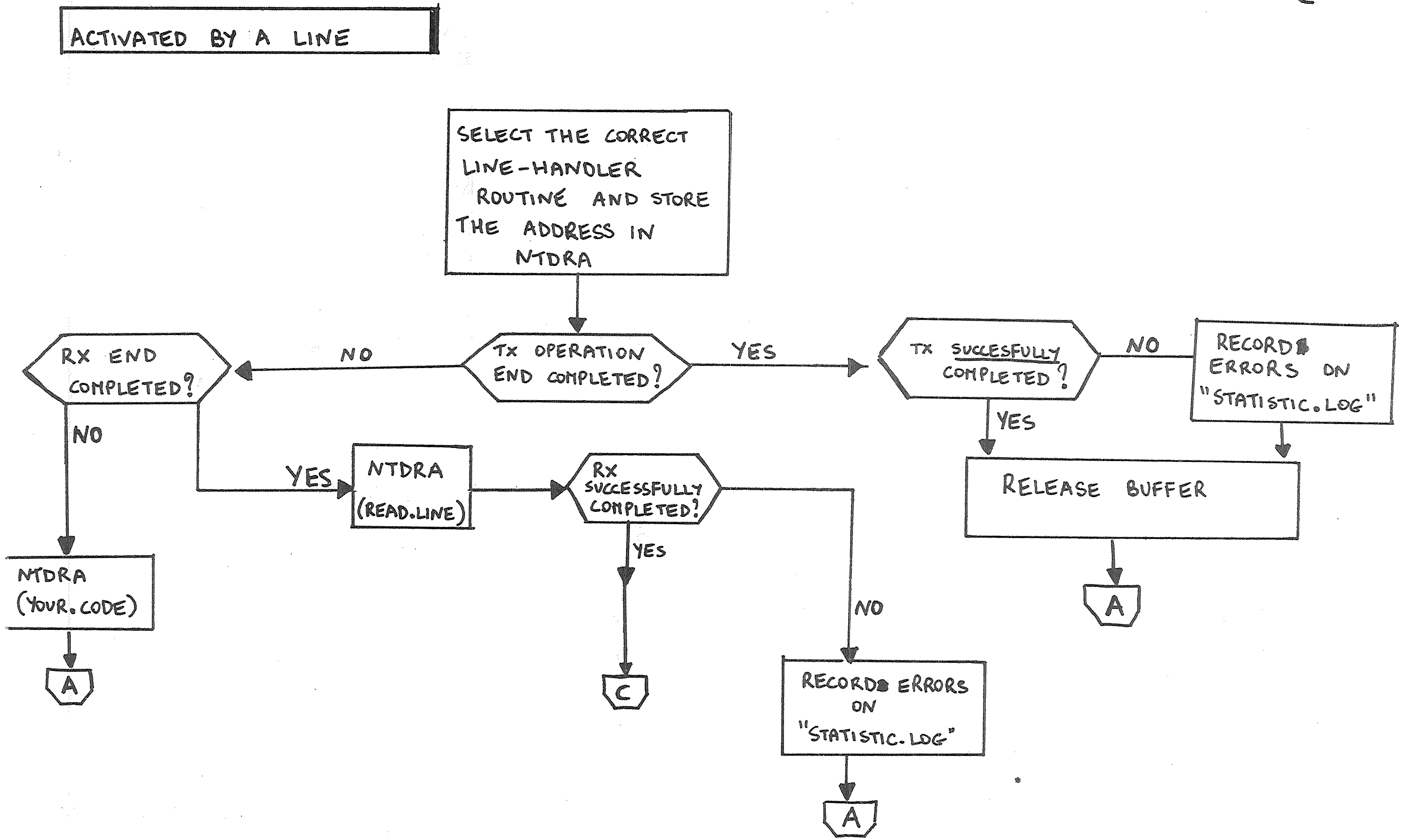
LOGICAL

FLOW - CHARTS

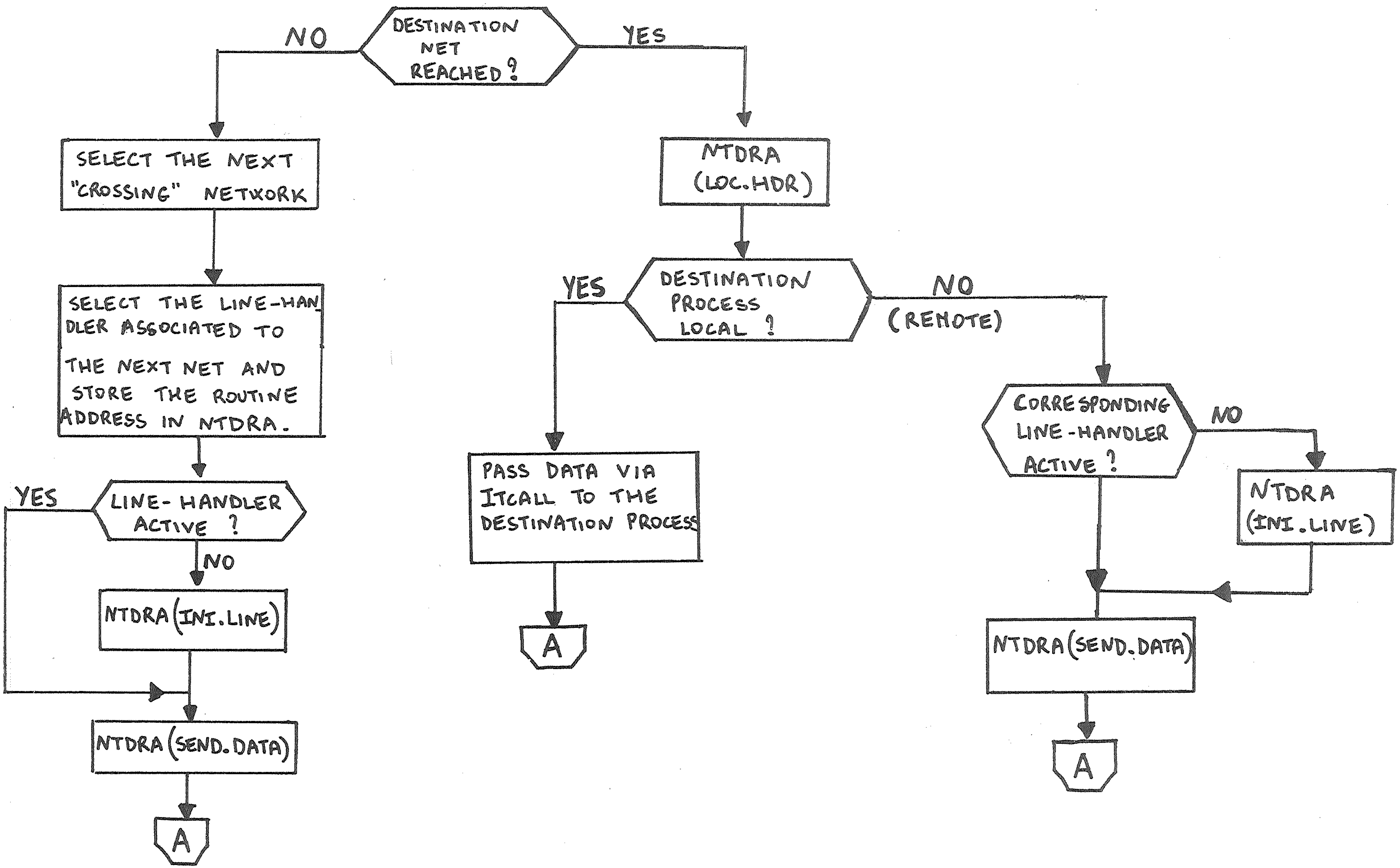
INTERNET MAIN PROGRAM




ACTIVATED BY T.M.
SELECT THE CORRESPONDING T.M. INTERFACE ROUTINE AND STORE IN NTDRA
C
NTDRA (SET. INTHDR)
I.H. ALREADY EXISTENT
I.H. ADDED
ERROR
RELEASE BUFFER
A
COMPLETE INT. HDR.
(add version number)
NET - HANDLER
A


ACTIVATED BY A LINE
SELECT THE CORRECT LINE-HANDLER ROUTINE AND STORE THE ADDRESS IN NTDRA
TX OPERATION END COMPLETED?
NO
YES
RX END COMPLETED?
NO
YES
NTDRA (YOUR.CODE)
A
NTDRA (READ.LINE)
RX SUCCESSFULLY COMPLETED?
YES
C
NO
RECORD ERRORS ON "STATISTIC.LOG"
A
TX SUCCESFULLY COMPLETED?
NO
RECORD ERRORS ON "STATISTIC.LOG"
YES
RELEASE BUFFER
A

NET-HANDLER


DESTINATION NET REACHED ?
NO
YES
SELECT THE NEXT "CROSSING" NETWORK
SELECT THE LINE-HANDLER ASSOCIATED TO THE NEXT NET AND STORE THE ROUTINE ADDRESS IN NTDRA.
LINE-HANDLER ACTIVE ?
YES
NO
NTDRA (INI.LINE)
NTDRA (SEND.DATA)
A
NTDRA (LOC.HDR)
DESTINATION PROCESS LOCAL ?
YES
NO (REMOTE)
PASS DATA VIA ITCALL TO THE DESTINATION PROCESS
A
CORRESPONDING LINE-HANDLER ACTIVE ?
NO
NTDRA (INI.LINE)
NTDRA (SEND.DATA)
A