ARPA network: Functional areas.
TABLE OF CONTENTS

I Transmission features
   1. Transmission checking
   2. HOST(A) to HOST(B) links

II Functional software specifications
   1. User program - Del language
   2. Network program
   3. Transmission Handler

III Link establishment procedure
   1. General procedure
   2. Example
I. Transmission Features

I-1. Transmission checking

There exist two kind of transmission checking:

* IMP to IMP

It is a cyclic checksum computed and checked by the BBN hardware.

* HOST to HOST

It is a special 16 bit checksum computed and checked by the HOST programs.

For this purpose the HOST message is broken down into 152 bit pieces A, B, C, ... (1152 = 2.24 # packets). For each of these we calculate an end-around 16 bit sum and form the checksum as follows:

\[
\text{Checksum} = \text{Sum of A} + 2 \times \text{Sum of B} + 4 \times \text{Sum of C} + \ldots
\]

This is 16 bit checksum is located just after the masking of the HOST header, that is after the sending a message start (See fig 1)

This checking procedure allows the verification of the right IMP to IMP procedure. It also protects against HOST to IMP (or IMP to HOST) bad transmission, and against IMP packet number inversion.
Remark: Example of an end-around carry sum:

\[ \begin{array}{c}
101 \\
+ 101 \\
\hline
1010 \\
\end{array} \]

Checksum = 011

I-2. HOST(A) to HOST(B) Links.

32 links are possible between two HOSTs. Each of these links are viewed as full duplex.

Link 0 is considered as a control link (regardless of connection status of any kind:

The others are used either for "TTY type" line connections or for full transmission connections.

A "TTY Like" connection is one where:

- ASCII characters are sent or received.
- Echos are generated by the remote HOST.
- The remote HOST looks for specific sequence (break or interrupt control characters).
- The transmission is slow.
II Functional software specifications

- See fig 2 -

II-1 User program - DEL language

It's an application program that exists within a host. For example, the NLS program at SRI.
For network purposes this program should be viewed as parted in two: The local part and the hard part (the body).

- The hard part represents the user application.
- The local control part is the user interface:
It exerts immediate control of the terminal and provides specific responses to the man's inputs.

In order to facilitate and speed up remote interprocess communication the 'local control' program can be transmitted to another host. Thanks to that capability an UCLA user, for example, will use its terminal exactly like the SRI user uses its own. Also only the program data can transmit over the link (versus the user terminal dialogue). See:

DELF language (Decide Encode Language)

The 'local control' program should be written in a DEL language - when it is transmitted over to a remote host.
II-2 Network program

This program should provide:
1. The outgoing message multiplexing (and incoming message
distribution).
2. The link initiation procedure: see below.
3. The HOST message heading.
4. The "HOST-HOST" checksum computation/checking.
5. The receiving of the RFNM control message.
6. The supervisory control of the Handler program.

II-3 Transmission Handler program

This program is initiated either by the network program, or by the I/O interrupt. Its function is to
control the channel hardware unit.

The program is very short and closely related to
the Network program.

Remark: As the communication is full duplex, the Host
and Handler programs can be viewed as divided into
2 parts: one is concerned with the outgoing messages, the
other with the incoming messages.
III  Link establishment procedure

III-1 General procedure

* Establish link to Host (x)
  A "TTY like" connection is established to Host(x). This connection is in a pre-log-in state.
  Standard TTY codes are expected. The remote Host provides the echo.

* Send/Receive characters over "TTY like" link.

* Establish file transmission link parallel to existing "TTY like" link. This must be executed by both Host and user programs.

* Send/Receive over "file like" link.

III-2 Example

Suppose that we, at UCLA, want to use NLS at SRI

A) Local arrangements
   * Log in on local TTY to Sigma 7. We
   are now talking to the command level of the Sigma
   operating system.
   * Select an user program to put in execute.
We start up a program we previously wrote on a TTY and the transmission with SRI.

To select the standard U.C.L.A. communication program. This is the standard option for single control of a remote HOST.

b) Connection to SRI

* Initiate link to remote HOST

The previously selected program asks the U.C.L.A. network program to initiate a link to SRI. The Network program:

- Selects an open link e.g. 25
- Sends a message to SRI over link 25.
- Waits for an acceptance from the SRI network program. This acceptance is in the form of another message over link 25.
- If it should happen that both SRI and UCLA try to initiate a connection over 25, the one with the higher priority will prevail. (This is extremely rare). We suggest that the priority be exactly the HOST identification number.
- This connection is teletype-like connect only a standard subset of ASCII characters expected or accepted.
- The connection is a "pre-log-in" connect. The remote HOST expects its standard log-in sequence.
Log-in at SRI.

This may be done either by the UCLA program if it knows how, or by the man at UCLA by typing the required sequence. We are now talking the command level of the SRI operating system.

1) Request 'local control' program from SRI.

The UCLA selected program sends control over the link to the SRI user program. The user requests that SRI transmit to UCLA the 'local control' program which is written in the DEL language.

* We compile this program through a compiler.

* We turn control of the TTY link a terminal over the just compiled DEL program.
(Fig 1) : UCLA Host memo