package Transport_Route

The Rational system maintains a table used for routing IP packets, including TCP/IP and UDP/IP packets. When sending a packet to a machine on some other network, the packet must be routed through a gateway and not directly to the destination machine. Some gateways do not respond to ARP (Address Resolution Protocol) queries for destination machines whose traffic they carry; therefore, the sending machine must know the Internet address of the gateway in order to transmit packets to it.

The routing table contains a list of entries, each containing a route (the Internet address of a gateway) with a destination that can be reached by the route. The destination may be a specific Host_Id (Internet address), a network number (signifying all hosts in that network), or the Null_Host_Id (signifying any remote host). There can be multiple entries for each route, identifying multiple hosts or networks accessible by way of the specified route. The table is kept ordered starting with all host-specific entries, followed by all network-specific entries, followed by the default entry. Within each group, entries are maintained in the order in which they were defined. When deciding where to send an outgoing packet, the table entries are searched in order.

```
with Transport_Defs;

package Transport_Route is

     procedure Show      (Route       : String := "";
                          Destination : String := "";
                          Network     : Transport_Defs.Network_Name  := "";
                          Response    : String := "<PROFILE>");

     procedure Load      (Table       : String := "!Machine.Transport_Routes";
                          Form        : String := "";
                          Response    : String := "<PROFILE>");

     procedure Define    (Route       : String;
                          Destination : String := "";
                          Network     : Transport_Defs.Network_Name  := "IP";
                          Response    : String := "<PROFILE>");

     procedure Undefine  (Route       : String;
                          Destination : String := "";
                          Network     : Transport_Defs.Network_Name  := "IP";
                          Response    : String := "<PROFILE>");

end Transport_Route;
```

procedure Define

```
procedure Define (Route       : String;
                  Destination : String := "";
                  Network     : Transport_Defs.Network_Name  := "IP";
                  Response    : String := "<PROFILE>");
```

Description

Adds one entry to the routing table, with the given values. If there is already such an entry in the table, Define does nothing.

Parameters

```
Route : String;
```

When Network = "IP", Route is the Internet address of an IP gateway.

Route may be a name, or a Transport_Defs.Host_Id in decimal dotted notation, for example "89.64.1.22".

If Route is a name, it is resolved to a Host_Id by Transport_Name.Host_To_Host_Id.

If Route = "", it resolves to Transport_Defs.Null_Host_Id.

```
Destination :  String := "";
```

When Network = "IP", Destination can be either:

(a) the complete Internet address of a machine, or

(b) the network number of a remote network, or

(c) the Null_Host_Id.

A complete Internet address is used to route packets to that specific destination machine. A network number is used to route packets to any machine in that network. The Null_Host_Id is used to indicate a default route for machines that do not match any other table entry.

Destination may be a name, or a Transport_Defs.Host_Id in decimal dotted notation, for example "128.33".

If Destination is a name, it is resolved to a Host_Id by Transport_Name.Host_To_Host_Id.

If Destination = "", it resolves to Transport_Defs.Null_Host_Id.

```
Network :  Transport_Defs.Network_Name  := "IP";
```

The name of a transport service.

The default "IP" represents transport services based on the Internet Protocol and Ethernet (for example, TCP/IP and UDP/IP).

```
Response :  String := "<PROFILE>";
```

The options controlling logging and error handling. See package Profile.

Example

The following example adds an entry indicating that packets destined for the machine named "logo" should be routed to the IP gateway at Internet address 89.64.1.22.

```
Transport_Route.Define ("89.64.1.22", "logo");
```

The following example adds an entry indicating that packets destined for network number 128.33 should be routed to the IP gateway named "shemp".

```
Transport_Route.Define ("shemp", "128.33");
```

The following example adds an entry indicating that, by default, packets destined for other networks should be routed to the IP gateway named "fred".

```
Transport_Route.Define ("fred");
```

---

procedure Load

```
procedure Load (Table    : String := "!Machine.Transport_Routes";
                Form     : String := "";
                Response : String := "<PROFILE>");
```

Description

Loads routing information from the named Table into the system's routing table.

Reads the object named by Table, using package Text_Io, passing the specified Form to Text_Io.Open. For each line in the Table, Load calls the Define procedure with parameter values parsed from the line.

Parameters

```
Table :  String := "!Machine.Transport_Routes";
```

The name of a file or other object that contains routing information. Within this object, each text line must contain the Host_Id or name of a route, optionally followed by the Host_Id or name of a Destination, optionally followed by a Network_Name. These values must be separated by spaces. If the Network Name is omitted, "IP" is assumed. If the destination Host_Id is omitted, the Null_Host_Id is assumed.

```
Form :  String := "";
```

The options controlling how the Table is read. This string is passed to Text_Io-_.Open.Form.

```
Response :  String := "<PROFILE>";
```

The options controlling logging and error handling. See package Profile.

---

procedure Show

```
procedure Show      (Route       : String := "";
                     Destination : String := "";
                     Network     : Transport_Defs.Network_Name  := "";
                     Response    : String := "<PROFILE>");
```

Description

Creates a text listing of routing table entries, and writes it to the current output file. Only entries that match the given Route, Destination, and Network values are listed. In each case, the value "" is a wildcard that matches all entries.

Parameters

```
Route :  String;
```

When Network = "IP", Route is the Internet address of an IP gateway.

Route may be a name, or a Transport_Defs.Host_Id in decimal dotted notation, for example "89.64.1.22".

If Route is a name, it is resolved to a Host_Id by Transport_Name.Host_To_Host_Id.

If Route = "", entries for all routes are shown.

```
Destination :  String := "";
```

When Network = "IP", Destination can be either:

(a) the complete Internet address of a machine, or

(b) the network number of a remote network, or

(c) the Null_Host_Id.

A complete Internet address is used to route packets to that specific destination machine. A network number is used to route packets to any machine in that network. The Null_Host_Id is used to indicate a default route for machines that do not match any other table entry.

Destination may be a name, or a Transport_Defs.Host_Id in decimal dotted notation, for example "128.33".

If Destination is a name, it is resolved to a Host_Id by Transport_Name.Host_To_Host_Id.

If Destination = "", entries for all destinations are shown.

```
Network : Transport_Defs.Network_Name := "";
```

The name of a transport service.

The default "" causes entries for all networks to be shown. "IP" represents transport services based on the Internet Protocol and Ethernet (for example, TCP/IP and UDP/IP).

```
Response : String := "<PROFILE>";
```

The options controlling logging and error handling. See package Profile.

---

procedure Undefine

```
procedure Undefine (Route       : String;
                    Destination : String := "";
                    Network     : Transport_Defs.Network_Name  := "IP";
                    Response    : String := "<PROFILE>");
```

Description

Deletes the entry with the given values from the routing table. If there is no such entry in the table, Undefine does nothing.

Parameters

```
Route :  String;
```

When Network = "IP", Route is the Internet address of an IP gateway.

Route may be a name, or a Transport_Defs.Host_Id in decimal dotted notation, for example "89.64.1.22".

If Route is a name, it is resolved to a Host_Id by Transport_Name.Host_To_Host_Id.

If Route = "", it resolves to Transport_Defs.Null_Host_Id.

```
Destination :  String := "";
```

When Network = "IP", Destination can be either:

(a) the complete Internet address of a machine, or

(b) the network number of a remote network, or

(c) the Null_Host_Id.

A complete Internet address is used to route packets to that specific destination machine. A network number is used to route packets to any machine in that network. The Null_Host_Id is used to indicate a default route for machines that do not match any other table entry.

Destination may be a name, or a Transport_Defs.Host_Id in decimal dotted notation, for example "128.33".

If Destination is a name, it is resolved to a Host_Id by Transport_Name.Host_To_Host_Id.

If Destination = "", it resolves to Transport_Defs.Null_Host_Id.

```
Network :  Transport_Defs.Network_Name  := "IP";
```

The name of a transport service.

The default "IP" represents transport services based on the Internet Protocol and Ethernet (for example, TCP/IP and UDP/IP).

```
Response :  String := "<PROFILE>";
```

The options controlling logging and error handling. See package Profile.

---

end Transport_Route;

EXOS 8043

TCP/IP Protocol Package
For VAX/VMS Systems

Reference Manual

Publication No. 4200012-00
Revision A July 31, 1985

Excelan, Inc.
2180 Fortune Drive
San Jose, CA 95131

TABLE OF CONTENTS

Appendix F
TROUBLESHOOTING

F.1. INTRODUCTION

Most problems reported by end users can be resolved by reference to Appendix E, where the error messages displayed by the utilities are explained. More severe problems are generally reported to the system console and must resolved by the system administrator. This appendix provides guidance for the system administrator in resolving common faults in the operation of networking software.

The system administrator has access to more diagnostic facilities than other users do. These facilities include the following:

- System console. The EXOS 8043 logs general error messages on the system console. The conditions reported on the console include suspicious situations detected by the front-end processor. Refer to the section on replies in the VAX/VMS *Command Language User's Guide* for information on how to redirect and/or disable the logging of error messages.
- The *bstat* utility. This utility, described in Section 3.6.2 and in Appendix A, reports statistics on low-level communication activity. This information contains clues about problem conditions.
- *route* and *arp* utilities. These utilities allow the system administrator to inspect and alter tables in the front-end processor that affect the transmission of packets over the network. *route* is described in Section 3.6.4 and *arp* in Section 3.6.1; both are detailed in Appendix A.

F.2. INSTALLATION PROBLEMS

If problems arise during the installation and startup of EXOS 8043, the most likely point is when "netstart" is invoked. Symptoms can vary. If the problem can be diagnosed by VMS, a message appears on the console; an example is an "initialization failed" message. These error messages are explained in the VAX/VMS *System Messages and Recovery Procedures Manual*. More serious situations can have consequences as severe as crashing the system.

Typical causes to investigate include the following:

- The front-end board was not properly installed. Read Appendix B carefully, especially the notes which indicate common trouble spots.
- The values typed during installation for UNIBUS adapter number, CSR address or interrupt address did not match the hardware configuration. The consequences could be that the host attempted to manipulate the wrong device or a nonexistent device.

To ascertain the front-end processor's status during initialization, examine the LEDs mounted on the board. They flash status codes, which are explained in the *EXOS 204 Ethernet Front-End Processor Reference Manual*.

F.3. USER MESSAGES

Most user messages can be resolved without extensive investigation. However, three messages require some explanation.

No route to ...

This message from the EXOS board indicates that no route to the remote host can be found. This can happen if two hosts on the *same* network were assigned Internet addresses containing *different* network numbers. This can also happen if the remote host is on a different network but you have not added the proper route entry using the *route* command. Check the routing tables and Hosts file to resolve the incompatibility.

If the remote system is from Sun Microsystems, be aware that its default network number is 193, while Excelan's is 89. Change one or both to make them compatible.

Connection refused

This message typically indicates that even though a path was established to the remote host, the remote host is not prepared for the type of connection attempted. Therefore, the connection was refused. The most likely cause is that the server (daemon) has not been enabled on the remote host. Servers are normally enabled automatically when a system is booted. Check with the system administrator of the remote host.

Connection timed out

This is a more serious problem than a refused connection. It means that low-level packet transmission could not be accomplished in one or both directions. Either the connection request or its reply failed to get through. To isolate the problem, first determine if it occurs for all communicating partners or for just a specific one. Bearing this in mind, examine the history of console messages from EXOS. The next section details the console messages.

F.4. CONSOLE MESSAGES

Two types of information appear on the system console. When the EXOS front-end board is initialized, it reports a variety of configuration and version information, including the software/firmware/hardware versions and the Internet and Ethernet addresses that the networking software thinks are its own. If problems are suspected, addresses should be checked for consistency with the Hosts file and with the other hosts on the network.

The second category of console messages consists of warnings that occur during system operation. The following messages can appear; they provide clues to various trouble situations.

EXOS CODE 0001

This occurs when the TCP checksum calculated by the EXOS board for an incoming packet does not agree with the checksum in the packet header. Most likely, checksums were disabled or incorrectly calculated on the system that sent the packet. Otherwise, the packet may have suffered a transmission error (very unlikely on Ethernet) or the EXOS TCP checksum calculation was in error. Check the intercommunicability of various hosts with each other to determine which is malfunctioning.

EXOS CODE 0003 rxmt time = nnn

This indicates that the front-end was forced to retransmit packets due to a lack of response. It is normal for this to happen with *nnn* = 2 the first time you connect with a host. Trouble is indicated if it happens repeatedly with increasing values of *nnn*. The most likely causes are the following:

- The host with which communication is being attempted is down or is unable to communicate. Check if that host can communicate with another system.
- The IP address of the remote host is not properly entered in the Hosts file. You should ensure that both hosts agree on what each others' Internet addresses are. Check the following places for consistency: the Hosts files of the two machines and the addresses that the networking software believes to be their own. In the case of EXOS software systems, the address that the networking software believes to be its own is shown on the operator's console when the network module is started on the front-end board.
- There is an ARP mismatch. All hosts on a network must agree on whether ARP is in use. EXOS supports both alternatives, but other systems may not. Consult Section 3.6.1 and Appendix A for a description of ARP usage. Check the documentation of the other host's networking software to determine if the host supports ARP and, if so, how to enable/disable it. If the remote host does not understand ARP, you can use the ARP command to set up the address of the remote host in the ARP cache manually. Alternatively, you can operate the entire network without ARP; see Section 3.6.3.4 for information on mixing Ethernet boards from different vendors on the same network.
- If the remote computer is running the UNIX 4.2BSD operating system, beware of the following problem. As with EXOS, some 4.2BSD systems let you use "old style" mapping, in which Internet addresses are converted to Ethernet addresses by taking the low-order three bytes of the Internet address and appending them to the high-order three bytes of the Ethernet address. Unlike EXOS, UNIX 4.2BSD makes the decision based on the local host number (which is extracted from the Internet address). If this host number is greater than or equal to *oldmap*, old style mapping is used; otherwise ARP is used. *oldmap* is a variable inside the 4.2BSD operating system. Current 4.2BSD systems set *oldmap* to 1024. To circumvent this problem, you can either force your Internet address to meet the above constraint (that is, choose your local host number to be less than 1024) or modify the *oldmap* variable in the UNIX 4.2BSD kernel to be the largest positive number on the 4.2BSD system. The *adb* utility can be used to change *oldmap*.
- A bad routing table entry may be causing packets for the intended host to be launched to the wrong host or to a gateway that is either nonexistent or out of service. Use the *route* utility to examine the current state of the routing table. Check gateways for proper operation. Be aware that routing table updates can originate not only from the operator (using the *route* utility) but also as a result of "redirect" packets sent from a gateway.

- Be aware that for two host to communicate, *both* their routing tables need to contain paths to the other.

EXOS CODE 0102 xmit err 10

This message appears when the EXOS board is having trouble transmitting. This typically happens because the transceiver is not properly connected to the Ethernet cable. This causes excessive collisions. Ensure that the transceiver is properly connected.

Another possible cause is a general problem with the network, such as a short circuit.

EXOS CODE 0102 xmit err 04

This message indicates DMA underrun for the Ethernet chip. It can be reported by NX revisions greater than 4.7. If this problem persists, report it to Excelan.

EXOS CODE 0102 xmit err 20

This message indicates a problem in the attachment of the local node to the Ethernet cable. The symptom is lack of carrier sense during transmission. Review the material in Section B.3 that describes how to connect to the network.

EXOS CODE 0102 xmit err CB

This message results from one of two causes. Either the attachment to the network is faulty (as in the preceding paragraph) or there is a hardware fault on the front-end processor. The symptom is a timeout during transmission.

EXOS CODE 0105 recv err04

This indicates a DMA overrun within the front-end configuration. If it happens persistently and interferes with system operation, report the problem to Excelan.

EXOS CODE 0105 recv err 10

A packet longer than the specificaton permits was received.

EXOS CODE 0106
EXOS CODE 0107

An excessively long "trailer" packet was received. UNIX 4.2BSD networking produces trailer packets. These codes indicate that the packets are being produced incorrectly. Pursue the problem with the system(s) that might be transmitting the packets.

EXOS CODE 0115

No route exists to a correspondent. See Section F.3.

EXOS CODE 0207
EXOS CODE 0208
EXOS CODE 0616

These messages indicate a shortage of data buffers on the front-end processor. They may occur if many connections are actively transferring data concurrently, especially if the host stops reading data for numerous connections, forcing data to accumulate on the front end. If this situation persists in normal operation, add memory or reduce the number of active connections. In any event, arrange for host applications to read input data from the board most of the time.

EXOS CODE 0301 *parameters*

A host socket request contained illegal parameters such that it failed to specify a protocol that EXOS supports. *parameters* are displayed in hexadecimal in the following order:

- Type of protocol within a protocol family
- Protocol family
- Protocol within family
- Socket options
- Internet protocol family
- Internet TCP/UDP port
- Internet socket address

To find the source of the problem, investigate application programs attempting to open sockets.

EXOS CODE 0408 nnn

The EXOS software does not recognize a request code received from the host. There is probably a bug in the host's driver software. The number *nnn* is the improper request code. To find the source of the problem, investigate application programs trying new things. If you are using only Excelan-provided software, report this error to Excelan as a driver or utility problem.

EXOS CODE 0705
EXOS CODE 0706

These are checksum calculation mismatches similar to EXOS CODE 0001. Code 0705 pertains to the ICMP protocol, code 0706 to IP.

EXOS CODE xxxx

If codes that are not listed above are displayed and the EXOS is not working correctly, report the problem to Excelan.

CIRCUIT WAS RESET

This is actually a report from the host-resident device driver. It indicates that a connection was closed because an administrator caused the front-end processor to be reinitialized.

F.5. ABSENCE OF CONSOLE MESSAGES

If there are no messages on the system console to indicate the problem, investigate by looking at the traffic statistics of both noncommunicating partners. On any current EXOS system, use the *bstat* command (see Section 3.6.2 and Appendix A). If the remote system is running UNIX 4.2BSD, use its *netstat -i* command. Bracket the communication attempt with calls to these commands. For each host, you will then be able to tell if it is successfully sending and/or receiving. You will also be able to note if there are excessive numbers of abnormal conditions occurring, such as alignment errors, parity errors, and excessive collisions. All these situations occur occasionally; if one dominates the statistics, it points to the problem.

F.5.1. Alignment Errors

This error condition occurs because not all Ethernet controller boards are compatible with Version 2.0 of the Ethernet specification. In particular, the specification states that "alignment errors" are acceptable as long as the (Ethernet) checksum is correct. (In the previous version of the Ethernet specification, indication of an alignment error was sufficient grounds to discard

the packet.) This can be a problem since certain (transmitting) boards that use Seeq chips can generate an alignment error on the receiving side. If the specification is not totally met, this can cause packets to be discarded.

If the remote host is using an Interlan board (in a link-level mode), the board may be incompatible with early Excelan boards. Some older EXOS versions can cause alignment errors to be reported by the Interlan board. As a result, packets sent by the Excelan board are discarded by the Interlan board, but packets sent by the Interlan board are accepted by the Excelan board.

If this problem persists after investigating the possibility of a hardware failure or installation error, upgrade the Ethernet board(s) to a newer revision.

F.5.2. Excessive CRC Errors

This is most likely a hardware malfunction in one of the systems. The checksum on packets being transmitted is failing the test on receipt. Try communicating among a variety of hosts to see if one is unable to communicate with any others. This host is the likely culprit.

F.5.3. Excessive Collisions

Most likely, one station on the network is suffering a hardware malfunction causing it to continually assert carrier, in violation of Ethernet specifications. If this is really the problem, you will notice that no systems will be able to communicate successfully. Try selectively disconnecting hosts from the network to isolate the culprit.

F.5.4. SQE Test Failures

If you are using an Ethernet Version 1.0 transceiver, you have probably jumpered the EXOS board for responding to SQE tests; this can only be done by Ethernet Version 2.0. If you are using an Ethernet Version 2.0, an SQE test failure indicates that there is no heartbeat. Other possible causes are a cabling fault, a transceiver fault, or an EXOS hardware problem.

F.5.5. UDP and UNIX 4.2BSD

If you are using a UDP datagram protocol (not FTP or TELNET – these are TCP-based) and the remote machine is running UNIX 4.2BSD, there is a possible checksum problem. In early versions of 4.2, there was a bug in the UDP checksum calculation algorithm. Obtain a new version of software from your 4.2BSD supplier.

F.5.6. DMA Underrun

If this occurs frequently, report the problem to Excelan.

F.5.7. No Receive Buffers

This means there is insufficient data buffer space on the board for the level of activity. Add memory or reduce the number of active connections.

F.5.8. Duplicate IP Address

If you are communicating with a 4.2BSD system, this message occurs on the 4.2BSD system when the Hosts file from one host is copied to another system and the *localhost* entry in the file is not modified to reflect the new host. As a result, *netload* uses the same IP address (indexed by *localhost*) as the system from which the file was copied.

If you copy the Hosts file from another system, be sure to update it so that the local IP address has *localhost* on the proper line.

Network Working Group
Request for Comments: 960

J. Reynolds
J. Postel
ISI

December 1985

Obsoletes RFCs: 943, 923, 900, 870,
820, 790, 776, 770, 762, 758,
755, 750, 739, 604, 503, 433, 349
Obsoletes IENs: 127, 117, 93

ASSIGNED NUMBERS

Status of this Memo

This memo is an official status report on the numbers used in protocols in the ARPA-Internet community. Distribution of this memo is unlimited.

Introduction

This Network Working Group Request for Comments documents the currently assigned values from several series of numbers used in network protocol implementations. This RFC will be updated periodically, and in any case current information can be obtained from Joyce Reynolds. The assignment of numbers is also handled by Joyce. If you are developing a protocol or application that will require the use of a link, socket, port, protocol, network number, etc., please contact Joyce to receive a number assignment.

Joyce Reynolds
USC - Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90292-6695

Phone: (213) 822-1511

ARPA mail: JKREYNOLDS@USC-ISIB.ARPA

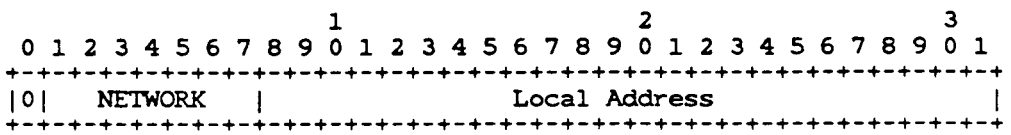
Most of the protocols mentioned here are documented in the RFC series of notes. The more prominent and more generally used are documented in the "Internet Protocol Transition Workbook" [39] or in the old "ARPANET Protocol Handbook" [40] prepared by the NIC. Some of the items listed are undocumented. Further information on protocols can be found in the memo "Official ARPA-Internet Protocols" [104].

In all cases the name and mailbox of the responsible individual is indicated. In the lists that follow, a bracketed entry, e.g., [nn,iii], at the right hand margin of the page indicates a reference for the listed protocol, where the number ("nn") cites the document and the letters ("iii") cites the person. Whenever possible, the letters are a NIC Ident as used in the WHOIS service.

ASSIGNED NETWORK NUMBERS

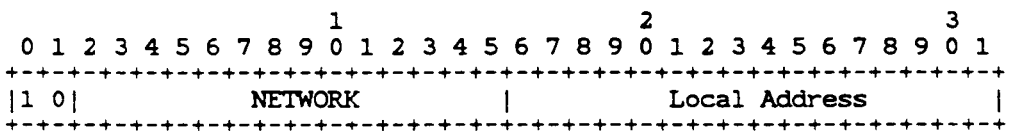
The network numbers listed here are used as internet addresses by the Internet Protocol (IP) [39,92]. The IP uses a 32-bit address field and divides that address into a network part and a "rest" or local address part. The division takes 3 forms or classes.

The first type of address, or class A, has a 7-bit network number and a 24-bit local address. The highest-order bit is set to 0. This allows 128 class A networks.



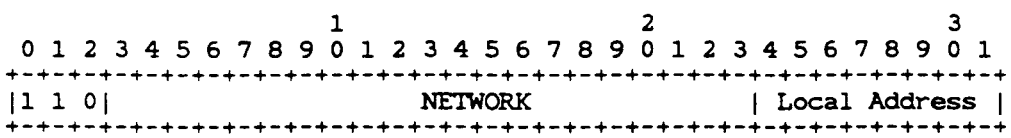
Class A Address

The second type of address, class B, has a 14-bit network number and a 16-bit local address. The two highest-order bits are set to 1-0. This allows 16,384 class B networks.



Class B Address

The third type of address, class C, has a 21-bit network number and a 8-bit local address. The three highest-order bits are set to 1-1-0. This allows 2,097,152 class C networks.



Class C Address

Note: No addresses are allowed with the three highest-order bits set to 1-1-1. These addresses (sometimes called "class D") are reserved.

One commonly used notation for internet host addresses divides the 32-bit address into four 8-bit fields and specifies the value of each field as a decimal number with the fields separated by periods. This is called the "dotted decimal" notation. For example, the internet address of USC-ISIB.ARPA in dotted decimal is 010.003.000.052, or 10.3.0.52.

The dotted decimal notation will be used in the listing of assigned network numbers. The class A networks will have nnn.rrr.rrr.rrr, the class B networks will have nnn.nnn.rrr.rrr, and the class C networks will have nnn.nnn.nnn.rrr, where nnn represents part or all of a network number and rrr represents part or all of a local address.

There are four catagories of users of Internet Addresses: Research, Defense, Government (Non-Defense), and Commercial. To reflect the allocation of network identifiers among the categories, a one-character code is placed to the left of the network number: R for Research, D for Defense, G for Government, and C for Commercial (see Appendix A for further details on this division of the network identification).

Network numbers are assigned for networks that are connected to the ARPA-Internet and DDN-Internet, and for independent networks that use the IP family protocols (these are usually commercial). These independent networks are marked with an asterisk preceding the number.

The administrators of independent networks must apply separately for permission to interconnect their network with either the ARPA-Internet of the DDN-Internet. Independent networks should not be listed in the working tables of either the ARPA-Internet or DDN-Internet hosts or gateways.

For various reasons, the assigned numbers of networks are sometimes changed. To ease the transition the old number will be listed for a transition period as well. These "old number" entries will be marked with a "T" following the number and preceding the name, and the network name will be suffixed "-TEMP".

Special Addresses:

In certain contexts, it is useful to have fixed addresses with functional significance rather than as identifiers of specific hosts. When such usage is called for, the address zero is to be interpreted as meaning "this", as in "this network". The address of all ones are to be interpreted as meaning "all", as in "all hosts". For example, the address 128.9.255.255 could be

interpreted as meaning all hosts on the network 128.9. Or, the address 0.0.0.37 could be interpreted as meaning host 37 on this network.

Assigned Network Numbers

Class A Networks

*	Internet Address		Name	Network	References
	000.rrr.rrr.rrr			Reserved	[JBP]
R	004.rrr.rrr.rrr		SATNET	Atlantic Satellite Network	[SHB]
D	006.rrr.rrr.rrr	T	YPG-NET-TEMP	Yuma Proving Grounds	[10,BXA]
D	007.rrr.rrr.rrr	T	EDN-TEMP	DCEC EDN	[EC5]
R	008.rrr.rrr.rrr	T	BBN-NET-TEMP	BBN Network	[JSG5]
R	010.rrr.rrr.rrr		ARPANET	ARPANET	[10,40,SA2]
D	011.rrr.rrr.rrr		DODIIS	DoD INTEL INFO SYS	[AY7]
C	012.rrr.rrr.rrr		ATT	ATT, Bell Labs	[MH12]
C	014.rrr.rrr.rrr		PDN	Public Data Network	[REK4]
R	018.rrr.rrr.rrr	T	MIT-TEMP	MIT Network	[20,103,DDC1]
D	021.rrr.rrr.rrr		DDN-RVN	DDN-RVN	[MLC]
D	022.rrr.rrr.rrr		DISNET	DISNET	[FLM2]
D	023.rrr.rrr.rrr		DDN-TC-NET	DDN-TestCell-Network	[DH17]
D	024.rrr.rrr.rrr		MINET	MINET	[10,DHH]
R	025.rrr.rrr.rrr		RSRE-EXP	RSRE	[RNM1]
D	026.rrr.rrr.rrr		MILNET	MILNET	[FLM2]
R	027.rrr.rrr.rrr	T	NOSC-LCCN-TEMP	NOSC / LCCN	[RH6]
R	028.rrr.rrr.rrr		WIDEBAND	Wide Band Satellite Net	[CJW2]
D	029.rrr.rrr.rrr	T	MILX25-TEMP	MILNET X.25 Temp	[MLC]
D	030.rrr.rrr.rrr	T	ARPAX25-TEMP	ARPA X.25 Temp	[MLC]
G*	031.rrr.rrr.rrr		UCDLA-NET	UCDLA-CATALOG-NET	[CXL]
R	032.rrr.rrr.rrr		UCL-TAC	UCL TAC	[PK]
R	036.rrr.rrr.rrr	T	SU-NET-TEMP	Stanford University Network	[PA5]
R	039.rrr.rrr.rrr	T	SRINET-TEMP	SRI Local Network	[GEOF]
R	041.rrr.rrr.rrr		BBN-TEST-A	BBN-GATE-TEST-A	[RH6]
R	044.rrr.rrr.rrr		AMPRNET	Amateur Radio Experiment Net	[HM]
	001.rrr.rrr.rrr-003.rrr.rrr.rrr			Unassigned	[JBP]
	005.rrr.rrr.rrr			Unassigned	[JBP]
	009.rrr.rrr.rrr			Unassigned	[JBP]
	013.rrr.rrr.rrr			Unassigned	[JBP]
	015.rrr.rrr.rrr-017.rrr.rrr.rrr			Unassigned	[JBP]
	019.rrr.rrr.rrr-020.rrr.rrr.rrr			Unassigned	[JBP]
	033.rrr.rrr.rrr-035.rrr.rrr.rrr			Unassigned	[JBP]
	037.rrr.rrr.rrr-038.rrr.rrr.rrr			Unassigned	[JBP]
	040.rrr.rrr.rrr			Unassigned	[JBP]
	042.rrr.rrr.rrr-043.rrr.rrr.rrr			Unassigned	[JBP]
	045.rrr.rrr.rrr-126.rrr.rrr.rrr			Unassigned	[JBP]
	127.rrr.rrr.rrr			Reserved	[JBP]

ASSIGNED PORT NUMBERS

Ports are used in the TCP [39,93] to name the ends of logical connections which carry long term conversations. For the purpose of providing services to unknown callers, a service contact port is defined. This list specifies the port used by the server process as its contact port. The contact port is sometimes called the "well-known port".

To the extent possible, these same port assignments are used with the UDP [39,91].

To the extent possible, these same port assignments are used with the ISO-TP4 [57].

The assigned ports use a small portion of the possible port numbers. The assigned ports have all except the low order eight bits cleared to zero. The low order eight bits are specified here.

Port Assignments:

Decimal	Keyword	Description	References
0		Reserved	[JBP]
1-4		Unassigned	[JBP]
5	RJE	Remote Job Entry	[17,40,JBP]
7	ECHO	Echo	[82,JBP]
9	DISCARD	Discard	[80,JBP]
11	USERS	Active Users	[76,JBP]
13	DAYTIME	Daytime	[79,JBP]
15	NETSTAT	Who is up or NETSTAT	[JBP]
17	QUOTE	Quote of the Day	[87,JBP]
19	CHARGEN	Character Generator	[78,JBP]
20	FTP-DATA	File Transfer [Default Data]	[39,83,JBP]
21	FTP	File Transfer [Control]	[39,83,JBP]
23	TELNET	Telnet	[99,JBP]
25	SMTP	Simple Mail Transfer	[39,89,JBP]
27	NSW-FE	NSW User System FE	[23,RHT]
29	MSG-ICP	MSG ICP	[74,RHT]
31	MSG-AUTH	MSG Authentication	[74,RHT]
33	DSP	Display Support Protocol	[MLC]
35		any private printer server	[JBP]
37	TIME	Time	[95,JBP]
39	RLP	Resource Location Protocol	[1,MA]
41	GRAPHICS	Graphics	[40,115,JBP]
42	NAMESERVER	Host Name Server	[39,86,JBP]
43	NICNAME	Who Is	[39,48,JAKE]
44	MPM-FLAGS	MPM FLAGS Protocol	[JBP]

45	MPM	Message Processing Module [recv]	[85,JBP]
46	MPM-SND	MPM [default send]	[91,JBP]
47	NI-FTP	NI FTP	[122,SK]
49	LOGIN	Login Host Protocol	[PHD1]
51	LA-MAINT	IMP Logical Address Maintenance	[66,AGM]
53	DOMAIN	Domain Name Server	[81,71,PM1]
55	ISI-GL	ISI Graphics Language	[14,RB6]
57		any private terminal access	[JBP]
59		any private file service	[JBP]
61	NI-MAIL	NI MAIL	[12,SK]
63	VIA-FTP	VIA Systems - FTP	[DXD]
65	TACACS-DS	TACACS-Database Service	[11,RHT]
67	BOOTPS	Bootstrap Protocol Server	[35,WJC2]
68	BOOTPC	Bootstrap Protocol Client	[35,WJC2]
69	TFTP	Trivial File Transfer	[39,102,DDC1]
71	NETRJS-1	Remote Job Service	[16,40,RTB]
72	NETRJS-2	Remote Job Service	[16,40,RTB]
73	NETRJS-3	Remote Job Service	[16,40,RTB]
74	NETRJS-4	Remote Job Service	[16,40,RTB]
75		any private dial out service	[JBP]
77		any private RJE service	[JBP]
79	FINGER	Finger	[40,46,KLH]
81	HOSTS2-NS	HOSTS2 Name Server	[EAK1]
83	MIT-ML-DEV	MIT ML Device	[DPR]
85	MIT-ML-DEV	MIT ML Device	[DPR]
87		any private terminal link	[JBP]
89	SU-MIT-TG	SU/MIT Telnet Gateway	[MRC]
91	MIT-DOV	MIT Dover Spooler	[EBM]
93	DCP	Device Control Protocol	[DT15]
95	SUPDUP	SUPDUP	[26,MRC]
97	SWIFT-RVF	Swift Remote Vitural File Protocol	[MXR]
98	TACNEWS	TAC News	[ERAN]
99	METAGRAM	Metagram Relay	[GEOF]
101	HOSTNAME	NIC Host Name Server	[39,47,JAKE]
103		Unassigned	[JBP]
105	CSNET-NS	Mailbox Name Nameserver	[113,MHS1]
107	RTELNET	Remote Telnet Service	[88,JBP]
109	POP-2	Post Office Protocol - Version 2	[19,JKR1]
111	SUNRPC	SUN Remote Procedure Call	[DXG]
113	AUTH	Authentication Service	[116,MCSJ]
115	SFTP	Simple File Transfer Protocol	[60,MKL1]
117	UUCP-PATH	UUCP Path Service	[38,MAE]
119	UNTP	USENET News Transfer Protocol	[61,PL4]
121	ERPC	HYDRA Expedited Remote Procedure Call	[118,JXO]
123	NTP	Network Time Protocol	[70,DLM1]
125	LOCUS-MAP	Locus PC-Interface Net Map Server	[124,BXG]
127	LOCUS-CON	Locus PC-Interface Conn Server	[124,BXG]
129		Unassigned	[JBP]

ETHERNET NUMBERS OF INTEREST

Many of the networks of all classes are Ethernets (10Mb) or Experimental Ethernets (3Mb). These systems use a message "type" field in much the same way the ARPANET uses the "link" field.

If you need an Ethernet number, contact the XEROX Corporation, Office Products Division, Network Systems Administration Office, 333 Coyote Hill Road, Palo Alto, California, 94304.

Assignments:

Ethernet		Exp. Ethernet		Description	References
decimal	Hex	decimal	octal		
512	0200	512	1000	XEROX PUP	[1,HGM]
513	0201	-	-	PUP Addr. Trans.	[HGM]
1536	0600	1536	3000	XEROX NS IDP	[128,HGM]
→ 2048	0800	513	1001	DOD IP	[39,91,JBP]
2049	0801	-	-	X.75 Internet	[HGM]
2050	0802	-	-	NBS Internet	[HGM]
2051	0803	-	-	ECMA Internet	[HGM]
2052	0804	-	-	Chaosnet	[HGM]
2053	0805	-	-	X.25 Level 3	[HGM]
2054	0806	-	-	ARP	[74,JBP]
2055	0807	-	-	XNS Compatability	[HGM]
2076	081C	-	-	Symbolics Private	[DCP1]
32771	8003	-	-	Cronus VLN	[116,DT15]
32772	8004	-	-	Cronus Direct	[116,DT15]
32774	8006	-	-	Nestar	[HGM]
32784	8010	-	-	Excelan	[HGM]
32821	8035	-	-	Reverse ARP	[42,JCM]
36864	9000	-	-	Loopback	[HGM]

The standard for transmission of IP datagrams over Ethernets and Experimental Ethernets is specified in RFC 894 [54] and RFC 895 [76] respectively.

ASSIGNED ADDRESS RESOLUTION PROTOCOL PARAMETERS

The Address Resolution Protocol (ARP) specified in RFC 826 [75] has several parameters. The assigned values for these parameters are listed here.

Assignments:

Operation Code (op)

1 REQUEST
2 REPLY

Hardware Type (hrd)

Type	Description	References
1	Ethernet (10Mb)	[JBP]
2	Experimental Ethernet (3Mb)	[JBP]
3	Amateur Radio AX.25	[PXK]
4	Proton ProNET Token Ring	[JBP]
5	Chaos	[GXP]

Protocol Type (pro)

Use the same codes as listed in the section called "Ethernet Numbers of Interest" (all hardware types use this code set for the protocol type).

John Leong, Carnegie-Mellon University, Pittsburgh, Pa.

Nuts-and-bolts guide to Ethernet installation and interconnection

A large university reports on some frequently overlooked problems faced when linking several departmental networks.

Currently, various departments within Carnegie-Mellon University run 12 separate 10-Mbit/s Ethernet networks. The networks vary in geographical size, ranging from small backbone networks that connect computing facilities inside a computer room to big department networks with 50 or more stations covering a broad area of the campus. The larger networks tend to be populated by a higher percentage of transient stations or microcomputers and terminals that leave and rejoin the network frequently.

There are, at present, three versions of the Ethernet specification:

- Ethernet specification version 1, September 1980.
- Ethernet specification version 2, November 1982.
- IEEE 802.3 carrier-sense multiple access with collision detection (CSMA/CD) specification, revision D, December 1982.

Version 1 is the original Ethernet specification jointly developed by Digital Equipment Corp. (DEC), Intel Corp., and Xerox Corp. The majority of Ethernet products available today are designed according to this specification. Yet in the process of making this specification an IEEE standard, some changes have been made. The main differences among the three versions lie in the connection from the data-terminal equipment (DTE) controller to the media-access unit (MAU) transceiver interface.

For instance, the line-idle state was changed from 0.7 volts idle (version 1, Fig. 1A) to zero volts idle (version 2, Fig. 1B). As a result, the first bit of the message preamble for version 2 starts with a voltage half step instead of a voltage full step. Differences between the versions also exist with the collision-present test as well as with the interface coupling specification.

Because of these differences, a controller designed according to one specification may not work with a transceiver designed according to another. In the past, this has not been much of a problem, since most of the equipment on the market conforms to the Ethernet version 1 specification. However, within the past few months, equipment conforming to the official IEEE 802.3 standard has started to appear in quantity. Compatibility problems arise from the Manchester encoder/decoder chips now on the market, which are designed for the IEEE 802.3 specification with Ethernet version 1 support only optional. (The Ethernet version 2 specification was introduced as an attempt to bring version 1 in line with IEEE 802.3, but it has not generated much interest.)

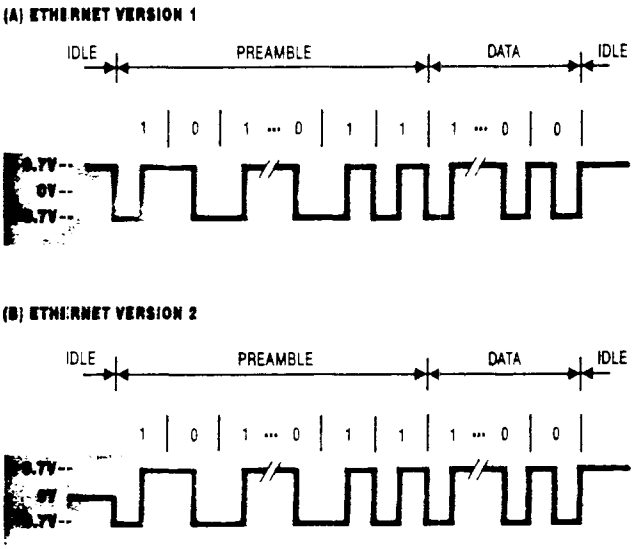
Down to earth

An often overlooked aspect of Ethernet installation is cable grounding. All the specifications stress the importance of cable grounding at a single point to ensure reliable communications. Still, networks are frequently installed with either multiple groundings or no ground connection at all. And though networks so installed work in most cases, stability problems can easily creep in as the sites grow.

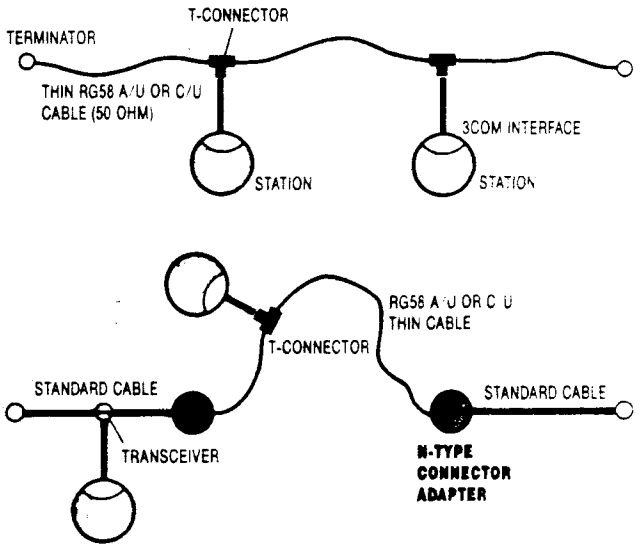
For proper installation, cable connectors and terminators should be covered by insulating sleeves to prevent accidental grounding. During installation and maintenance, accidental grounding can be a potential safety hazard. Detailed safety issues are described in the IEEE 802.3 Ethernet specification.

The most common Ethernet cable is the standard 50-ohm coaxial cable covered with yellow polyvinyl chloride (PVC). Carnegie-Mellon uses Beldon 9880 cable, which costs roughly $0.80 a foot. (Similar cable is available from other manufacturers.) The basic PVC

1. Incompatibilities. *Ethernet version 1, Fig. 1A specifies an idle of 0.7 volts; version 2, Fig. 1B specifies an idle of zero volts. Version 2's preamble is a half step.*



2. Thick and thin. *RG58 50-ohm cable is cheap and flexible and has high signal attenuation. An N-type adapter allows connection with Ethernet cable.*



cable cannot be used for plenum (air space in the environment) installation. A Teflon-coated cable is required for these applications. Teflon is a much stiffer cable and has a large bending radius. Predictably, it is more difficult to install and costs up to four times as much as the PVC cable. But in most cases, the Teflon alternative is still cheaper than installing new conduits.

The maximum length of an unrepeatered Ethernet segment is 500 meters. Cable can be installed in short sections joined by connectors. According to the 802.3 specification, the preferable section lengths are 23.4, 70.2, and 117 meters. These lengths will produce minimum signal reflection. To ensure an even cleaner signal, the cable sections should be obtained from the same manufacturer and—even better—from the same manufacturer's lot.

According to the IEEE specification, transceivers must be placed at least 2.5 meters apart. It is therefore desirable to have the cable marked at those intervals. Some cable comes with marking already on the jacket. If not, most dealers will mark the cable at no additional cost, since they have to measure out the cable anyway. During installation, number the markings sequentially to make troubleshooting the network easier.

For operation and maintenance, it is desirable to have accurate and detailed documentation of the cable installation. This should include not only the location of the cable, but also descriptions of the cable trays, raceways, and conduits that the cable traverses. Such documentation is valuable for troubleshooting and future expansion (DATA COMMUNICATIONS, "How to avoid some common pitfalls of a local net installation," March 1984, p. 243).

Another popular type of cable is the 50-ohm RG58 A/U or C/U. This is a thin, flexible, inexpensive cable that costs approximately $0.09 per foot. Both Ungermann-Bass and 3Com support this type of cable—especially with their microcomputer products. When RG58 is used with the 3Com Etherlink IBM PC interface, no transceiver is required. This represents a savings of approximately $250 per connection. Figure 2 shows a configuration using such a cable. Note that intermixing the RG58 cable with the regular Ethernet cable is allowed.

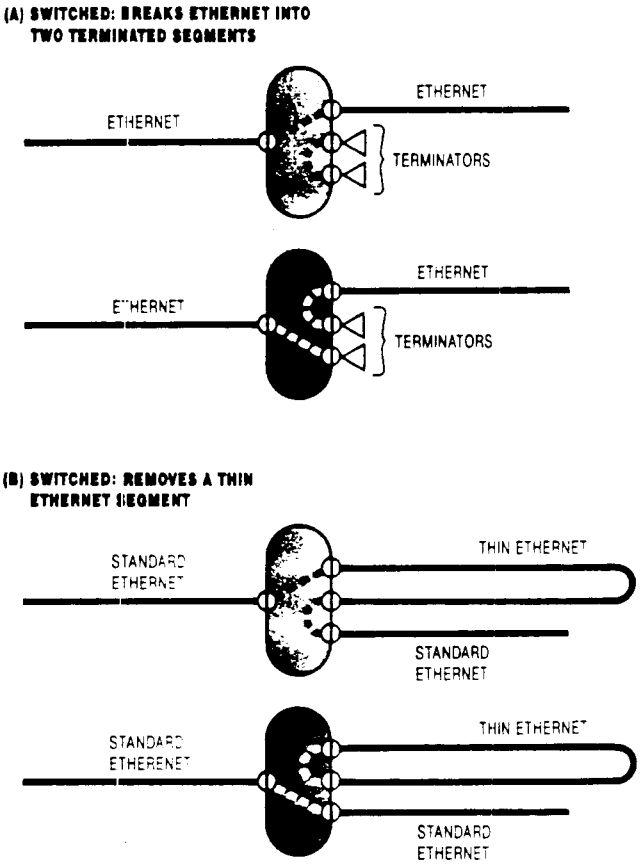
Weighing against the RG58 cable's low cost and ease of installation is its exposure to damage. With the standard Ethernet cable, the main cable is typically hidden in the ceiling or under the floor, and only the drop cable is exposed. Damage to or disconnection of a drop cable will normally have no effect on the overall network. But with the RG58, there is no drop cable. This thin Ethernet cable is exposed and runs directly into the stations. Damage or disconnection (at the T-connector) can bring down the entire network. Another RG58 disadvantage is its high signal attenuation, which can reduce the maximum permissible segment length by a factor of three. In general, the RG58 thin cable is suitable for use in a small environment with microcomputers. Further, if this cable is selected, the authentic RG58 A/U or C/U should be purchased. The cheaper RG58-like cables are not as good and should be avoided.

Links

There are two popular types of connector—crimp-on and screw-on. The crimp-on type requires a tool for installation. Barrels and connectors cost approximately $5 apiece from Amphenol. Amphenol also sells crimp-on tools for $120. The older version of the screw-on connector comes with many pieces and is difficult to assemble.

A newer connector from Cambridge Products consists of only two parts: a contact and the connector itself. It requires no special tool and is relatively simple

3. Critical placement. *Special circuit-insertion relays: (A) shows Ethernet terminated in two discrete sections, (B) presents a thin-wire Ethernet for fault isolation.*



to install. This connector works well and costs from $3.50 to $4.75 apiece. Plastic boots or sleeves should be used with the connectors to provide ground isolation.

Placement of circuit insertion relays at strategic points of the Ethernet cable can be invaluable for fault isolation. Carnegie-Mellon has used N-series coaxial RF (radio frequency) relays from Amphenol. These cost about $100 and can be configured either for breaking the Ethernet cable into two discrete terminated segments (Fig. 3A), or for taking a loop out of troublesome cable (Fig. 3B). The loop configuration is particularly useful when using a mixture of thick and thin Ethernets. If the thin cable is damaged, it can be easily taken out of service.

Drop cable can either be homemade or bought off-the-shelf. Almost every company that sells transceivers also sells drop cables. If the cable must be threaded through narrow conduits or other tricky access paths, one should get cables with shells that can be taken apart. DEC offers two grades of cable: the standard drop cable and a thinner, more flexible variety. The more flexible grade is easier to install but has roughly four times the attenuation.

Carnegie-Mellon has used a variety of homemade drop cables. They range in cost from $0.30 to $0.60 a foot. Although all of the cables worked well electrically, some were better than others in terms of their thickness, flexibility, and the ease of installation. Carnegie-Mellon has had positive experience with both Phalo 2384 and Belden 9891 cables.

Before selecting a transceiver, one should make sure that it matches the controller. That is, a controller conforming to the Ethernet version 1 specification should be used with a version 1 transceiver. And an IEEE 802.3 controller should be used with a similar transceiver. The transceiver must supply the collision-present test if the controller requires it.

Taps

Transceiver installation can be tricky since the transceiver must be placed directly on the Ethernet cable; and often, the cable tray or conduit is difficult to access. Furthermore, most of the conduits and cable trays are often already congested and are not designed for placement of bulky objects like transceivers. Such forced transceiver installation into crowded conduits can cause shorting. This is the most common cause of Ethernet failure.

Carnegie-Mellon has used 3Com, TCL, DEC, and Interlan transceivers. Of the four, 3Com is the only invasive type. It must be inserted in series on the cable and requires cutting the cable during installation. All the other transceivers are of the "vampire" tap variety, clamping directly onto the cable with metal teeth. The 3Com transceiver does not supply a heartbeat, or collision-present test signal. Nevertheless, it works well without interfaces and can be placed less than 2.5 meters apart.

The Interlan and DEC transceivers have the same physical housing but different electronics. Both are physically bulky. They require special, and rather costly, installation tools. Removing these units from the cable is also difficult.

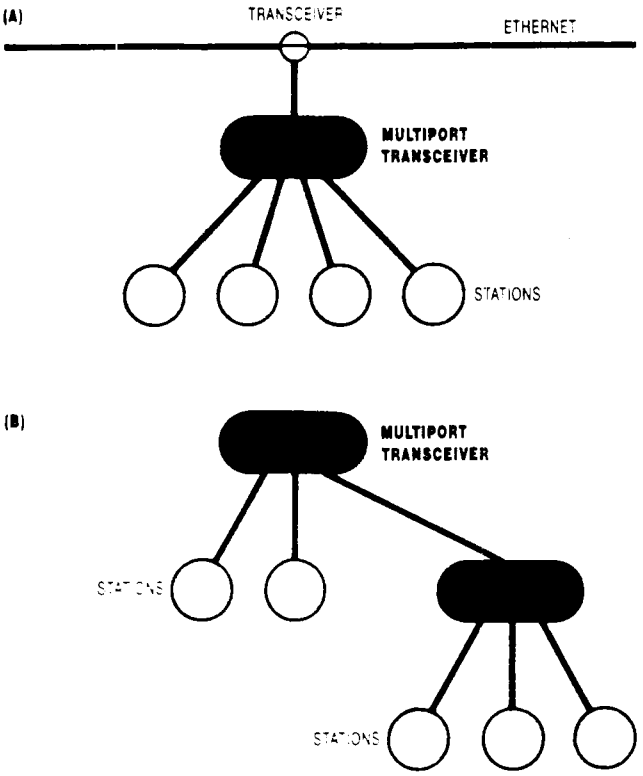
The TCL transceiver is smaller than the DEC and 3Com devices. It is easier to install and remove, though it does not fasten as securely to an Ethernet cable as do the others. The Interlan transceivers come in two varieties: the E-series and the I-series. The E-series conforms to Ethernet version 1, and the I-series conforms to IEEE 802.3.

Carnegie-Mellon has been experiencing compatibility problems with the new IEEE 802.3 interface boards because it has mainly installed Ethernet version 1 transceivers. All the transceivers cost from $200 to $250. Some are available with a substantial quantity discount. The university prefers the TCL model and has installed more than 100 of them.

Multiport transceivers

All multiport transceivers support two operating modes: They can be connected to a transceiver and provide attachments to multiple stations or used as standalones. When standalone, the multiport transceiver itself functions as an Ethernet network. By cascading devices (Fig. 4B), most models will support up to 64 stations. When used in-line with an Ethernet network transceiver (Fig. 4A), the maximum distance

4. Cascading. *Multiport Ethernet transceivers can act as separate local networks and can be cascaded to include as many as 64 stations.*



between the Ethernet and the station is typically reduced from 50 meters to 40 meters. This is due to the added delay incurred in the multiport transceiver.

Not only can a multiport transceiver reduce the per-connection cost, but it can also provide a degree of centralized control for problem determination and isolation. In addition, this type of transceiver eliminates the 2.5-meter separation requirement between devices and is particularly useful in a cluster situation. In one department at Carnegie-Mellon, minicomputers are connected to one multiport transceiver.

Repeaters

Repeaters connect multiple Ethernet segments. They are either local or remote. Local repeaters can be used if the two Ethernet segments are less than 100 meters apart. Otherwise a remote repeater must be used. Ungermann-Bass and DEC sell both types of repeaters.

A local repeater is typically housed in one cabinet with two transceiver-cable connectors, one for each Ethernet segment. A remote repeater consists of a pair of boxes containing half-repeaters, one for each segment of the Ethernet. Two half-repeaters are connected by point-to-point links of up to 1 kilometer. Since remote repeaters are mainly used for interbuilding connections, and taps are rarely required between the two repeaters, optical fiber is often used for this link. Fiber-optic cable also avoids the problems that lightning and other electrical interference can cause.

The fiber-optic cable used by AT&T and other Bell operating companies (BOCs) is the 50-micron or 62.5-micron variety. Although this type of cable is good for long-distance connections, it is not as suitable for local network environments, where connector loss is a significant factor. Furthermore, since the 50-micron fiber-optic cable has 25 percent of the surface area of the 100-micron light source, the 75 percent light loss—approximately 8 decibels (dB)—can be damaging. As a result, the university has had difficulty using the DEC remote repeater, which has a flux budget (total amount of repeater power) of 10 dB. The Ungermann-Bass repeater fares better with a flux budget of 16 dB.

While remote repeaters from both DEC and Ungermann-Bass specify 100-micron fiber-optic cables, the university has used them over its 50-micron cable plant with no serious problem. Repeaters from both manufacturers have status lights for each segment. These are useful aids for indicating incoming data and collision detection.

If one segment to the DEC repeater goes down, the device will isolate that segment from the network, rather than drag the other side down. In addition, it is possible to deploy a spare DEC repeater in parallel with the main unit and have the spare set for standby mode. When the main unit fails to repeat eight full packets, the spare repeater will automatically take over.

Other Ethernets

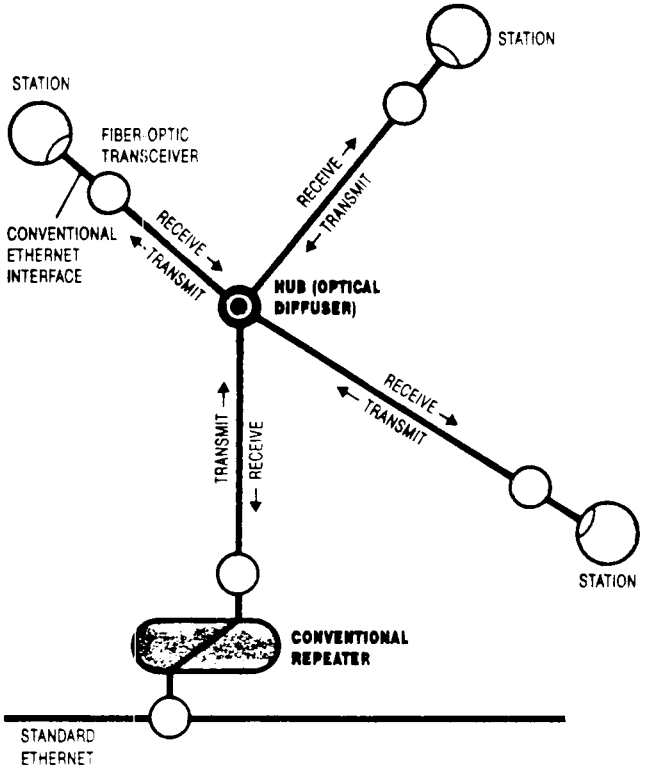
Both Codenol and Ungermann-Bass market fiber-optic Ethernets. These are star-shaped networks with optical defusers at the hubs. A station is connected to the hub through a pair of fiber-optic cables (Fig. 5). The hub ranges in price from $900 to $4,000, depending on the number of ports (5, 8, 16, or 32 ports).

At present there are two basic problems with this technology. In the first instance, the flux loss by the hub is relatively high—especially with 50-micron cable. The second problem is with collision detection in an asymmetrical network: If the network consists of stations a long way from the hub, as well as some that are close to the hub, a station close to the hub may not be able to detect collision with a station that is far away. Ungermann-Bass overcomes this problem by placing an optional collision-detection unit at the hub, but the unit is quite expensive ($12,000 for seven ports and $20,000 for 14 ports).

American Photonics has a fiber-optic drop cable called an Ethernet expander—model number RL5000. The expander allows stations to connect to a transceiver over an optical-fiber pair at distances up to 1 km. While this violates the Ethernet drop-cable distance specification, the device works if the overall network topology does not exceed the Ethernet-specified slot-time limitation. Careful network planning is required when using this product. The RL5000 is listed for $3,400.

Ungermann-Bass also offers a broadband 5-Mbit/s network over a 6-MHz channel. The network uses CSMA without CD and is therefore subject to a higher probability of packet loss. IBM's recently announced PC Network uses CSMA/CD over broadband. The

5. Stars. *Fiber-optic Ethernets are usually configured as star networks radiating from a single hub. A station is connected to a hub by a pair of fiber cables.*



interface board contains a simple fixed-frequency RF modem, an Intel 82586 CSMA/CD chip, an Intel 8088 processor, and 128 kbytes of RAM. Although the IBM network uses CSMA/CD, it is not Ethernet compatible in the way it implements its network parameters. Furthermore, it operates at sub-Ethernet speeds (2 Mbit/s) over a 6-MHz channel.

Monitoring tools

Excelan makes an Ethernet monitor called the Nutcracker. The product is similar to serial-line monitors commonly used on conventional data communications networks. The Nutcracker allows capture of all network traffic with or without user-specifiable filters. The device, however, has no special provision for protocol interpretation and display. At $50,000, the Nutcracker is an expensive diagnostic tool.

For general protocol development and network administration not requiring the capture of continuous back-to-back packets, there are a number of diagnostic software packages available for popular machines such as the IBM PC. These are often free.

Problem determination

An operation logbook can also be a valuable problem-solving tool. Lots of problems are caused when installing or changing connectors or transceivers. Often, matching a problem's symptom with the most recent logbook entry can help users locate the problem.

Another important tool is a set of cabling documents for the plant. These help tremendously in locating suspected faulty components.

The availability of cable status indicators can also be useful. These indicators—the most common and useful of which are the receive-data and collision-detection lights—are available on most Ethernet repeaters. With these, it is possible to determine the network load and the frequency of collisions. If there are no repeaters on the network, signal indicators can be purchased for approximately $150.

For a network of any significant size, users should purchase a time domain reflectometer (TDR). These devices help determine the position of cable shorts, breakages, bad connectors, sharp cable kinks, and other abnormalities. TDRs have been used with the telephony and cable industry for almost a decade. Essentially, the unit sends out a sample pulse. Depending on the resultant return pulse, abnormalities on the cable can be observed on the device. These devices sell for approximately $5,000.

Once the distance between the test point and the suspected problem location is known, it is necessary to find out exactly where that section of cable is on the network. If the cable is marked at 2.5-meter intervals, and each marker is sequentially labeled, it is easy. For example, if a problem is 250 meters away from a test point near marker 5, the nearest marker to the problem area should be number 105. With a cable-plant diagram, it is easy to get an idea of the area in the building where marker 105 should be.

Common problems

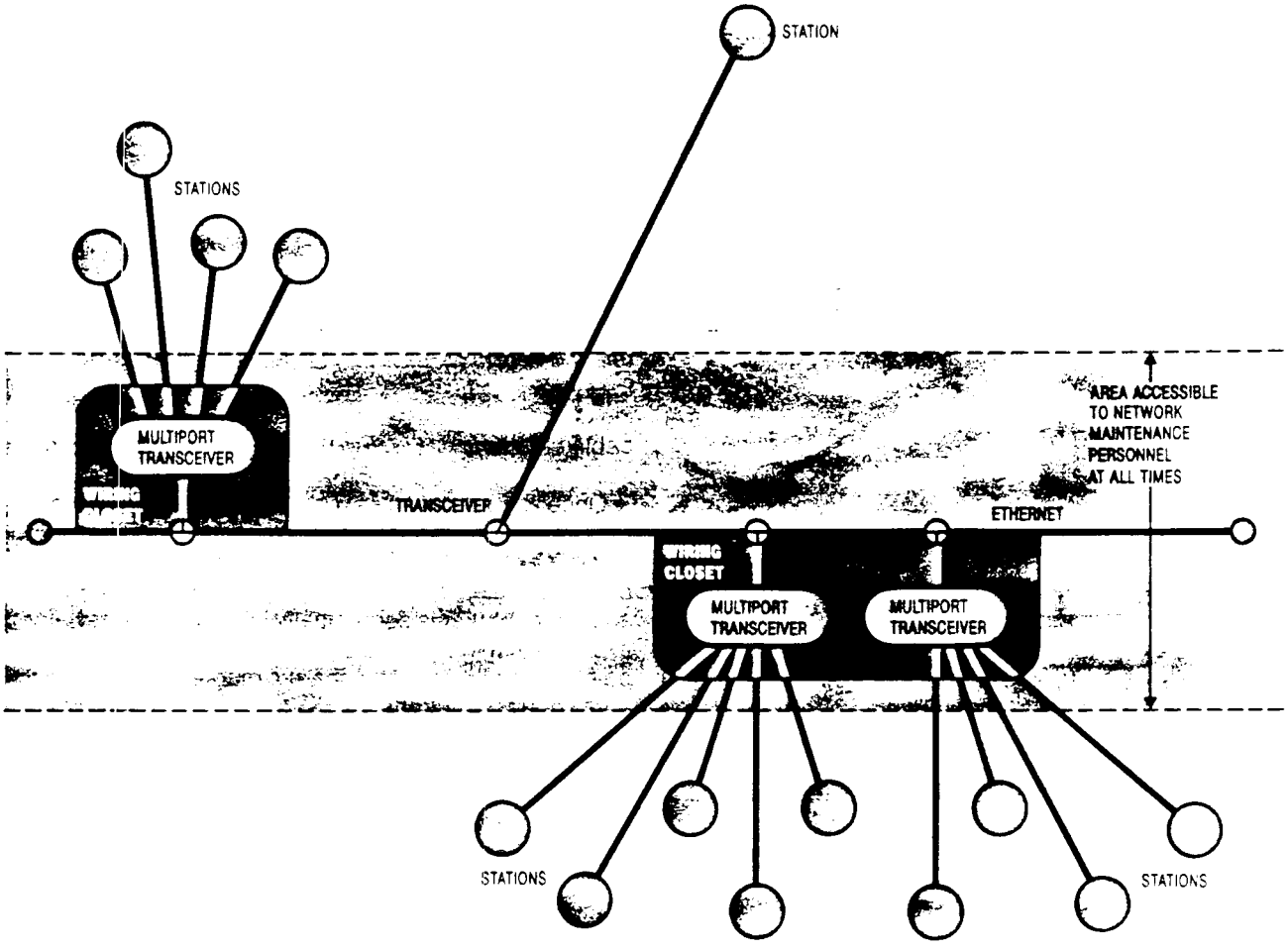
In general, Ethernet and related problems produce one of three symptoms: no network communications, a station malfunction, or network performance degradation.

- *No network communications.* This symptom is probably the result of a cable break, a short, or a jabber problem (in which a device sends data out of control). In all three instances, the problems can be detected with an ohm meter. In the case of jabber, the cable impedance will be normal, whereas impedance will be exceptionally high with a cable break and low with a shorting. A TDR will also allow quick detection of shorting and cable breaks.

If the jabber-control logic on all the transceivers is working properly, this problem should not occur. If not, it is sometimes possible to determine which station is jabbering by monitoring the packets and collision fragments in the network. If this is not possible, and if the network is small, stations may be powered-off, one by one, until the offending station is found. A potential problem with this approach is that the stations are often located inside locked rooms. In this case, it helps if the stations are connected to transceivers that are accessible. Stations can then be disconnected from the transceiver without entering the room.

If the network is large, the problem must first be isolated to a small manageable section of cable. This is applicable to all three classes of problems. Cable segmentation can be done in the following order:

6. Easy access. *It is desirable to configure large networks as a series of star networks connected by a bus. The stars are multiport transceivers, the bus is the Ethernet. But with this configuration it is essential that maintenance personnel have complete access to all transceivers on the network.*



power-off repeaters, switch RF relays, disconnect the cable where the invasive transceivers are located, and power-off multiport transceivers.

From an operational point of view, it is highly desirable to configure the network as a series of stars or stars connected by a bus (Fig. 6). The stars are multiport transceivers in strategically located wiring closets with drop cables radiating from them to stations. The bus is the Ethernet. Although this topology calls for longer drop cables, it provides network operations staff with focal points for troubleshooting and maintenance.

■ *Station malfunction.* This is most likely due to a problem with the station hardware, the Ethernet interface, station software, the transceiver unit, and/or the drop cable. If it works to replace the suspected station with a good station, the problem is likely to be with the suspected station, not with the transceiver. For this type of testing, it would be ideal if all the stations implemented the configuration-testing protocol defined in the Ethernet version 2 specification. But few machines offer it. Some networks use a higher-level protocol that provides signal echo. Although echo provides less comprehensive testing than the configuration testing protocol, it is useful as a go/no-go test.

■ *Network performance degradation.* It is possible for a station to fail in such a way that it will continuously send out maximum-size packets. Although this will not completely overload the network, the overall performance can be significantly degraded. This problem can often be identified with a network monitor, especially one that can show the network's heavy users.

Such problems are not frequent unless the cable is often being worked on or network management has been careless about the cable. At Carnegie-Mellon the Ethernet networks have been more than satisfactory, even under relatively heavy loading. The university has supported a variety of uses—from low traffic to high-response requirement applications, such as terminal emulation or paging across networks to disk servers—and has not experienced any performance problems due to the Ethernet networks. ■

John Leong received a B. S. and an M. S. in computer science from the University of Manchester, Manchester, England.

I. ranges of logical channels

The range of logical channel numbers (LCNs) 0 through 255, inclusive, is used. The logical channel group number (LCGN) is always zero (0). This means that there may no more than 256 simultaneous connections over an AX25 link.

This protocol conforms to Annexes A through G of CCITT-X.25, where applicable. Many parts of the CCITT standard are not applicable, since they pertain to facilities which are not used.

II. revision history

The first version of this specification was dated 28 November 1986. Since that version, the following changes have been made:

- The network_name used to denote this protocol was changed from "AX.25" to "AX25" (section 8.1.1).
- Diagnostic packets were eliminated from the protocol (section 3).
- Cause and diagnostic code bytes were eliminated from Restart and Clear packets (section 5).
- Various areas of ambiguity were clarified.

DEFENSE COMMUNICATIONS AGENCY

DDN PROTOCOL HANDBOOK

Volume Two

DARPA INTERNET PROTOCOLS

DECEMBER 1985

Editors:

Elizabeth J. Feinler
Ole J. Jacobsen
Mary K. Stahl
Carol A. Ward

Additional copies of this document may be obtained from the DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Room EJ291, Menlo Park, CA 94025 or from the Defense Technical Information Center (DTIC), Cameron Station, Alexandria, VA 22314.

Cost Approx. $100.00

INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

RED BOOK

VOLUME VIII – FASCICLE VIII.3

DATA COMMUNICATION NETWORKS INTERFACES

RECOMMENDATIONS X.20-X.32

VIIITH PLENARY ASSEMBLY
MALAGA-TORREMOLINOS, 8-19 OCTOBER 1984

Geneva 1985

ISBN 92-61-02321-5

AVAILABLE FROM OMNICOM INT.'L
501 CHURCH ST. N.E., SUITE 304
VIENNA, VA 22180 703/281-1135

COST APPROX $30

package Tcp_Ip_Test

Tcp_Ip_Test is a test program, run from the system console, which may be used to test the EXOS/204 Ethernet controller and the EXOS 8010 TCP/IP controller software. It is intended for use as a diagnostic tool for manufacturing and field support.

The user interface of Tcp_Ip_Test is a rudimentary command line interpreter, based on the console_manager. The useful commands are documented here.

Before you can use Tcp_Ip_Test, you must install it on your machine, along with a file containing the controller code. Here's how.

1. Load the Ethernet controller software. Mount the tape labelled "Ethernet controller software". At the system console:

```
EEDB: elab om                          -- Elaborate om_tests.
...
om_tests: read_tape                    -- Read in software.
om_tests: quit
EEDB: unelab om                        -- Unelaborate om_tests.
Subsystem:
...
EEDB:
```

2. Install the TCP_IP_TEST_PROC software. Mount the TCP_IP_TEST_PROC tape. At the system console:

```
EEDB: read_tape                        -- Read in the exerciser.
EEDB: build tcp_ip_test                -- Create a new configuration.
existing configuration: mux
parent subsystem: omm
Subsystem.Version: tcp_ip_test_proc.xx.yy.zz
Subsystem.Version:
EEDB: snapshot                         -- Snapshot all these changes.
```

Once Tcp_Ip_Test is installed, it can be run repeatedly. To start the exerciser:

```
EEDB: elab tcp_ip_test
```

Type ctrl-Z to get to the TCP/IP test program prompt.

```
====> TCP/IP test program <====
command:
```

If the exerciser appears to be hung up, please call JMK to look at it. If that's not possible, the exerciser can be stopped and restarted. Type ctrl-Z to get to the EEDB prompt:

```
EEDB: unelab tcp_ip_test
subsystem:
...
EEDB: elab tcp_ip_test
```

The simplest test to run is AUTO: it runs on any machine with a controller installed.

A slightly better test is FCC: it requires that you have a transceiver or loopback plug connected to the controller.

The most complete test requires two R1000's, with an Ethernet connecting them. First, run the Passive test on one machine. Make a note of the host_id (Internet address) of the Passive machine (it is displayed on the console). Then run the Active test on the other machine, with the remote host_id set to the host_id of the Passive machine. The minutes parameter on the Passive machine should be set a little longer than on the Active machine, to allow for the time it takes you to move from one machine to the other when starting the test. The other parameters can be set to any legal values, but must be the same on both sides (Active and Passive). The recommended settings are:

```
maximum packet size => 1024
minimum packet size => 1
last socket  => 1
first socket => 1
```

procedure Active

```
====> TCP/IP test program <====
command: active
```

Description

Test one or more active TCP/IP connections.

This test should be run after Tcp_Ip_Test.Passive has been started on a remote machine. You must know the host_id (Internet address) of the remote machine: this information is displayed on that machine's console when the Ethernet controller is downloaded, and also when the Passive test is started.

A set of tasks are created which initiate connections to sockets in the range first socket .. last socket, all on the specified remote machine. Once a connection is established, each task transmits and receives data. Received data are checked, on the assumption that they came from a machine running the Tcp_Ip_Test.Passive.

Parameters

`remote host : byte_string;`

The 4-byte Internet address of a remote machine. That machine should already be running TCP_IP_Test.Passive.

`minimum packet size : Positive;`

The length of the shortest data packet to transmit, in bytes.

`maximum packet size : Positive;`

The length of the longest data packet to transmit, in bytes.

`first socket : Positive;`

The lowest numbered socket on which to establish a connection.

`last socket : Socket_Number;`

The highest numbered socket on which to establish a connection.

`minutes : Socket_Number;`

The number of minutes to run the test.

Restrictions

Do not specify a remote address other than 4 bytes long. If you do, the test program will fall over dead.

Do not specify packet lengths (either minimum or maximum) less than 1 or greater than 1024. If you do, the test program will fall over dead.

Do not specify a socket range containing more than 3 sockets. If you do, the controller will crash, because of a bug in its memory management software. The socket values aren't important, just don't have more than 3 going at once.

References

Passive

Loopback

---

procedure Auto

```
====> TCP/IP test program <====
command: auto
```

Description

Test one loopback TCP/IP connection.

Parameters

```
minutes : Positive;
```

The number of minutes to run the test.

References

Loopback

procedure Boot

```
====> TCP/IP test program <====
command: boot
```

Description

Bootstrap the Ethernet controller.

Reset the controller, download its code file, and start it running. This procedure always boots the controller, even if it has already been booted.

The other Tcp_Ip_Test procedures will automatically boot the controller if they have to. You need not run this procedure to use the tests.

The controller's Internet address is copied from the machine.ID (cluster_id) of the local machine. This is a crude way to ensure that different machines have different addresses.

type Byte_String

```
type Byte_String is array (range <>) of byte;
```

Description

A string of 8-bit bytes.

To enter a byte string, enter each byte in hexadecimal notation, separated by spaces. Alternatively, you may type a quoted string, in which case each character in the string will be converted to a byte (ASCII code).

An empty line terminates input.

procedure Fcc

```
====> TCP/IP test program <====
command: fcc
```

Description

Transmit raw Ethernet packets to the transceiver, and check to make sure they are echoed.

This procedure assumes that the Ethernet controller is connected to a working transceiver, or to a loopback plug. It transmits random packets out the transceiver interface. The controller automatically checks to make sure that every packet transmitted is also received: if not, error messages are displayed on the console.

This test can be used as a simple check of the transceiver and transceiver cable.

Parameters

```
minutes : Positive;
```

The number of minutes to run the test.

---

procedure Loopback

```
====> TCP/IP test program <====
command: loopback
```

Description

Test one or more loopback TCP/IP connections.

A set of tasks are created which wait for incoming connections on sockets in the range first_socket .. last_socket. Fifteen seconds later, another set of tasks are created which initiate connections to the same sockets, using the controller loopback feature. Each pair of tasks (one active, one passive) transmits data to each other for the specified number of minutes. Data transmitted by one task are checked by the other.

Parameters

`minimum packet size : Positive;`

The length of the shortest data packet to transmit, in bytes.

`maximum packet size : Positive;`

The length of the longest data packet to transmit, in bytes.

`first socket : Socket_Number;`

The lowest numbered socket on which to establish a connection.

`last socket : Socket_Number;`

The highest numbered socket on which to establish a connection.

`Minutes : Positive;`

The number of minutes to run the test.

Restrictions

Do not specify packet lengths (either minimum or maximum) greater than 1024 or less than 1. If you do, the test program will fall over dead.

Do not specify a socket range containing more than 3 sockets. If you do, the controller will crash, because of a bug in its memory management software. The socket values aren't important, just don't have more than 3 going at once.

References

Active

Passive

Fcc

---

procedure Passive

```
====> TCP/IP test program <====
command: passive
```

Description

Test one or more passive TCP/IP connections.

This test is usually run in conjunction with Tcp_Ip_Test.Active running on another machine. The Passive test is run first. Once it is started, the Active test is run on the other machine.

This test displays the Internet address of the local machine on the console when it starts. Make a note of this address: you will need it to run the Active test.

A set of tasks are created which wait for connections on sockets in the range first socket .. last socket. Once a connection is established, each task transmits and receives data. Received data are checked, on the assumption that they came from a machine running Tcp_Ip_Test.Active.

Parameters

`minimum packet size : Positive;`

The length of the shortest data packet to transmit, in bytes.

`maximum packet size : Positive;`

The length of the longest data packet to transmit, in bytes.

`first socket : Socket_Number;`

The lowest numbered socket on which to establish a connection.

`last socket : Socket_Number;`

The highest numbered socket on which to establish a connection.

`minutes : Positive;`

The number of minutes to run the test.

Restrictions

Do not specify packet lengths (either minimum or maximum) greater than 1024 or less than 1. If you do, the test program will fall over dead.

Do not specify a socket range containing more than 3 sockets. If you do, the controller will crash, because of a bug in its memory management software. The socket values aren't important, just don't have more than 3 going at once.

References

Active

Loopback

---

type Socket_Number

```
type Socket_Number is range 1 .. 2 ** 16 - 1;
```

Description

A number identifying a TCP/IP socket.

Socket_Numbers 1 .. 256 are reserved for particular uses.

References

Active

Passive

Loopback

end Tcp_Ip_Test;

Asynchronous X.25

This document specifies a data communication protocol for use with an asynchronous RS-232 physical medium.

The protocol is substantially identical to CCITT-X.25, that is, CCITT recommended standard X.25, as described in the CCITT Red Book, Fascicle VIII.3. This document follows the organization of the X.25 standard, describing only those areas that are different. It is strongly recommended that you read and understand the X.25 standard in conjunction with this document.

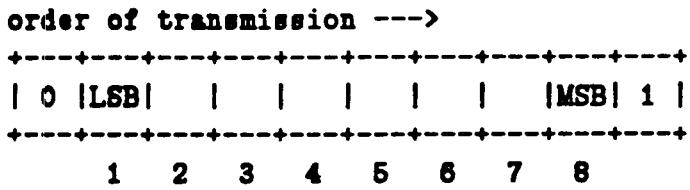
The service interface to the protocol is an extension of package Transport, as described in the Rational Networking Manual, Rational Document Control Number 700, July 1986. It is strongly recommended that you read and understand this relevant sections of the manual in conjunction with this document.

Table of Contents

1. physical level

The physical interface is specified by ANSI RS-232c, or CCITT V.24. This is entirely different from the physical interface specified by X.25, which is a synchronous interface specified by CCITT X.21.

Asynchronous, full-duplex transmission is used. Each octet is transmitted as one 8-bit character, with no parity.



The idle line state is continous 1's. Each octet must be preceded by one bit = 0 (the start bit), and followed by at least one bit = 1 (the stop bit). The bits of each octet are transmitted least significant bit first.

The transmission speed must be agreed upon by both parties to communication. The default speed is 9600 bits/second.

There is no flow control at the physical level. This is because correct operation of the higher level protocols requires immediate transmission of all data: delays imposed by flow control will cause timeouts to expire erroneously, and delayed delivery of frames will cause protocol violations.

Control characters (for example XON/XOFF) must not be used for flow control. This is because the transmitted data can include any bit pattern, including control characters.

2. link access procedures

2.1. scope and field of application

Same as CCITT-X.25. This protocol is limited to the following options:

- LAPB. LAP is not used.
- Single link. MLP is not used.
- Modulo 8 sequence numbers.

2.2. frame structure

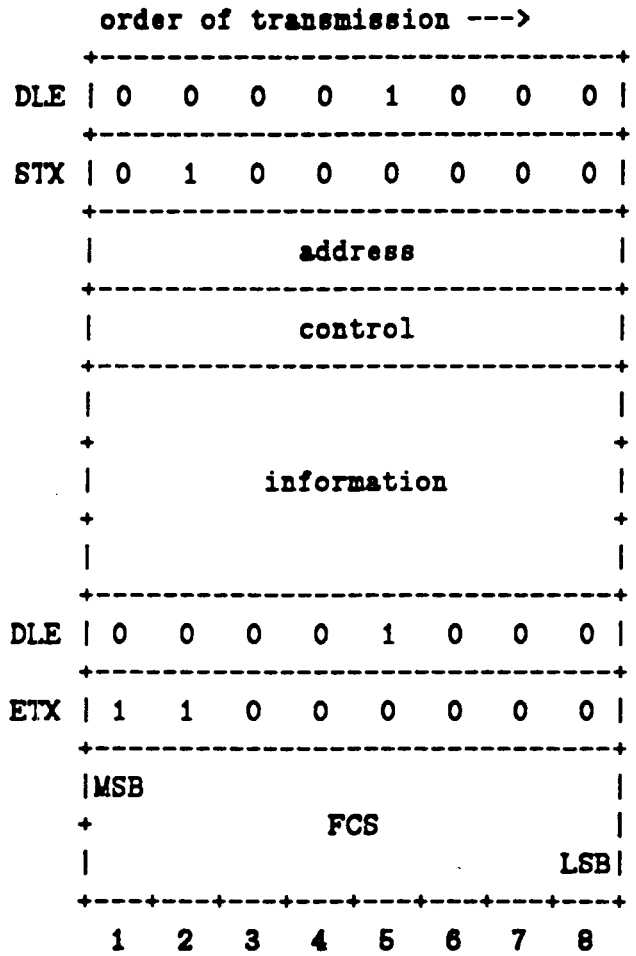
2.2.1. Introduction

The frame format is taken from IBM Binary Synchronous Communication (Bisync) Transparent-Text mode, adapted for an asynchronous environment. This is entirely different from CCITT-X.25, which specifies the HDLC frame format.

The Frame Check Sequence, and the format of address, control and data within a frame, are the same as CCITT-X.25.

2.2.2. frame format

Each frame has the following structure:



The start of frame marker is DLE STX, and the end of frame marker is DLE ETX. The Frame Check Sequence (FCS) immediately follows the end of frame marker.

2.2.3. address field
Same as CCITT-X.25.

2.2.4. control field
Same as CCITT-X.25.

2.2.5. information field
Same as CCITT-X.25, with the restriction that the information field must be octet-aligned (a multiple of 8 bits in length).

2.2.6. transparency
Between the start of frame and end of frame markers, any occurrence of the octet DLE will be doubled, that is, the sequence DLE DLE will be transmitted to represent such an octet. The FCS is tranmitted verbatim, without doubling DLEs.

2.2.7. frame check sequence field
CRC-CCITT is used, as in CCITT-X.25. The FCS is calculated over the octets between the start of frame and end of frame markers, exclusive. That is, the start of frame and end of frame markers are not included in the FCS.

2.2.8. order of bit transmission
Same as CCITT-X.25. All octets except the FCS are transmitted least significant bit first. The FCS is transmitted most significant bit first.

2.2.9. aborting a frame
There is no provision for aborting a frame. A frame is transmitted entirely or not at all.

2.3. LAPB elements of procedure
Same as CCITT-X.25.

2.4. description of the LAPB procedure
Same as CCITT-X.25.

2.5. Multilink procedure
Not applicable. The multilink procedure is not used.

2.6. LAP elements of procedure
Not applicable. LAP is not used.

2.7. description of the LAP procedure
Not applicable. LAP is not used.

3. description of the packet level interface

This section is a subset of CCITT-X.25. The following facilities are *not* used:

- Diagnostic packets are not used. If a Diagnostic packet is received, it is ignored.

All other facilities conform to CCITT-X.25.

4. procedures for virtual circuit services

This section is a subset of CCITT-X.25. The following facilities are *not* used:

- The Permanent Virtual Circuit (PVC) procedure is not used. There are no PVCs. X.25 Section 4.2 is not applicable.

- The More Data mark is not used. All packets are transmitted with the M bit set to 0. X.25 Section 4.3.4 is not applicable.

- The Qualifier bit is not used. All packets are transmitted with the Q bit set to 0. X.25 Section 4.3.6 is not applicable.

- The Interrupt procedure is not used. Neither DTE nor DCE may transmit an Interrupt packet at any time. X.25 Section 4.3.7 is not applicable.

- The Reset procedure is not used. Neither DTE nor DCE may transmit a Reset packet at any time. If an error occurs, the station detecting the error must clear the virtual circuit by transmitting a Clear Request packet. X.25 Section 4.4.3 and parts of section 4.5 are not applicable.

All other facilities conform to CCITT-X.25.

5. packet formats

Same as CCITT-X.25.

There are many formats which are not applicable, since the facilities they support are not used.

Extended format Clear packets are *not* used. These packets do not contain addresses.

The following packet types do *not* contain cause or diagnostic code bytes. These packets end with the packet type identifier.

- Clear
- Clear Confirmation
- Restart
- Restart Confirmation

6. procedures for optional facilities

Not applicable. No optional facilities are used.

7. formats for facility and registration fields

Not applicable. No optional facilities are used.

8. service interface

This section specifies the service interface used by programs which communicate via asynchronous X.25. This is an interface between software modules within a DTE or DCE. CCITT-X.25 does not specify this interface.

The service interface is specified here in the form of Ada packages. Implementations in other languages are possible, but should provide the same semantics as the Ada implementation.

8.1. communication

For connecting, disconnecting, and data communication, the service interface is an extension of package Transport, as implemented on the R1000 as part of the Rational Networking TCP/IP product. For complete documentation of package Transport, see the Rational Networking manual.

8.1.1. protocol selection

The same package Transport is used for TCP/IP/Ethernet and asynchronous X.25 communication. The desired protocol is selected by the Network parameter to Transport.Open:

```
Transport.Open (..., Network => "TCP/IP"); -- TCP/IP/Ethernet
Transport.Open (..., Network => "AX25");   -- Asynchronous X.25
```

8.1.2. addressing

The addressing scheme does not support packet switching, only point-to-point links. A remote host is identified by the link to it, that is, the local RS-232 interface to which it is connected.

For AX25, a Transport_Defs.Host_Id may be any number of bytes long. The bytes are interpreted as a binary number. The first byte is the most significant. The number identifies an asynchronous port (RS-232 interface) on the local machine. This implicitly identifies the remote host which is connected to the specified port by an RS-232 line.

For AX25, a Transport_Defs.Socket_Id may be up to 6 bytes long. The bytes are interpreted as a binary number. The first byte is the most significant.

When an X.25 Call Request packet is transmitted, the Called DTE Address field identifies the socket (in the called machine) which is being called, and the Calling DTE Address field identifies the socket (in the calling machine) from which the call is initiated. Both fields are encoded in binary coded decimal, representing the socket number, that is, the bits of the Socket_Id interpreted as a binary number. For example:

Socket_Id	X.25 address
0.21	21
1.0	256
1.0.0	65536

The Host_Ids of the calling and called machines are not transmitted in the X.25 Call Request packet.

8.1.3. connecting

A task may wait for an incoming connection by calling Transport.Open and the passive form of Transport.Connect. The Local_Socket parameter of Transport.Open specifies what socket to wait on. Transport.Connect will return when a call is received for the socket that was specified in Transport.Open.

If there are several X.25 ports in the local machine, a passive connect waits on all of them, that is, the first call to arrive on any X.25 port (for the specified socket) will cause Transport.Connect to return.

A task may initiate a connection by calling Transport.Open and the active form of Transport.Connect. The Remote_Host parameter of Transport.Connect specifies the port via which to send the Call Request. The Remote_Socket parameter of Transport.Connect specifies the socket (in the remote machine) to which to connect.

8.1.4. exchanging data

Transport.Transmit and Transport.Receive are used to exchange data.

The user cannot control the setting of the Q and M bits in transmitted X.25 data packets. All data packets are transmitted with Q=0 and M=0.

8.1.5. disconnecting

Either end of a connection may disconnected at any time by calling Transport.Disconnect or Transport.Close. If Transport.Close is called, the connection will be broken immediately, and data that are in transit may be lost. If Transport.Disconnect is called, all data in transit will be delivered, and then the connection will be broken.

The X.25 D-bit procedure (CCITT-X.25 section 4.3.3) is used to ensure the delivery of data in transit at the time of a Transport.Disconnect. The last data packet transmitted before clearing will contain D=1, requesting end-to-end acknowledgement. This packet may or may not contain user data. The transmitting machine will wait to receive acknowledgement of this packet before transmitting a Clear Request packet. The receiving machine must deliver all data up to and including the data (if any) in the packet with D=1 before returning a status of Not_Connected from Transport.Receive.

8.2. configuration

Each machine must provide some means of designating which ports (RS-232 interfaces) are to be used for asynchronous X.25 communication, the protocol half which is to be used on each port, and the identity in the remote machine of each port. This information may be bound into the system software, or may be controlled at run time.

8.2.1. on the R1000

The R1000 implementation of X.25 will be fully configurable while the system is running. At least two configuration procedures will be provided:

Enable — enables X.25 communication on a port. Parameters include:

- the port number to enable.
- whether this machine is to act as DTE or DCE on the port. The X.25 protocol is asymmetric: on each link, one machine must act as DCE, and the other must act as DTE.
- the Transport_Defs.Host_Id which is used in the remote machine to identify the local port. This value will be returned to callers of Transport.Local_Host.

After this procedure is executed, the system begins transmitting LAPB DISC commands. When a DISC is acknowledged, a SABM is transmitted, and so on. This activity is carried out by system tasks, after the Enable procedure has returned.

Disable — disables X.25 communication on a port. There is only one parameter:

- the port number to disable.

The port becomes available for other uses, for example, logging in to the Rational Environment.

By default, all ports will be disabled. Rational system managers will usually set up their Machine.Initialize procedure to call the AX25 Enable procedure for each port which is used for AX25 communication.

I. ranges of logical channels

The range of logical channel numbers (LCNs) 0 through 255, inclusive, is used. The logical channel group number (LCGN) is always zero (0). This means that there may no more than 256 simultaneous connections over an AX25 link.

This protocol conforms to Annexes A through G of CCITT-X.25, where applicable. Many parts of the CCITT standard are not applicable, since they pertain to facilities which are not used.

II. revision history

The first version of this specification was dated 28 November 1986. Since that version, the following changes have been made:

- The network_name used to denote this protocol was changed from "AX.25" to "AX25" (section 8.1.1).
- Diagnostic packets were eliminated from the protocol (section 3).
- Cause and diagnostic code bytes were eliminated from Restart and Clear packets (section 5).
- Various areas of ambiguity were clarified.