Excerpts taken from:

Network Troubleshooting
By Othmar Kyas

An Agilent Technologies Publication

Section IV

Troubleshooting High-Layer Protocols

Chapter 17

Network Services and Applications

Network Services and Applications

17

"Become a student of change. It is the only thing that will remain constant."

ANTHONY J. D'ANGELO

17.1 E-Mail

E-mail systems, like most network services, are based on a client–server principle. Mail servers—properly known as Message Transfer Agents or MTAs—spool, manage, and transport the messages. E-mail clients, called Mail User Agents or MUAs, provide the interface to the user. MUAs exchange incoming and outgoing messages with the nearest MTA. A number of transport protocols are used for mail communication between e-mail servers and clients, and between one mail server and another. Usually MTAs transfer mail to one another using the classic Simple Mail Transfer Protocol (SMTP, RFC 821). This protocol has been in use since its adoption in 1982, and has gained widespread popularity due to its simplicity and robustness.

Naturally SMTP has limitations, and these have become more apparent as the use of the protocol has increased. For example, because SMTP was developed to transport text messages, it is unable to process any data format except 7-bit ASCII text. The growing demand not just for text messaging, but for asynchronous transfer of all kinds of data makes SMTP an unsatisfactory solution. Another problem is the increasing popularity of X.400 messaging systems. X.400 includes mechanisms to transport non-text messages, so it is difficult if not impossible to interconnect X.400 and SMTP networks. X.400 messages with non-text contents therefore have been rejected at SMTP/X.400 gateways.

In the early nineties it became clear that these major limitations of SMTP would have to be overcome. For this reason, an extension was adopted in 1993, called Multipurpose Internet Mail Extensions (MIME), RFC 1522. MIME defined extensions to the SMTP header—a "content type" header—that managed the problems mentioned previously without creating incompatibilities with existing RFC 821 SMTP implementations. MIME mechanisms are now widely used and have also been incorporated in the Hypertext Transfer Protocol (HTTP) on which the

WWW service is based. Furthermore, in the Secure MIME (S/MIME) specification two additional MIME content types were defined to support the transport of encrypted and digitally signed e-mail messages. With these extensions, SMTP forms the base for a powerful, universal messaging service.

Other protocols besides SMTP are used to transfer messages between mail servers and mail clients. The choice of protocol depends in part on the type of network connection between the client and the server. With dial-up network access (that is, a client dials up a connection to the server, exchanges mail messages, and clears down the connection) the most common protocol is currently Version 3 of the Post Office Protocol (POP3, RFC 1939). For on-line mail access, in which the mail client is connected with the server during the entire mail session and all messages are stored on the server's storage media, POP3 is not ideal and is increasingly being supplanted by the more modern Internet Message Access Protocol (IMAP-4), RFC 1730. As an alternative to dedicated mail protocols, LAN-based intranets often implement messaging services over LAN file system protocols—such as NFS (UNIX) or SMF (Novell) with the corresponding user interface programs (such as MAPI, VIM, AOCE), or with a WWW-based front end over the HTTP protocol (or S-HTTP or SSL). Mailbox information is usually managed and structured by means of the Lightweight Directory Access Protocol (LDAP), a simplified variant of the more complex X.500 protocol.

17.1.1 Simple Mail Transfer Protocol (SMTP, RFC 821)

As the name implies, SMTP is a simplified version of an earlier mail transport scheme, the Mail Transfer Protocol (MTP). As a matter of fact, the SMTP mechanism is surprisingly simple. Messages are transferred in three steps. The sender initiates the process by transmitting a MAIL command with the sender's address:

MAIL FROM:<othmar_kyas@agilent.com> <CRLF>

(where <CRLF> stands for a new line character: Carriage Return and Line Feed). This message informs the receiver that a mail transfer is ready to begin and the input buffer can be cleared. If the receiver is ready, it responds to the MAIL command with:

250 OK <CRLF>

In the second step, the sender provides the name of the intended recipient:

RCPT TO:<michael_meier@hp.com> <CRLF>

If the destination address is known, the receiving MTA acknowledges this command with:

220 OK <CRLF>

Otherwise, if the recipient is not known on the receiving server, it answers:

550 Failure reply <CRLF>

This RCPT TO exchange may be repeated several times if the message is to be delivered to more than one recipient. In the third step, the sender announces the actual message by means of the DATA command:

DATA <CRLF>

The response is:

354 Intermediate reply <CRLF>

after which all characters received are treated as the message (including both the header and the body of the message). A line containing only a period indicates the end of the data stream:

<CRLF>.<CRLF>

The receiving SMTP server acknowledges this by replying:

250 OK <CLRF>

Two further commands are used to open and close communication channels between two SMTP hosts:

HELO <domain name><CLRF>

and :

QUIT <CLRF>

The SMTP server responds to the HELO command with its own domain name:

250 <domain name> <CLRF>

The following example shows a complete SMTP session:

HELO mail.agilent.com
250 mail.agilent.com

MAIL FROM:<Smith@agilent.com>
250 OK

RCPT TO:<Jones@hp.com>
250 OK

RCPT TO:<Green@hp.com>
550 No such user here

RCPT TO:<Brown@hp.com>
250 OK

DATA
354 Start mail input; end with <CRLF>.<CRLF>

Test message – Test message – Test message. <CRLF>.<CRLF>
250 OK

QUIT
221 mail.agilent.com Service closing transmission channel

Figure 17.1 lists all the SMTP commands; Figure 17.2 the response codes.

HELLO (HELO)
Identification of the sender-SMTP to the receiver-SMTP

MAIL (MAIL)
Initiation of a mail transaction

RECIPIENT (RCPT)
Identification of the recipient of the mail data

DATA (DATA)
Following this command mail data is being transmitted

SEND (SEND)
Initiation of mail transaction to terminal

SEND OR MAIL (SOML)
Initiation of mail transaction to terminal or mailbox

SEND AND MAIL (SAML)
Initiation of mail transaction to terminal and mailbox

RESET (RSET)
Abortion of current mail transaction

VERIFY (VRFY)
This command asks the receiver to confirm that the argument identifies a user.

EXPAND (EXPN)
This command asks the receiver to confirm that the argument identifies a mailing list, and if so, to return the membership of that list.

HELP (HELP)
This command causes the receiver to send helpful information to the sender of the HELP command

NOOP (NOOP)
Receiver to reply with OK

QUIT (QUIT)
Receiver to reply with OK and to close the transmission channel.

TURN (TURN)
Receiver to either send an OK reply and then take on the role of the sender-SMTP, or to send a refusal reply and retain the role of the receiver-SMTP.

Figure 17.1 SMTP commands (RFC 821)

211	System status, or system help reply
214	Help message
220	<domain> Service ready
221	<domain> Service closing transmission channel
250	Requested mail action okay, completed
251	User not local; will forward to <forward-path>
354	Start mail input; end with <CRLF>.<CRLF>
421	<domain> Service not available, closing transmission channel
450	Requested mail action not taken: mailbox unavailable
451	Requested action aborted: local error in processing
452	Requested action not taken: insufficient system storage
500	Syntax error, command unrecognized
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
550	Requested action not taken: mailbox unavailable
551	User not local; please try <forward-path>
552	Requested mail action aborted: exceeded storage allocation
553	Requested action not taken: mailbox name not allowed
554	Transaction failed

Figure 17.2 SMTP response codes (RFC 821)

The most commonly used software for implementing an SMTP MTA is the UNIX application *sendmail*. This program runs on the mail server as a continually active SMTP daemon "smtpd", receiving and sending e-mail.

From the point of view of communication security, one drawback of SMTP is that the entire communication session takes place in plain 7-bit ASCII text. Another is that no mechanism is provided for verification of the sender's identification: thus any SMTP client can introduce itself with a fraudulent or non-existent address (MAIL FROM:<fake@nobrain.cracker.com>, for example). Reliable identification of e-mail messages is only provided by encryption and digital signature techniques, such as used in Privacy Enhanced Mail (PEM) or S-MIME.

17.1.2 Multipurpose Internet Mail Extensions (MIME, RFC 1521)

The MIME extensions overcome SMTP's restriction to plain 7-bit ASCII text by introducing special MIME headers to supplement the standard mail headers. In addition to the MIME version header, a "content-type" header field is defined with seven possible values:

- Text for text information in various character sets
- Multipart for combining several message parts, which may be of different content types, into a single message
- Application for any kind of binary data
- Message for encapsulating other mail messages
- Image for transmitting still pictures
- Audio for audio or voice data
- Video for video data

Another special header field, the Content-Transfer-Encoding field, can contain specialized information about the way in which the data has been encoded for SMTP transfer. Finally, the Content-ID and Content-Description header fields can be used to identify more specifically the contents of the message body.

The specific values that are defined for the Content-Type header field are listed in a global document that is maintained centrally by IANA, the Internet Assigned Numbers Authority (ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types). Every new type or subtype must be submitted through a defined registration procedure (RFC 2048) in order to be added to this worldwide register.

The MIME-Version Header Field

Every MIME-compatible message must contain a MIME-version header field that indicates the version of MIME implemented by the sender:

MIME-version: 1.0

The Content-Type Header Field

The Content-Type header field informs the receiving software of the type of information contained in the message so that it can be presented with an appropriate application, for example. The specified content type is divided into

a general type, such as text, image, or audio; and a subtype, such as plain or html in the case of text, or jpeg or gif for an image. The header line:

Content-Type: image/xyz

allows a MIME-compatible e-mail client to determine that the data received represents an image file, whether or not it recognizes the "xyz" format. This information can be used to decide whether it is worthwhile to display the raw data on the screen even if the data format is not supported. For example, it may be a good idea to display unknown subtypes of "text", but not of the "image" or "audio" content types. Private content subtypes not registered with IANA may be used provided they begin with "x-", as in "Content-Type: application/x-pkcs10".

Content-Type: Text

The text content type is used to indicate text that can be displayed without specialized software. Certain formatted text documents may be included for which special application software may offer an improved display, but is not strictly necessary in order for the contents to be understood.

Content-Type: Multipart

Multipart messages consist of several concatenated message body parts. A mandatory parameter in the Content-Type: Multipart header field defines the boundary marker that separates the individual parts. The parts may be of the same type or of different types. The following four subtypes are defined for the multipart type:

- Mixed — for message body parts of different types
- Alternative — for alternative representations of the same content (such as plain and HTML-formatted text, for example)
- Parallel — for message body parts to be presented simultaneously
- Digest — for multipart messages in which each part consists of a mail message

Content-Type: Message

The "message" content type is itself an e-mail message. The subtype "partial" can be used to permit fragmented encapsulation of messages in order to overcome the length restrictions of certain gateways.

Content-Type: Image

Image data require special software and/or hardware for presentation (graphic display adapter, printer, fax machine). Originally, two subtypes were defined:

jpeg and gif. Now many other popular image formats have been added including x-windowdump, tiff, x-rgb, etc.

Content-Type: Audio

Data of the content type "audio" (subtypes: x-aiff, x-wav, etc.) require an audio player device for presentation, such as a loudspeaker or telephone.

Content-Type: Video

Video messages also require special software and/or hardware for presentation of the moving-picture content. Common subtypes include quicktime, mpeg, and x-msvideo.

Content-Type: Application

The application content type includes all kinds of binary data that are not covered by any of the content types listed here. If the data cannot be described in more detail, the primary subtype "octet-stream" is used. Today a great number of subtypes are defined, including msword, postscript, zip, and tar.

If no content type is indicated, the default type:

Content-Type: text/plain; charset=us-ascii

is assumed.

Encoding of Data Other Than 7-Bit ASCII

Because the SMTP specification (RFC 821) is limited to 7-bit US-ASCII messages with a maximum line length of 1,000 characters, MIME defines not only other data types but also an encoding scheme to convert all kinds of binary data into an SMTP-compatible format. The data to be encoded can be roughly divided into two kinds:

- Text-based data that does not fall within the 7-bit US-ASCII character set
- Binary data

Two encoding mechanisms have been chosen: quoted-printable and base64. The type of encoding used in a given message is indicated in the Content-Transfer-Encoding header field:

Content-Transfer-Encoding:	quoted-printable
	base64
	8bit
	7bit
	binary

The values *8bit*, *7bit,* and *binary* indicate that no encoding was performed on the data. At the present time, 8-bit and binary data cannot be sent over SMTP systems without other encoding. These field values have nonetheless been defined for use by future mail systems.

17.1.3 S/MIME – Secure Multipurpose Internet Mail Extensions

An extension to the MIME specification, S/MIME defines two new content types designed to allow the use of digital signatures and encryption mechanisms in MIME messages. The cryptography technique is described in PKCS #7. The contents of MIME-compatible messages can be digitally signed, encrypted, or both. S/MIME is undergoing development by an IETF working group; PKCS #7 is a specification of RSA Security, Inc.

17.1.4 Privacy Enhanced Mail (PEM)

PEM is a proposed Internet standard for the encryption of SMTP messages (RFCs 1421 through 1424). Both RSA (public key encryption implementation by Rivest, Shamir, Adleman) public key encryption and symmetrical Data Encryption Standard (DES) are specified as encryption techniques. Data is first encrypted using the DES algorithm and a randomly generated DES key. The key is then encrypted using the RSA algorithm and the addressee's public key, and sent together with the DES-encrypted data. The advantage of this method is that only a small part of the message, namely the DES key, needs to be encrypted using the CPU-intensive RSA algorithm. The DES algorithm used to encrypt the actual message contents is quicker. Before the encrypted data can be sent by SMTP, it must be converted to 7-bit ASCII by a MIME content transfer encoding method.

17.1.5 E-Mail Standards

RFC 821	Simple Mail Transfer Protocol
RFC 822	Internet Mail Header Format
RFC 1123	Internet Host Requirements
RFC 1869	SMTP Service Extensions
RFC 1891	SMTP Delivery Status Notifications
RFC 1892	Multipart/Report
RFC 1893	Mail System Status Codes
RFC 1894	Delivery Status Notification

RFC 1985	SMTP Service Extension for Remote Message Queue Starting
RFC 2034	SMTP Service Extension for Returning Enhanced Error Codes
RFC 2045	MIME
RFC 2476	Message Submission
RFC 2554	SMTP Service Extension for Authentication
RFC 1421 – 1424	Privacy Enhanced Mail (PEM)
RFC 1939	Post Office Protocol 3 (POP3)
RFC 1730	Internet Message Access Protocol (IMAP-4)
RFC 1777	Lightweight Directory Access Protocol (LDAP)

17.1.6 Troubleshooting E-Mail

The first step in diagnosing e-mail problems is to test the connection to the mail server. The best way to do this is to establish a telnet connection to port 25 on the mail server (a ping may be used to test the host's IP address, but cannot determine whether an MTA is active on port 25):

telnet 15.70.23.112 25

Once the connection to port 25 has been successfully established, a SMTP session can be performed:

SMTP Commands	Expected Response	Description
HELO	250 OK	Initiates the conversation
MAIL FROM: <*user@domain.tld*>	250 OK	Identifies the sender
RCPT TO: <*user@domain.tld*>	250 OK	Identifies the recipient of the mail message
DATA	354 Send Data	Announces message data
[a single period]	250 OK	Marks the end of the message data
QUIT	221	Closes the session

Telnet can also be used in the same way to test the POP3 and IMAP protocols by connecting to the corresponding port addresses:

Protocol	Port
SMTP	25
HTTP	80
NNTP	119
LDAP	389
POP3	110
IMAP4	143

POP3 Commands	Expected Response	Description
USER *Account name*	+OK	Initiates the authentication process
PASS *Password*	+OK	Specifies the password for the POP3 account
LIST	+OK	Lists the messages in the user's mailbox
RETR *Message No.*	*The message text*	Retrieves the text of the specified message
DELE *Message No.*	+OK	Deletes the specified message
QUIT	+OK	Closes the session

All commands issued to an IMAP4 server must be prefixed with a command identifier. This identifier can be used by the client to keep track of command and response pairs. For example, when the IMAP client sends:

```
local SELECT inbox
```

the server responds with:

```
local OK
```

IMAP4 Commands	Expected Response	Description
LOGIN *Accountname Password*	OK LOGIN	Logs on to the mailbox
SELECT *folder*	*Folder mode* & OK SELECT	Selects a folder to view
FETCH *Message No.*	*Message text* & OK FETCH	Retrieves the specified message
STORE *Message Flags* *flag*	OK STORE	Marks a message for deletion or as read/ unread
EXPUNGE	OK	Deletes all marked messages
LOGOUT	OK	Closes the session

Problems can usually be quickly traced by recording and decoding these communication processes with a protocol analyzer. The causes are usually to be found at the TCP/IP level in the configuration or availability of the server, or in the client configuration.

17.2 Internet News

The Internet service "news" consists of a multitude of discussion forums, comparable to mailing lists, which are hierarchically organized by topic. The distribution and function of news differs significantly from e-mail and mailing lists. Internet news articles, posted by users, are not automatically delivered to other users, but are stored on central news servers. To read news, users must have access to such a server. The format in which the articles are stored is specially adapted to the requirements of discussion forums, and for a long time required the use of special presentation programs called news readers. Today, complete news reader capabilities are standard equipment in Web browsers. News servers exchange new articles posted by users at regular intervals. This exchange takes place using the Network News Transfer Protocol (NNTP), with the reserved TCP port 119. A few sites, such as former Usenet nodes, still use the UUCP protocol to exchange news.

17.2.1 The Network News Transfer Protocol (NNTP, RFC 977)

The NNTP protocol defines the organization of the news service, both for server–server and for client–server communication. Client–server news communication involves the following functions:

- Retrieval: sending requested news articles to clients
- Posting: accepting new articles from clients

Furthermore, NNTP also governs replication, that is, the distribution of new articles to all other servers, and provides a reliable, interactive communication mechanism for this purpose. Other server tasks are indexing and linking related articles, monitoring their age, and deleting them when obsolete.

Every NNTP session includes the following seven steps:

- CONNECTION
- GREETING
- CAPABILITIES DISCOVERY
- AUTHENTICATION
- NEWS TRANSFER
- CONCLUSION
- DISCONNECTION

The NNTP server listens on TCP port 119 and responds to client requests with the following three-digit response codes. The first digit indicates whether the operation was successful or not, or whether the preceding command was carried out:

1xx Informative message

2xx Command OK

3xx Command OK, awaiting further commands

4xx Command was correct but could not be executed

5xx Command incorrect, unavailable, or not implemented

The second digit of the response code indicates which function the code refers to:

x0x Connection, setup, and miscellaneous messages

x1x Newsgroup selection

x2x Article selection

x3x Distribution functions

x4x Posting

x8x Nonstandard (private implementation) extensions

Greeting

On requesting the connection to the server's TCP port 119, the client does not need to send any command, but simply waits for the server's first response code, the greeting. This code indicates how the client may proceed. The NNTP server greets the client with code 200 if the client is authorized to post new articles using the POST command, or with 201 if posting is not allowed. The response code 205 indicates that further authentication is required before any other action, and code 502 (service unavailable) indicates that the client is not authorized to interact with the server at all. In all other cases, the server responds with code 400 (service temporarily unavailable). Figure 17.3 lists the NNTP greetings:

200	Hello, you can post
201	Hello, you can't post
205	Authentication required
400	Service temporarily unavailable
502	Service unavailable

Figure 17.3 Response codes during the NNTP greeting phase

Optionally, the client may use the command MODE READER to inform the server that it only wants access to read existing news. The server again responds with the appropriate code. During the greeting phase the client can also issue the command LIST EXTENSIONS. This is a request to the server to list its implemented extensions to the NNTP protocol. If no extensions are supported, the server returns code 503 (program error, function not performed). Otherwise it can respond with 205 (authentication required) and a list of extensions consisting of keywords and the associated parameters.

Authentication

If the server has requested authentication from the client, the client must answer with the command AUTHINFO SIMPLE. If the server accepts this, it responds with code 350 (continue with authorization sequence). The client must send its user name and password, to which the server responds with code 250 (authorization accepted). Then the client can continue with whatever command was interrupted by the authentication request (see Figure 17.4).

250	Authorization accepted
350	Continue with authorization sequence
450	Authorization required for this command
452	Authorization rejected
501	Command not supported or Command syntax error
502	Program error, function not performed

Figure 17.4 Response codes during the authentication phase

Retrieval of News Articles

The news articles stored on news servers can be identified by two parameters: the message ID and the article number. The message ID is an identification string that is unique in the entire network. The article number is formed locally on the news server from the newsgroup name and the number of the article within the newsgroup. The article number is unique on the local news server, but need not be unique in the network. Because one message may be posted to several newsgroups, it is possible for several article numbers from different newsgroups to point to the same article. In order to retrieve an article, the client must first set the "article pointer" to the desired article. Then the pointer can be used to retrieve the article.

17.2.2 NNTP Standards

RFC 977 NNTP protocol
RFC 1036 News article format

17.2.3 Troubleshooting News

The first step in diagnosing problems with the news service is to test the TCP connection to the news server. The easiest way to do so is by using Telnet to connect to port 119 of the corresponding host:

```
telnet 15.70.23.112 119
```

If the news server responds on port 119, a NNTP session can be carried out step by step. There are two modes of NNTP access: authenticated and anonymous. Authentication requires the first two commands listed below:

NNTP Commands	Expected Response	Description
AUTHINFO USER *Username*	381 More Info Needed	Supplies authentication information
AUTHINFO PASS *Password*	281 Accepted	Supplies password for authentication
LIST	*List of groups*	Lists all available groups
GROUP *Group*	*Group specifications*	Sets the current group
ARTICLE *Article No.*	*Article text*	Retrieves the specified article in the current group
QUIT	205	Closes the session

17.3 The File Transfer Protocol (FTP)

FTP is one of the oldest network service protocols. It is used to transfer data of any kind—text, image, sound, video, executable programs, etc. The FTP client, the system initiating a connection, controls the complete procedure. The client must have access authorization to the FTP server, however, which is established by checking user ID and password information during the FTP connection setup. FTP is based on the TCP transport protocol and uses two simultaneous TCP connections to transfer files (see Figure 17.5). The client begins by establishing a control connection from any TCP port to the server's port 21. The actual file transfer takes place over a second connection initiated by the server. This data connection is opened from the server's TCP port 20, the standard FTP data port, to a port specified by the client using the PORT command. The data requested by the client flows over this connection in the format used by the Telnet protocol, described in the Network Virtual Terminal (NVT) specification.

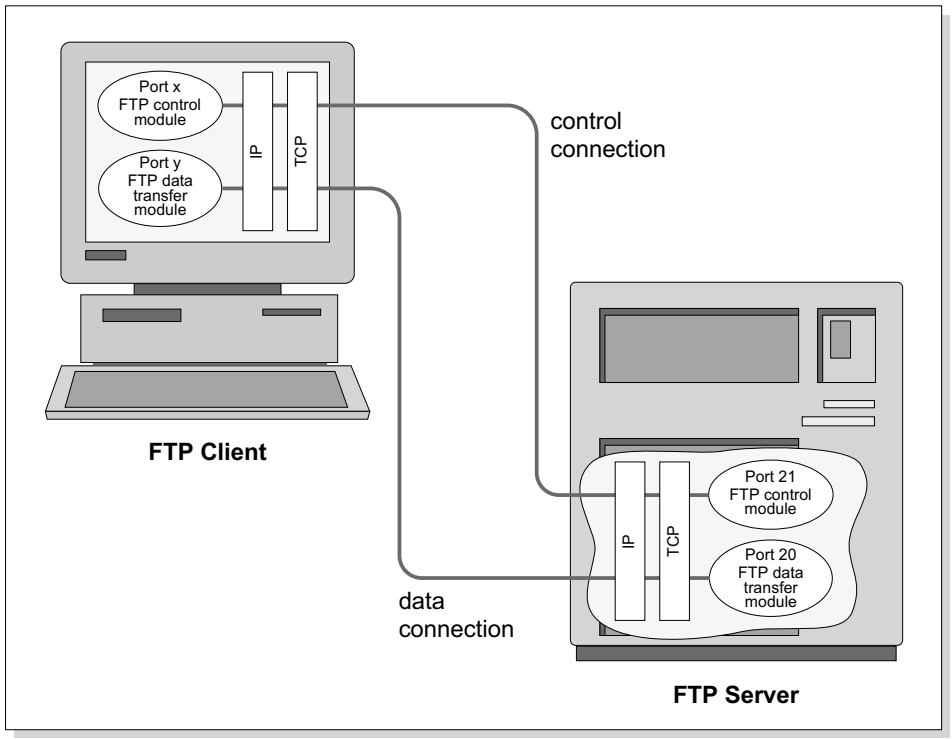

Figure 17.5 The FTP protocol: control and data connections

For each file transfer another data connection is opened. Due to a peculiarity of the TCP protocol, a different client port address is used each time.

dir / ls	display directory / display listing
cd / pwd	change directory / print working directory
bin / ascii	binary / ascii transfer mode
hash	display transfer process using the # sign
get <file>	retrieve file

Figure 17.6 The most important FTP commands

The IETF "FTPEXT" working group is currently coordinating proposed extensions to FTP.

17.3.1 FTP Standards

RFC 959 File Transfer Protocol (FTP). The basic FTP definition (from 1985). All FTP software must comply with this document. Other requests for comments (RFCs) may extend or clarify this document.

RFC 1123 Requirements for Internet Hosts—Application and Support. This RFC defines and discusses the requirements for Internet host software. It covers the application and support protocols; its companion, RFC-1122, covers the communication protocol layers: link layer, IP layer, and transport layer.

RFC 1639 FTP Operation Over Big Address Records (FOOBAR). Description of the convention for specifying address families other than the default Internet address family in FTP commands and replies.

RFC 2228 FTP Security Extensions

RFC 2389 Feature Negotiation Mechanism for the File Transfer Protocol. Amendment of the File Transfer Protocol (FTP). Two new commands (FEAT, OPTS).

RFC 2428 FTP Extensions for IPv6 and NATs

RFC 2577 FTP Security Considerations. Provides a proxy mechanism to decrease traffic load on the network.

RFC 2640 Internationalization of the File Transfer Protocol. Support for multiple character sets.

17.3.2 Troubleshooting FTP

The most frequent sources of problems with FTP are user errors from logging in and connection timeouts. Typical login errors are incorrect (or mistyped) user names, IP addresses or passwords, non-existent or deleted accounts, and missing or modified access privileges. After a successful login, connection timeouts can occur when TCP idle timers expire due to system overload (on the server or client end) or high network loads. Large server directories with thousands of entries may lead to timeouts after a LIST command. The most common symptoms and their causes are listed in Section 17.11.

17.4 Trivial File Transfer Protocol (TFTP)

TFTP is a simplified version of FTP with no authentication mechanisms. It is used for diskless workstations, local software updates, or applications that do not require the full capabilities offered by FTP. Because TFTP does not claim to

offer reliable service, it uses UDP as the transport protocol rather than TCP. The resulting program code for TFTP implementations is thus much smaller than for FTP.

17.5 Telnet

Telnet is an application that originated in the UNIX world to allow interactive use of remote computers across data networks: the Telnet client application on the local computer functions as a remote terminal for the server host. Its remote control capabilities are limited to programs with a text-based user interface, however, because the Telnet protocol only transmits ASCII characters. In the Internet this restriction is usually unimportant because the available bandwidth is generally not sufficient for graphical user interface based remote control (such as X Window sessions). Telnet programs are available for practically all computer platforms and operating systems, and are often incorporated in the capabilities of Web browsers.

When the Telnet application has been started and the Internet address of the desired remote system entered, the Telnet program begins a TCP connection to the remote host and invokes its login routine. When the user name and password have been approved, the Telnet session is open. (The remote system that provides a service, such as remote access, is generally called the server; the computer

```
telnet> ?
Commands may be abbreviated.  Commands are:

close       close current connection
display     display operating parameters
mode        try to enter line-by-line or character-at-a-time mode
open        connect to a site
quit        exit telnet
send        transmit special characters ('send ?' for more)
set         set operating parameters ('set ?' for more)
status      print status information
toggle      toggle operating parameters ('toggle ?' for more)
z           suspend telnet
?           print help information
```

Figure 17.7 Telnet commands

that requests the service is the client or user). From this point on, all keyboard input is transmitted to the server, and all the server's output is transmitted to the client. Because all Telnet applications use a standardized interface to the network, called the Network Virtual Terminal (NVT), two systems can still communicate via Telnet even if their internal architectures use completely different data formats. The NVT data format is based on the standard 7-bit US ASCII character set in an 8-bit field, with the higher 8-bit values used as control codes. Thus the entire Telnet session consists of plain text transmission and can be easily recorded by third parties with access to the intermediate systems.

17.5.1 Troubleshooting Telnet

As with FTP, the most common causes of trouble with Telnet are user errors in logging in and connection timeouts. Here once again, entering incorrect user names, IP addresses, or passwords (due to mistyping or to an unfamiliar keyboard layout), nonexistent or deleted accounts, and missing or modified access privileges are typical causes of login errors. Connection timeouts can occur due to system overload (on the server or client end) or high network loads. The most common symptoms and their causes are listed in Section 17.11.

17.6 Hypertext Transfer Protocol (HTTP)

HTTP is the transport protocol on which the Internet service World Wide Web is based. The original 1990 version 0.9 of HTTP only specified the transfer of data identified by Uniform Resource Identifiers (URIs), which combine a Uniform Resource Locator (URL) and a resource name. In HTTP 1.0 (RFC 1945), these capabilities were extended through the definition of MIME elements that could be included in the protocol header to convey information about the type of information transferred. Nonetheless, HTTP 1.0 soon showed limitations in the face of rapidly growing demands in the Internet and in intranets. Most of all, the effects of cascaded proxies (necessitated by firewall schemes) and caching mechanisms led to long response times and high network traffic with a relatively low proportion of user data. Finally, however, the HTTP 1.1 standard that has been adopted represents a significant improvement in protocol performance.

17.6.1 The Basic Mechanisms of HTTP

The HTTP protocol is s single-state, send-and-receive protocol. The HTTP client sends the HTTP server a request containing the following elements:

- Request method
- URI
- Protocol version
- MIME information elements
- Information about the HTTP client
- Message contents (optional)

The server responds with a message containing the following:

- Status information
- Protocol version
- Success/failure rate
- MIME information elements
- Entity information (Meta-information about the content to be transferred)
- The message itself

In the simplest case, the client–server communication consists of a single connection between client and server (see Figure 17.8).

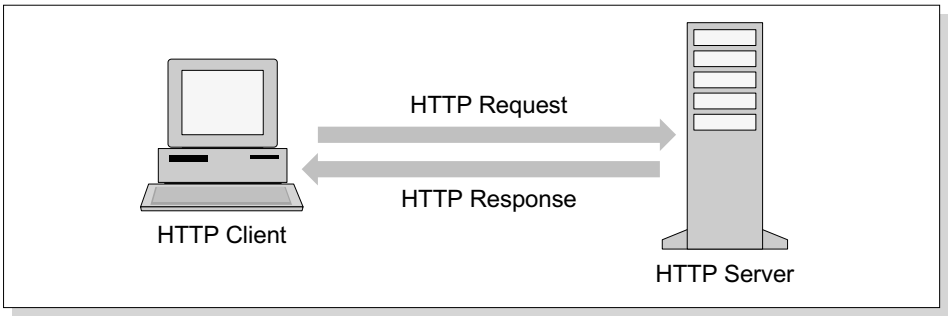

Figure 17.8 The principle of a direct HTTP connection

The situation is somewhat more complex if the communication path is not direct, but leads through several stations. Three kinds of indirect HTTP communication can be distinguished:

- Proxy connections
- Gateway connections
- Tunneling connections

Proxy connections are used mainly in connections with firewalls. Proxy servers HTTP requests from certain clients—usually those in a protected LAN—and set up the connection to the entity addressed by the URL in the client's request, then forward the resulting information received from the actual HTTP server to the requesting client. The purpose of the proxy server is to avoid a through connection from the client to the actual server. The advantage is that the client

does not need a public IP address and can be protected against a number of potential attacks from the Internet.

Gateways are systems that can interpret between two different protocols. HTTP messages can be received by gateways and translated into the desired destination protocol.

Tunneling refers to a transport-layer capability that does not affect the contents of the data transported. HTTP messages can be transported between the HTTP client and server by any tunneling transport protocol (Frame Relay, PPP, SLIP, etc.). Figure 17.9 illustrates an indirect connection across three intermediate stations.

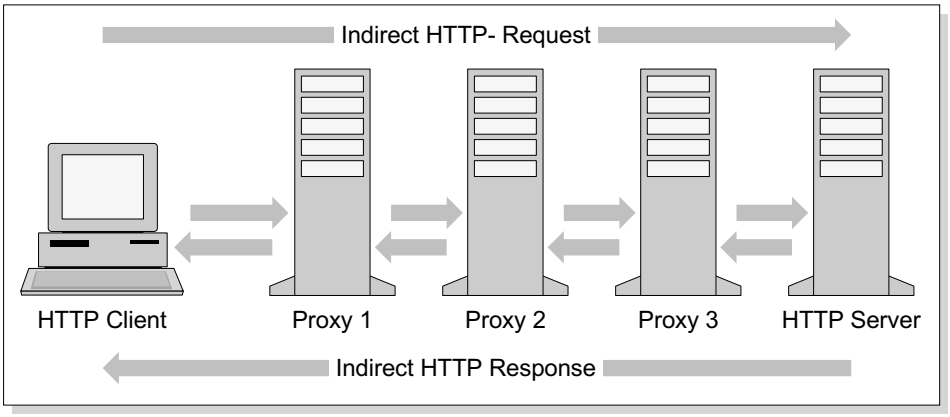

Figure 17.9 Indirect HTTP connections

In this illustration every HTTP request and every HTTP response must travel across the four individual links shown in order to reach their destinations. This is important to remember because certain HTTP options affect only the nearest non-tunneling neighbor, the endpoints of the overall connection, or all links along the connection path. Remember, furthermore, that every node along the path is generally involved in a number of connections simultaneously.

Every communication node that is not a tunnel can also maintain an internal cache from which it can answer requests. The purpose of a cache is to shorten the response times in the case where a requested document has been transferred once before. Besides reducing response time, the caching principle is also important for reducing the total data traffic in the network. For this reason, HTTP 1.1 incorporates special management functions to optimize the use of caches (see Figure 17.10).

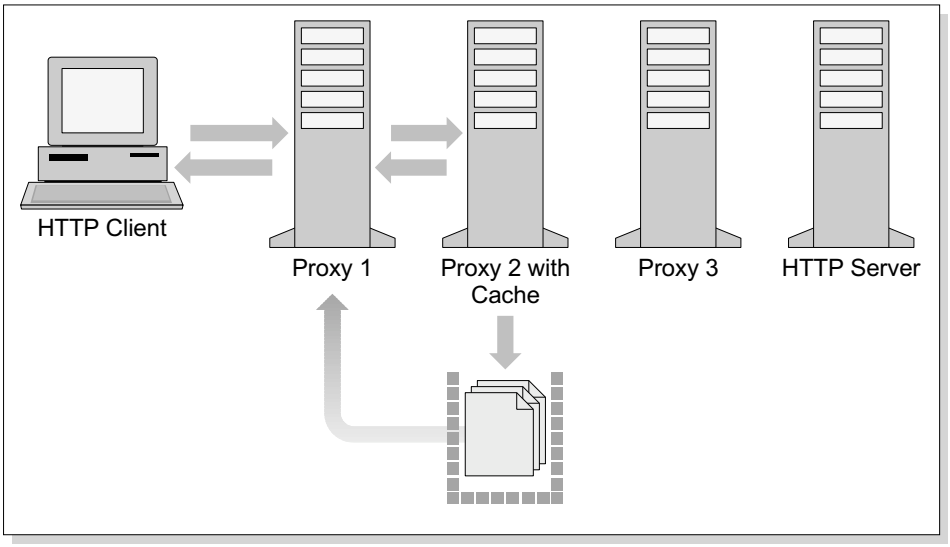

Figure 17.10 HTTP connection using cache

17.6.2 Differences Between HTTP 1.1 and HTTP 1.0

The main difference between the HTTP versions 1.0 and 1.1 is that HTTP 1.0 requires a separate TCP/IP connection to the HTTP server for every request by the client. Once the server has answered the request, it clears down the TCP/IP connection again. Because only one URI element is transferred for each request, five, ten or more TCP connections may have to be set up and cleared down to transfer a single Web page, which can consist of a great many URIs. Because setting up a TCP connection involves a three-step handshake procedure (>request, <response, >response-acknowledge), the network overhead traffic generated is considerable. Furthermore, the server must save information about the connection for a certain period after it has been cleared down (in the TIME-WAIT state, with a default timer value of 240 seconds!) in case a late packet arrives for the connection. This could lead to thousands of connection control blocks being maintained on the server.

The window size for TCP also reduces the performance of HTTP 1.0 over TCP/IP. Under HTTP 1.0 the many TCP connections are generally too brief to attain large window sizes so that the proportion of acknowledgment packets remains extraordinarily high and the thoughput is low.

HTTP 1.1 makes it possible to send several HTTP requests over the connection and to maintain the connection while waiting for the responses. Furthermore, HTTP requests can be “pipelined”. This means that a client can send several

HTTP requests in immediate succession, rather than waiting for the response to one request before sending the next. In conjunction with optimized support for cache mechanisms, these TCP features yield substantially improved performance. Tests in the laboratories of the World Wide Web Consortium (W3C) showed significant reductions in response time and network loads.

17.6.3 HTTP Messages

HTTP defines two types of messages:

- HTTP requests
- HTTP responses

The header of an HTTP data packet consists of General-Header, Request-Header, Response-Header, and Entity-Header fields. Every header field consists of the field name followed by a colon and the field value, if any. The actual data payload is called the message body, and its length is indicated in the Content-Length header field.

HTTP Requests

Every request from a client to a server begins with the Request-Line, which contains information about the method to be performed on the requested object, the URI that identifies the object on which to perform the method, and the protocol version used. Example:

```
GET http://www.lycos.com/graphics/backdrop.gif HTTP/1.1
```

This request asks the server to perform the method "GET" on the file identified by "http://www.lycos.com/graphics/backdrop.gif". In addition to "GET", the following methods are defined:

- HEAD
- POST
- PUT
- DELETE
- TRACE
- OPTIONS

The methods GET and HEAD must be supported by every server; all other methods are optional. The URI—"http://www.lycos.com/graphics/backdrop.gif", in our example—can take one of three forms. It may be an asterisk (*), indicating that the request applies to the server in general and not to a particular resource. This may be the case, for example, when a client requests the options supported by the server software:

```
OPTIONS * HTTP/1.1
```

If the request is being sent to a proxy, then the request URI must take the form of an absolute URI with scheme, colon, double slash, host, and absolute path. This is because the proxy needs to know what host to connect to before it can send the message to an actual HTTP server. If the request is being sent not to a proxy or gateway, but to the actual server on which the resource is located—that is, if the destination of the request message is the next network node in the HTTP connection chain—then a relative URI is sufficient, consisting of just the path name of the resource. (In any case, the host addressed is indicated in the second line of the request.) When an HTTP request with an absolute URI is sent through a number of cascaded proxies, the last proxy sends only the path name in its request to the server's HTTP port:

GET /graphics/backdrop.gif HTTP/1.1

Host: www.lycos.com

17.6.4 Connection Types

One of the most important innovations in HTTP 1.1 is its use of persistent TCP connections, which are maintained while several HTTP messages are exchanged between server and client. Under HTTP 1.0, a separate TCP connection had to be set up and cleared down for every URI transferred, so that a number of connections were required for one Web page containing in-line images, for example. Every HTTP 1.1 server assumes that an HTTP 1.1 client will attempt to carry out its communication over persistent TCP connections, unless the client request header contains the token "close" in a connection header.

Pipelining

Clients that support persistent TCP connections can also send several requests in immediate succession over an existing TCP connection without waiting for the server's responses. This mode is called pipelining and is a significant factor in the improved performance of 1.1 implementations over HTTP 1.0. It is particularly important, however, that the principles of persistent connections and pipelining be correctly implemented in proxies. The TCP timeout must be set higher for persistent TCP connections on proxy servers than on HTTP servers because it must be assumed that the client uses the proxy as a communication link for a longer period. Furthermore, when persistent connections are used the client should not open more than two connections simultaneously to prevent excessive loads on the network. (HTTP 1.0 clients used the technique of opening up to eight connections in order to accelerate the transfer of Web pages consisting of multiple URIs. This method is unnecessary due to the pipelining capabilities of HTTP 1.1.)

17.7 The Server Message Block (SMB) Protocol

The Server Message Block Protocol (formerly an Open Group standard "Protocols for X/Open PC Interworking: SMB, Version 2", since withdrawn; see *http://www.opengroup.org/*) provides printing, redirection and server functions to the application layer in OS2, Windows for Workgroups, and Windows 95/98/NT/2000 environments. The redirection function converts local application commands into SMB commands to the appropriate server. SMB is also supported by some UNIX systems. The most popular implementation of SMB for UNIX is Samba. Microsoft is currently trying to establish SMB as an Internet standard under the name CIFS ("Common Internet File System"; see *http://www.cifs.com/*). The SMB packet format is illustrated in Figure 17.11.

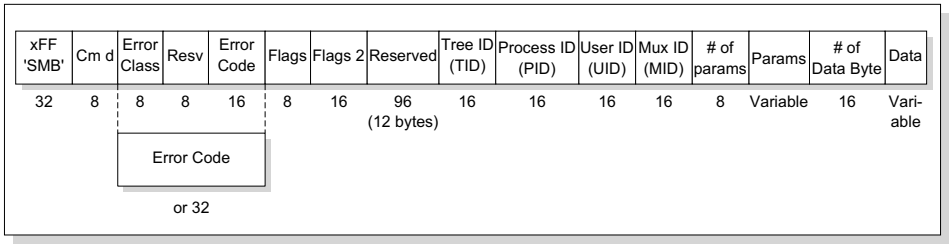

Figure 17.11 The SMB packet format

The protocol process for logging into a Windows NT server is as follows:

a) Request an IP address by DHCP.

b) Register the NetBIOS name with the WINS server (if present).

c) Broadcast SMB NETLOGON Requests to obtain the server name.

d) Request the server address from the WINS server, or by broadcast if no WINS server is present.

e) Start a TCP connection to the server, then a NetBIOS session.

f) Open an SMB session.

Figure 17.12 lists the SMB commands in the latest version of SMB (CIFS). Figures 17.13 through 17.16 list the error classes and error codes. ERRDOS messages generally refer to problems on the client; ERRSRV to server configuration problems; and ERRHRD to server hardware problems:

SMB_COM_CREATE_DIRECTORY	0x00	SMB_COM_IOCTL_SECONDARY	0x28
SMB_COM_DELETE_DIRECTORY	0x01	SMB_COM_COPY	0x29
SMB_COM_OPEN	0x02	SMB_COM_MOVE	0x2A
SMB_COM_CREATE	0x03	SMB_COM_ECHO	0x2B
SMB_COM_CLOSE	0x04	SMB_COM_WRITE_AND_CLOSE	0x2C
SMB_COM_FLUSH	0x05	SMB_COM_OPEN_ANDX	0x2D
SMB_COM_DELETE	0x06	SMB_COM_READ_ANDX	0x2E
SMB_COM_RENAME	0x07	SMB_COM_WRITE_ANDX	0x2F
SMB_COM_QUERY_INFORMATION	0x08	SMB_COM_CLOSE_AND_TREE_DISC	0x31
SMB_COM_SET_INFORMATION	0x09	SMB_COM_TRANSACTION2	0x32
SMB_COM_READ	0x0A	SMB_COM_TRANSACTION2_SECONDARY	0x33
SMB_COM_WRITE	0x0B	SMB_COM_FIND_CLOSE2	0x34
SMB_COM_LOCK_BYTE_RANGE	0x0C	SMB_COM_FIND_NOTIFY_CLOSE	0x35
SMB_COM_UNLOCK_BYTE_RANGE	0x0D	SMB_COM_TREE_CONNECT	0x70
SMB_COM_CREATE_TEMPORARY	0x0E	SMB_COM_TREE_DISCONNECT	0x71
SMB_COM_CREATE_NEW	0x0F	SMB_COM_SESSION_SETUP_ANDX	0x73
SMB_COM_CHECK_DIRECTORY	0x10	SMB_COM_LOGOFF_ANDX	0x74
SMB_COM_PROCESS_EXIT	0x11	SMB_COM_TREE_CONNECT_ANDX	0x75
SMB_COM_SEEK	0x12	SMB_COM_QUERY_INFORMATION_DISK	0x80
SMB_COM_LOCK_AND_READ	0x13	SMB_COM_SEARCH	0x81
SMB_COM_WRITE_AND_UNLOCK	0x14	SMB_COM_FIND	0x82
SMB_COM_READ_RAW	0x1A	SMB_COM_FIND_UNIQUE	0x83
SMB_COM_READ_MPX	0x1B	SMB_COM_NT_TRANSACT	0xA0
SMB_COM_READ_MPX_SECONDARY	0x1C	SMB_COM_NT_TRANSACT_SECONDARY	0xA1
SMB_COM_WRITE_RAW	0x1D	SMB_COM_NT_CREATE_ANDX	0xA2
SMB_COM_WRITE_MPX	0x1E	SMB_COM_NT_CANCEL	0xA4
SMB_COM_WRITE_COMPLETE	0x20	SMB_COM_OPEN_PRINT_FILE	0xC0
SMB_COM_SET_INFORMATION2	0x22	SMB_COM_WRITE_PRINT_FILE	0xC1
SMB_COM_QUERY_INFORMATION2	0x23	SMB_COM_CLOSE_PRINT_FILE	0xC2
SMB_COM_LOCKING_ANDX	0x24	SMB_COM_GET_PRINT_QUEUE	0xC3
SMB_COM_TRANSACTION	0x25	SMB_COM_READ_BULK	0xD8
SMB_COM_TRANSACTION_SECONDARY	0x26	SMB_COM_WRITE_BULK	0xD9
SMB_COM_IOCTL	0x27	SMB_COM_WRITE_BULK_DATA	0xDA

Figure 17.12 SMB commands

Class	Code	Comment
SUCCESS	0x00	The request was successful
ERRDOS	0x01	Error is from the core DOS operating system set
ERRSRV	0x02	Error is generated by the server network file manager
ERRHRD	0x03	Error is a hardware error
ERRCMD	0xFF	Command was not in the "SMB" format

Figure 17.13 SMB error codes and classes

Error	Code	Description
ERRbadfunc	1	Invalid function The server did not recognize or could not perform a system call generated by the server, for example, set the DIRECTORY attribute on a data file, invalid seek mode
ERRbadfile	2	File not found The last component of a file's pathname could not be found
ERRbadpath	3	Directory invalid A directory component in a pathname could not be found
ERRnofids	4	Too many open files The server has no file handles available
ERRnoaccess	5	Access denied The client's context does not permit the requested function. This includes the following conditions: • invalid rename command • write to FID (Format Identifier) open for read only • read on FID (Format Identifier) open for write only • attempt to delete a non-empty directory
ERRbadfid	6	Invalid file handle The file handle specified was not recognized by the server
ERRbadmcb	7	Memory control blocks destroyed
ERRnomem	8	Insufficient server memory to perform the requested function
ERRbadmem	9	Invalid memory block address
ERRbadenv	10	Invalid environment
ERRbadformat	11	Invalid format
ERRbadaccess	12	Invalid open mode
ERRbaddata	13	Invalid data (generated only by IOCTL calls in the server)
ERRbaddrive	15	Invalid drive specified
ERRremcd	16	A Delete Directory request attempted to remove the server's current directory
ERRdiffdevice	17	Not same device (for example, a cross volume rename was attempted)
ERRnofiles	18	A File Search command can find no more files matching the specified criteria
ERRbadshare	32	The sharing mode specified for an Open conflicts with existing FIDs on the file
ERRlock	33	A Lock request conflicted with an existing lock or specified an invalid mode, or an Unlock request attempted to remove a lock held by another process
ERRfilexists	80	The file named in the request already exists

Figure 17.14 SMB ERRDOS return codes

Error	Code	Description
ERRerror	1	Non-specific error code Returned under the following conditions: • Resource other than disk space exhausted • First SMB command was not negotiated • Multiple negotiations attempted • Internal server error
ERRbadpw	2	Bad password The name/password pair in a Tree Connect or Session setup are invalid
ERRaccess	4	The client does not have the necessary access rights in the specified context for the requested function
ERRinvnid	5	The TID specified in a command was invalid
ERRinvnetname	6	Invalid network name in tree connect
ERRinvdevice	7	Invalid device Printer request made to non-printer connection or non-printer request made to printer connection
ERRqfull	49	Print queue full (files) Returned by open print file
ERRqtoobig	50	Print queue full-no space
ERRqeof	51	EOF on print queue dump
ERRinvpfid	52	Invalid print file FID
ERRsmbcmd	64	The server did not recognize the command received
ERRsrverror	65	The server encountered an internal error, for example, system file unavailable
ERRfilespecs	67	The FID and path name parameters contained an invalid combination of values
ERRbadpermits	69	The access permissions specified for a file or directory are not a valid combination The server cannot set the requested attribute
ERRsetattrmode	71	The attribute mode in the Set File Attribute request is invalid
ERRpaused	81	Server is paused (Reserved for messaging)
ERRmsgoff	82	Not receiving messages (Reserved for messaging)
ERRnoroom	83	No room to buffer message (Reserved for messaging)

Figure 17.15 SMB ERRSRV return codes

Error	Code	Description
ERRrmuns	87	Too many remote user names (Reserved for messaging)
ERRtimeout	88	Operation timed out
ERRnoresource	89	No resources currently available for request
ERRtoomanyuids	90	Too many UIDs active on this session
ERRbaduid	91	The UID is not known as a valid user identifier on this session
ERRusempx	250	Temporarily unable to support Raw, use MPX mode
ERRusestd	251	Temporarily unable to support Raw, use standard read/write
ERRcontmpx	252	Continue in MPX mode
ERRnosupport	255	Function not supported

Figure 17.15 SMB ERRSRV return codes

Error	Code	Description
ERRnowrite	19	Attempt to write on write-protected medium
ERRbadunit	20	Unknown unit
ERRnotready	21	Drive not ready
ERRbadcmd	22	Unknown command
ERRdata	23	Data error (CRC)
ERRbadreq	24	Bad request structure length
ERRseek	25	Seek error
ERRbadmedia	26	Unknown media type
ERRbadsector	27	Sector not found
ERRnopaper	28	Printer out of paper
ERRwrite	29	Write fault
ERRread	30	Read fault
ERRgeneral	31	General failure
ERRbadshare	32	An Open conflicts with an existing Open
ERRlock	33	A Lock request conflicts with an existing lock or specifies an invalid mode, or an Unlock request refers to a lock held by another process.
ERRwrongdisk	34	The wrong disk is found in a drive
ERRFCBUnavail	35	No FCBs are available to process request
ERRsharebufexc	36	A sharing buffer has been exceeded

Figure 17.16 SMB ERRHRD return codes

From NT LAN Manager Version 0.12 on, the server may send a 32-bit error code instead of the 8-bit error class and the 16-bit error code (see the SMB packet format in Figure 17.11).

17.7.1 Troubleshooting SMB

SMB errors may have many causes, as the variety of SMB error codes indicates. SMB retransmissions that are necessary when routers or switches drop packets due to overloads are a common cause of excessive response times. This is especially critical when NetBIOS is transported over LLC and the retransmissions must be performed on the SMB layer. Other error sources include server configuration faults, server performance problems, or problems with server applications being accessed by means of SMB.

17.8 The Microsoft Browsing Protocol

In addition to SMB, Microsoft networking environments often use the Microsoft browsing protocol. This protocol maintains the "browse list", that is, the list of computers that are visible when a Windows user browses the "Network Neighborhood" in the Windows Explorer application. Each node in the Microsoft network takes one of the following roles in this protocol:

Non-Browse Servers	Computers that cannot function as browse servers announce themselves every 12 minutes to the Master Browse Server, but do not maintain browse lists.
Potential Browse Servers	Computers that are not currently acting as Browse Servers, but can do so if necessary.
Backup Browse Servers	Computers that keep a list of known servers and domains for retrieval by the Master Browse Server.
Master Browse Server	(Also called the "Master Browser" or "Browse Master".) Computers in each network that respond to clients requesting the current browse list and that send lists of available servers to Backup Browse Servers.

In every domain or workgroup, at least one Master Browse Server exists for every 32 workstations. In TCP/IP networks, there is a Master Browse Server in each IP subnet. If no Master Browse Server can be found, an election takes place to select a new Master Browse Server. The NetBIOS name suffix 1E hex is used in this selection process (see Section 16.7.2). If an excessive number of such

election processes are observed, the causes may be a restarting NT server (NT servers are preferred Master Browse Servers) or intermittent problems of the current Master Browse Server.

17.9 Troubleshooting Application Protocols

The first step in diagnosing problems with application protocols such as FTP, Telnet, or HTTP is to analyze the server's message logs. If the logs do not contain any clue as to the cause of the error, then a connection attempt can be monitored using a protocol analyzer. The protocol trace usually permits prompt identification of the reason for the connection failure. If a ping to the server station is successful, but a login fails, this may be because the server application is not running, or because the router's access table disabled the corresponding TCP port. Other typical causes of trouble are defective client configurations (such as an incorrect FTP transfer mode, that is, ASCII rather than binary, or an incorrect Telnet terminal emulation, etc.), or incorrect user ID and password settings.

A common application-related problem is looping: this refers to an application sending the same command over and over again. In order to distinguish application looping from TCP retransmissions, it is important to conduct measurements on both the server and client segments to make sure that the repeated application requests are not caused by TCP responses not reaching their destination. Typical causes of application looping include bugs in the application software, poor bus performance on the server, memory allocation problems on the network interface, or problems with the network interface driver.

Another frequent source of application performance problems is simply inefficient software design. A number of applications are not designed to use the network infrastructure efficiently. They perform small, incremental file reads, or issue overlapping read commands, causing large amounts of data to be transported over the network repetitively.

17.10 Symptoms and Causes: Network Services and Applications in General

Symptom: **Application Looping**

Cause (1): Application software problem
Cause (2): Insufficient server bus performance
Cause (3): Memory allocation problems on the network interface card
Cause (4): Problems with the network interface driver

Symptom: Application Slow

Cause (1): Congested network
Cause (2): Overloaded server
Cause (3): Application performs small, incremental file reads
Cause (4): Application performs overlapping read commands (for example, every read command reads 20% or 30%, sometimes 100% of the data just read in the previous read operation)
Cause (5): Application performs overlapping write commands

Symptom: Connection Timeouts

Cause (1): System overloaded (memory, CPU load at server, client)
Cause (2): Network overloaded (a switch or router is dropping packets)

17.11 Symptoms and Causes: FTP and Telnet

Symptom: No Connection to FTP, Telnet Server

Cause (1): Incorrect user name (typing error, wrong keyboard driver, US/non-US keyboard layout)
Cause (2): Incorrect password (typing error, wrong keyboard driver, US/non-US keyboard layout)
Cause (3): User name, password changed
Cause (4): User account does not exist, no permission to log in
Cause (5): User account deleted
Cause (6): Mistyped or wrong IP address
Cause (7): TCP/IP driver misconfiguration
Cause (8): Domain name server incorrectly configured or not working (try to contact host by IP address)
Cause (9): Firewall blocks FTP or Telnet applications
Cause (10): FTP or Telnet client incorrectly configured for use across firewall (no proxy configured; wrong FTP proxy configuration)
Cause (11): FTP or Telnet server down (does it respond to ping?)
Cause (12): No route available to FTP or Telnet server segment (traceroute to server)

Symptom: FTP/Telnet Connection Timeouts

Cause (1): System overload (memory, CPU load at server or client)
Cause (2): Network overload (switch, router dropping packets)

17.12 Symptoms and Causes: Mail

Symptom: Problems Sending and Receiving Mail

Cause (1): No TCP connection to mail server
Cause (2): Mail server down
Cause (3): Server is running, but not listening on port 25
Cause (4): Incorrectly configured server software
Cause (5): Incorrectly configured mail client
Cause (6): Proxy Server Port is not on port 25

Symptom: Slow Mail Exchange

Cause (1): Large attachments
Cause (2): Communication via X.400 gateway
Cause (3): Slow connection to mail server
Cause (4): Proxy Server has a low performance

For additional excerpts from this chapter and other Network Troubleshooting book sections, be sure to regularly visit our web site at:

www.FreeTroubleshootingBook.com

New chapters will be posted every 2 to 3 weeks.
Be sure to visit our web site and vote for the chapters you would like to see posted!

Agilent Technologies