# Classic TCP/IP applications: TELNET, FTP, SMTP, NNTP and SNMP

Timo Harjunen
Antti Särkkä

02.11.1998

# Contents

# 1 TELNET

## *1.1 Introduction*

Where the TCP protocol makes it possible to connect the remote computers, the TELNET protocol makes it possible to use them. The TELNET protocol offers a user the possibility to connect and log on to any other hosts in the network from user's own computer by offering a remote log on capability. Historically TELNET was the first TCP/IP application and still is widely used as a terminal emulator. Today, while the applications are more and more equipped with the graphical user interface, the terminal-based applications are becoming minority among the applications, the TELNET has found its future as a toolkit lying below several client/server software. E.g. FTP, SMTP, SNMP, NNTP and HTTP are more or less dependent on the TELNET protocol.

## *1.2 Telnet model*

For the connections, TELNET uses the TCP protocol. The TELNET service is offered in the host machine's TCP port 23. The user at the terminal interacts with the local telnet client. The TELNET client acts as a terminal accepting any keystrokes from the keyboard, interpreting them and displaying the output on the screen. The client on the computer makes the TCP connection to the host machine's port 23 where the TELNET server answers. The TELNET server interacts with applications in the host machine and assists in the terminal emulation.

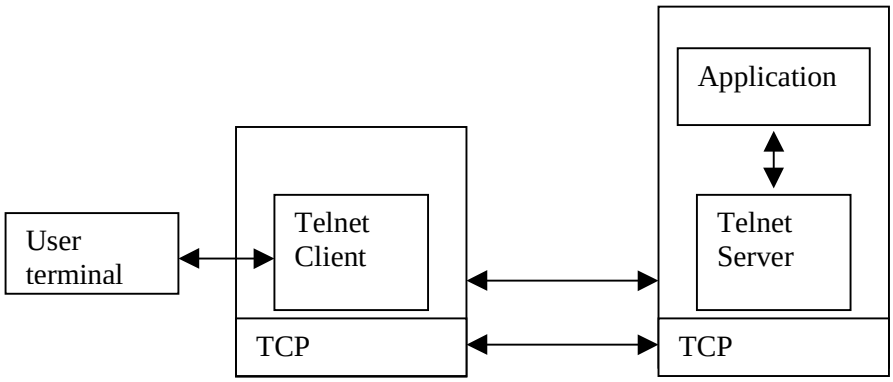


**Figure 1. TELNET protocol model**

As the connection is setup, the both ends of the TELNET connection are assumed to be originated and terminated at the network virtual terminal (NVT). The NVT is a network wide terminal which is host independent so that both the server and the client in the connection may not need to keep any information about each others terminal's characteristics as both sees each other as a NVT terminal. As there are several types of terminals, which may be able to provide additional services from those provided by the NVT, the TELNET protocol contains a negotiation method for the user and the server to negotiate changes to the terminal provided in the NVT. Typically the client and the server stays in the NVT just as long as it takes to negotiate some terminal type to be emulated. [Fei96]

## *1.3 Options*

The TELNET has a set of options and these options can be negotiated through a simple protocol inside the TELNET. The negotiation protocol contains commands DO, WILL, WON'T and DON'T. Following examples present the accepted command sequences:

DO (sender wants receiver to enable the option)

| | |
|---|---|
| WILL | (receiver acknowledges) |
| | |
| DO | (sender wants receiver to enable the option) |
| WON'T | (receiver will not acknowledge the request) |
| | |
| WILL | (sender wants to enable the option) |
| DO | (receiver gives permission) |
| | |
| WILL | (sender wants to enable the option) |
| DON'T | (receiver does not give permission to do so) |
| | |
| WON´T | (sender wants to disable option) |
| DON'T | (receiver has to answer OK) |
| | |
| DON'T | (sender wants receiver to disable option) |
| WON'T | (receiver must say OK) |

Mostly the options are used in the beginning of the connection to setup a desired set of options for the TELNET. In some cases the options are changed during the session. The key option to be negotiated is the terminal type.

The symmetry in the negotiation protocol indicates that some loops are possible. Without any further restrictions, the following sequence could take place:

DO TERMINAL TYPE VT100
WILL TERMINAL TYPE VT100
DO TERMINAL TYPE VT100
…

For this situation and similar situations the protocol introduces set of rules:

1. Parties may only request a change in the option (this rule overcomes previous problem)
2. If a party receives a request enter some mode that it is already in, the request should not be acknowledged
3. If the option affects the way, how the data is processed, the command must be inserted in the data stream exactly in the place where it is desired to take effect.
4. The rejected request should not be repeated until something changes in the operating environment E.G. the process runs other program or other user command is executed. [RFC854]

The options are set through TELNET commands. To indicate that the next byte is a command byte, the IAC (interpret as command) byte (0xFF) is sent. The data byte 0xFF is sent as two consecutive 0xFF bytes.

The option negotiation requires 3 bytes: IAC, request (WILL, WON'T, DO, DON'T) and the option ID byte to be enabled or disabled. The negotiations are either symmetrical or non-symmetrical. With a symmetrical option both sides may start the negotiation sequence. A nonsymmetrical option is always requested by the other part. As an example of a non-symmetrical option can be the linemode option, which can only be requested by the client. [Ste94]

The negotiation may require suboption negotiations. These negotiations take place when the option does not have only two modes: enable and disable. An example for a such subnegotiation is a terminal type negotiation. The Terminal type option is first enabled with a normal 3 byte negotiation:

IAC, WILL, 24 (24 = terminal type)

The server responds hopefully:

IAC, DO, 24

The server then asks the terminal type of the client:

IAC, SB, 24, 1, IAC, SE
(SB = suboption, 24 = suboption terminal type, 1 = sent your terminal type, SE = suboption end)

Client responds:
IAC, SB, 24, 0, '**V**´, ´**T**´, '**1**', '**0**', '**0**', IAC, SE
(0 = my terminal type, string VT100)

### *1.4 TELNET commands*

Typical for the terminals directly connected to a computer are that the keystrokes by the user are immediately interpreted by the computer's operating system. For the purpose certain keystroke combinations were invented. For example by pressing ctrl+'z', the processes were suspended or ctrl+'c' was used for killing current process. The TELNET cannot transmit such codes as they are since the codes are commands containing two keystrokes and do not map to the 7-bit ASCII chart used in the NVT. This requires that the client has to translate the terminal's control codes to the TELNET commands and transmit the commands to the server host's operating system.

As explained earlier, the TELNET commands are presented with IAC byte with a followed command and parameters. All the TELNET commands are presented in the Table 1. The TELNET command IP can be used for send what would be ctrl + 'c' in the keyboard and the command EC for what would be the backspace. The IP command is not always the right choice for interrupting the process because it acts similarly with the command sent from real terminal, which is interpreted right away while the IP signal may take some time to be transmitted over the connection and through the buffered data. For overcoming this problem the data mark (DM) command is introduced. With the DM command, the client gets server's attention faster and the client tells the server to throw away all the data but the commands. The client uses the Synch Signal TCP segment for the purpose. The Synch signal is marked as an urgent data. The server will throw away everything except the commands until DM is reached. The DM marks the spot where the data is no longer discarded and the TELNET resumes the normal operating mode.[Fei96]

| Name | Code | Description |
|---|---|---|
| EOF | 236 | End of file |
| SUSP | 237 | Suspend process |
| ABORT | 238 | Abort process |
| EOR | 239 | End of record |
| SE | 240 | Suboption end |
| NOP | 241 | No operation |
| DM | 242 | Data mark |
| BRK | 243 | Break |
| IP | 244 | Interrupt process |
| AO | 245 | Abort output |
| AYT | 246 | Are you there |
| EC | 247 | Escape character |
| EL | 248 | Erase line |
| GA | 249 | Go ahead |
| SB | 250 | Suboption |
| WILL | 251 | Option negotiation |
| WONT | 252 | Option negotiation |
| DO | 253 | Option negotiation |
| DONT | 254 | Option negotiation |
| IAC | 255 | Interpret as command |

**Table 1. Telnet Commands. [RFC854]**

### *1.5 TELNET for Client/Server applications*

The TELNET can be used as a tool set to build the client/server applications. As a basis for the application, TELNET rarely requires the terminal extensions and the negotiations but the TELNET operates in the basic NVT mode. The NVT is bi-directional half-duplex device, which contains a terminal and a keyboard. Basically the keyboard is the user keyboard. The keyboard produces client's outgoing data sent over the TELNET connection to the server. The printer is the user's display where the TELNET server sends the characters. The NVT is half-duplex. This means that in the TELNET protocol either the client or the server has the control. The control from the client to the server is changed by the CR/LF. The server uses the CR/LF for changing the line – not for returning the control to the client. The client receives the control after receiving the data from the server and accepting the GO AHEAD control code.

### *1.6 Telnet and security*

Since the eavesdropping and the snooping are easy to implement to any machine connected to a LAN and the fact that the password and the user ids are sent through the TELNET connection uncrypted, if not otherwise required, the TELNET protocol is a security risk. Therefore the TELNET protocol defines a option for the authentication. The actual authentication is exchanged in the authentication subnegoation. The TELNET's authentication options support such authentication standards like Kerberos, SPX, RSA, LOKI and SSA. [Fei96]

# 2 File Transfer Protocol

## *2.1 Purpose*

The File Transfer Protocol (FTP) is used to copy files between two computer systems over the TCP connection. The FTP overcomes the problem of different file systems used in the network. The different types of file systems introduces problems in how:

- The file names are converted
- The directories are used
- The files are accessed under restrictions
- The data and the text are represented in the files.

## *2.2 FTP Model*

In the FTP, the user communicates with a user interface in the local FTP client process. The local FTP client process makes a control connection to the remote server's FTP server protocol. FTP server protocol is located in the TCP port 21. The local FTP client acts as a protocol interpreter who interprets the user commands to the acronyms used between the client and the server protocol. The control connection is basically a simple TELNET's NVT session. The control connection is used a very simple way: The client sends commands across the control connection to the server. The server replies to the messages according to the server protocol.

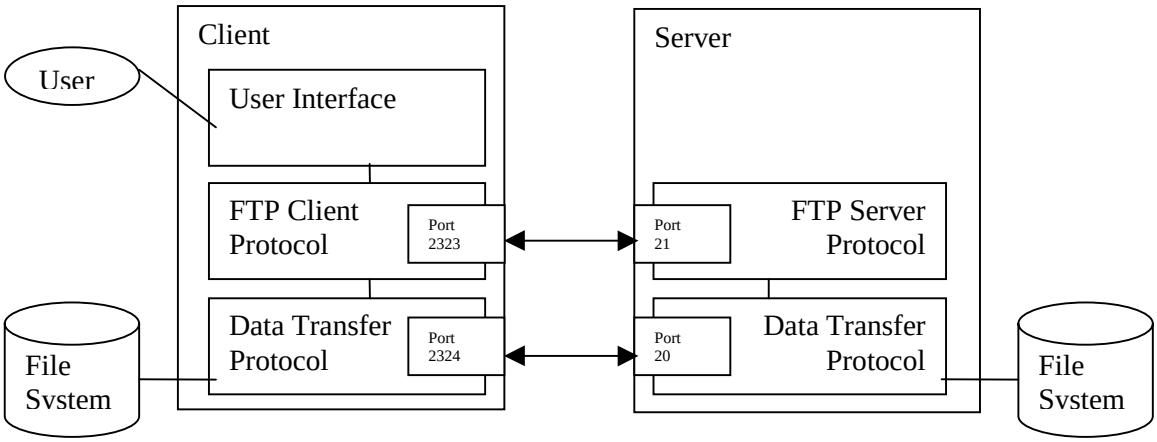


**Figure 2. FTP protocol model**

If the user request a data transfer, a special data connection is opened between the server and the client and the files are sent through this connection. Separate data transfer process created for the server and the client. The data connection exists until the command that it was created for is executed. Other FTP commands that require a data connection a new connection is created.

The data connection is used for three purposes:

1. For sending a file from the client to the server
2. To receive a file from the server
3. To receive listings of files or directories from the server

The data connection is created for each new file transfer. The data connection creation is always initiated by the client since the client is the provider for the command to be executed. The client selects a random (and free) TCP port number (over 1024) for the connection. The port number with the client's IP address identifies the client for the server. This address

information is transferred to the server across the control connection. This is done with the PORT command on the client. The server receives the client's address and opens the TCP connection to the given port. [RFC959]

Before sending the data over the data connection the both ends of the connection have to share the same view about the format of the data to be transferred. The FTP has several options for the data representation, which has to be specified before the data is transferred. The attributes to be specified are the data/file type, the data structure and the transmission mode.

The file type can be ASCII, EBCDIC, Image or Local. The ASCII and the EBCDIC are alternative ways to transfer text files. The image file type means continuos stream of bits and is used while transferring the binary files. The local file type is used when transferring the binary files between hosts with different byte sizes.

For the data structure there only two modes file or record. The file mode means that there is no structure just a continuos streams of bytes where the record mode indicates that the file is a sequence of records.

For the transmission modes the stream mode transmits the data as a stream of bytes. The FTP relies in the stream mode to the TCP's means to provide the data integrity. The additional information in the stream is not necessary. Except for the file structure record where the end of records and the end of files are transmitted. Otherwise server indicates that the end of file is reached by closing the data connection.

In the block mode a file is transmitted as a series of data blocks. Each block is preceded by the 3-byte header. The header contains a byte counter (16 bits) and descriptor codes (8 bits). The byte counter indicates the total length of the data block (indicates also the start of the next block). The descriptor codes contains the end of record, the end of file and the restart markers. The EOF is used to indicate the last block in the file transfer. The EOR is used to mark the record boundaries. The restart marker indicates whether the block contains a (NVT) text string, which can be used to identify the restart point. If the transfer fails, the restart point can be used for the retransmission.

The compression mode means that the files are compressed before the file transfer with a method specified in the RFC. The compression method is simply based on the collapsing strings of repeated bytes (like repeated spaces in the text files). [Fei96]

### 2.2.1 FTP commands

There are two types of commands in the FTP: the text-based dialogue and the file transfer protocol commands. The text-based dialogue commands are interpreted to the file transfer protocol commands, which are sent over the control connection to the server. The dialogue commands includes commands for sending and receiving files, changing directories in the remote host, for setting the transfer modes etc.

File transfer protocol contains set of command words and their parameters and numeric codes as responses. The command words can be classified to the access control, file management, data format setting, file transfer, site, error recovery and restart commands.

The Following table describes the command words used over the control connection

| Command type | Command | Parameters | Description |
|---|---|---|---|
| Access | USER | UserId | Identify user |
| | PASS | Password | Provide password |
| | ACCT | AccountId | Provide account |
| | REIN | - | Reinitialize start state |
| | QUIT | - | Logout |
| | ABOR | - | Abort previous command |
| File mngt | CWD | Dir name | Change directory |
| | CDUP | - | Change to parent directory |
| | DELE | Filename | Delete file |
| | LIST | Dir name | List information about files |
| | MKD | Dir name | Make a directory |
| | NLST | Dir name | List the files in the directory |
| | PWD | - | Print the name of the working directory |
| | RMD | Dir name | Remove directory |
| | RNFR | Filename | Identify file to be renamed |
| | RNTO | Filename | Rename the file |
| | SMNT | Filename | Mount a different file system |
| Data format | TYPE | A(scii),E(bcdic),I(mage),N(nonprint),T(elnet),C(ASA) | Identify the data type for the transfer |
| | STRU | F(ile),R(ecord) | Organization of the file |
| | MODE | S(tream),B(lock),C(ompressed) | Transmission format |
| File transfer | ALLO | No. of bytes | Allocate storage for data |
| | APPE | Filenames | Append local file to remote file |
| | PASV | - | |
| | PORT | IP Addr+port | Identify IP address and port for data connection |
| | REST | Marker value | Identify restart marker |
| | RETR | Filename | Get a file |
| | STOR | Filename | Put a file |
| | STOU | Filename | Store unique: version of the file with unique name |
| Misc. | HELP | - | Information about server implementation |
| | NOOP | - | Ask server to return an OK reply |
| | SITE | - | Server specific subcommands |
| | SYST | - | Identify servers operating system |
| | STAT | - | Connection status request |

**Table 2. FTP commands. [RFC959]**

### 2.2.2 FTP replies

The server uses reply codes for answering to the client's request. The reply code consist of the actual code (see Table 3) and the text message.

| Reply | Description |
|---|---|
| 1yz | Positive preliminary reply |
| 2yz | Positive completion reply |
| 3yz | Positive intermediate reply, command OK other command required |
| 4yz | Transient negative completion reply, command can be repeated later |
| 5yz | Permanent negative completion reply, command should not retried |
| X0z | Syntax errors |
| X1z | Information |
| X2z | Connections (either control or data) |
| X3z | Authentication and accounting |
| X4z | Unspecified |
| X5z | Filesystem status |

**Table 3. FTP replies. [Fei96]**

## *2.3 FTP Session example*

The following example presents how the FTP is used to transfer a file from the remote host. The file to be transferred is ASCII type.

| Session print-outs | Explanation |
|---|---|
| Cdlinux01> ftp | Starting the FTP from command line |
| Ftp>open ftp.funet.fi | Open the control connection to host ftp.funet.fi |
| Connected ftp.funet.fi | Client informs that the connection is OK. |
| 220 – ftp.funet.fi FTP server … ready | Server's greeting reply |
| Name (ftp.funet.fi): | Local client asks for user id. |
| Anonymous | User types anonymous as user id |
| 331 Guest login ok, send your e-mail address as password | |
| Password: | Local client prompts for the password |
| 230 Guest login ok | Server grants the access |
| ftp> cd pub | User changes the remote directory to pub |
| 250 CWD command successful. | cd command was sent to server as CWD command the server directory is now changed to pub directory |
| Ftp> get index.txt | User asks for a copy of index.txt file. The data connection will be created. |
| 200 PORT command successful | Local client has obtained a port and sent the information about the port to the server as PORT command, telling the server to connect this port. |
| 150 opening ASCII mode data connection for index.txt (130034 bytes) | The data connection is opened for the transfer. |
| 226 Transfer complete | The transfer is complete. |
| local: newfile remote: index.txt | A new local file is created |
| 132002 byte received in 1,22 seconds (xx Kbytes/s) | |
| ftp> quit | Session closing command from client |
| 221 Goodbye. | Last server's reply to indicate that the closing is OK. Control connection is terminated |

## *2.4 Trivial File Transfer Protocol*

The Trial File Transfer Protocol (TFTP) provides a very simple protocol and the functionality for transferring files. The TFTP may sound like it is some kind of simplified version of the FTP. This is not the case. The only thing the TFTP and the FTP has in common is actually the fact that both are capable of transferring files through the network. Due to simplicity, the TFTP is mostly used for loading e.g. operating system for the diskless workstations or download the initialization and configuration files for the software during the boot phase. The TFTP transfers its data as UDP datagrams. The TFTP requires from the system only the UDP and IP drivers to be able to operate and transfer files.

The TFTP sends data as 512 bytes blocks including 4 byte header. Each block is numbered in the header. The numbering starts from 1. Either ASCII or binary information can be transferred. The TFTP is capable of sending and receiving files. The receiving file is done with the read request primitive and the sending with the write request primitive. The client gets a free port and sends a read or write request primitive to the server's UDP port 69. The server changes the port to other, which it will use for the rest of the session while transferring the files with the client. Since the all the blocks should be the size of 512 bytes, the end of file block is indicated with a block, which size is less than 512 bytes. The read request is replied with the DATA datagram from the server. The client acknowledges the DATA and the server sends DATA again and the client acknowledges them as well. This is repeated until the end of file is reached or an error is found. The write request on the other hand is acknowledged by the server and the client can then start sending data. The server acknowledges the data until the end of file is reached or error is found.

For the security aspects the TFTP protocol is not a secure protocol since it does not contain any access control or authentication mechanisms. Therefore TFTP should not have an access only to the public files in the system and the other files should not be seen. [RFC1350]

# 3 SMTP

## *3.1 Purpose*

The simple mail transfer protocol (SMTP) provides a simple way to transfer electronic mails between hosts. In the SMTP there are two different roles: sender and receiver. The sender is a client and establishes a two-way transmission channel (TCP connection) to the receiver. The receiver can be the "real" receiver or some intermediate host on the way to the "real" receiver. The protocol commands are generated by the sender and are send to receiver, which replies the responses. [RFC821]

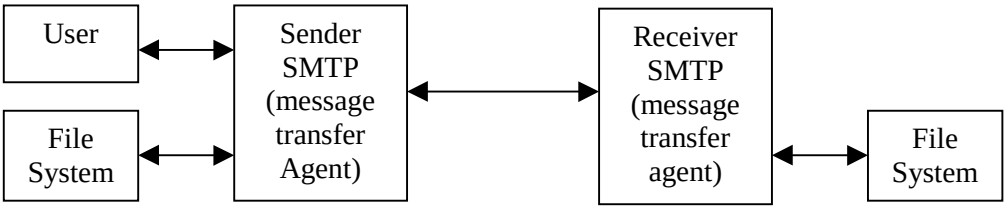


**Figure 3. SMTP Protocol model**

The Sender and the receiver are called message transfer agents (MTA). The communication between two MTAs uses the NVT ASCII like TELNET or FTP's control protocol.

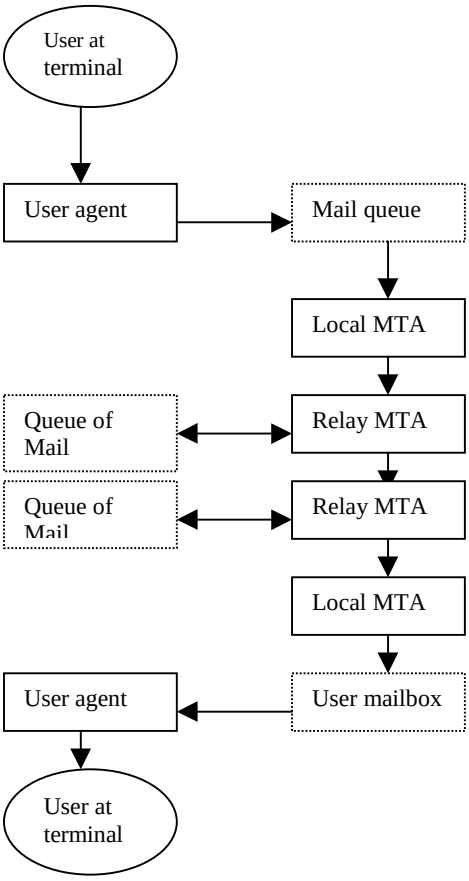


**Figure 4. Mail delivery through mail transfer agents. [Ste94]**

The MTA may pass the mail directly to the receiver by taking a direct contact to the receiver host's SMTP. The mail can be delivered through a relay MTAs like in
Figure 4. The relay MTA is centralized MTA under one organization and every outgoing or incoming mail is transferred through the relay MTA. This is useful also for more simple local MTA configuration and is useful for hiding the individual mail servers behind "mail hub" (relay MTA). The relay MTA takes the direct connection to the receiver relay MTA, which directs the mail to the local MTA on the receivers host.

## *3.2 Protocol and commands*

The actual mail transfer takes place between sender and receiver SMTPs. Both the sender and thereceiver uses the NVT terminal. The sender is the client in the protocol (and receiver acts as a server). The client sends SMTP command to server, which replies with numerical messages (with similar logic to FTP). A set of 12 commands that client can use is specified by the RFC821.

| Command | Description |
|---|---|
| HELO | Identifies the sender to the receiver. Host name as an argument |
| MAIL | Starts mail transaction and identifies the mail originator |
| RCPT | Identifies the recipient (several recipients several RCPT lines) |
| DATA | Sender's data in the text format. Each line terminated with CR/LF. The mail ends with CR/LF.CR/LF |
| RSET | Abort transaction |
| NOOP | Asks for positive reply |
| QUIT | Ask for positive reply and close the connection |
| VRFY | Verifies that receiver is valid |
| EXPN | Asks receiver to confirm that name identifies a mailing list |
| HELP | Ask information about counterparts implementation and commands |
| TURN | Switch roles, Sender becomes receiver and other way around |
| SEND | If the recipient is logged in deliver the mail straight to the recipients terminal |
| SOML | Send or mail. |
| SAML | Send and mail. |

**Table 4. SMTP commands**

### 3.2.1 SMTP replies

SMTP uses similar technique for replying as FTP.

| Reply | Description |
|---|---|
| 1yz | Positive preliminary reply |
| 2yz | Positive completion reply |
| 3yz | Positive intermediate reply, command OK other command required |
| 4yz | Transient negative completion reply, command can be repeated later |
| 5yz | Permanent negative completion reply, command should not retried |
| X0z | Syntax errors or unknown command |
| X1z | Reply to information request |
| X2z | Reply referring to a connection |
| X3z | Unspecified |
| X4z | Unspecified |
| X5z | Reply indicated the status of receiver mail system |

**Table 5. SMTP replies**

The very basic mail sending sequence contains following steps:
1. Receiver announces its name to the sender
2. Sender announces its host name
3. Sender identifies the message originator
4. Sender identifies the recipients (one or several)
5. Sender transmits the mail data
6. Sender transmits CR/LF.CR/LF sequence indicating that sending is complete

After this sender can continue or quit the session.

### 3.2.2 Mail sending as dialogue between server and client

This example shows the dialogue between the server and the client during the mail exchange

*220 goofy.gov Sendmail x.yz ready at Mon, 26 Oct 1998 18:00:21 –0400*
HELO tpk.fi
*250 Hello foobar.fi, Pleased to meet you*
MAIL FROM: <mara@foobar.fi>
*250* <mara@foobar.fi>*… Sender ok*
RCPT TO: <goofy@goofy.gov>
*250* <goofy@goofy.gov>
DATA
*354 Enter mail, end with "." on a line by itself*
qwerty
qwerty
qwerty
qwerty
.
*250 Mail accepted*
QUIT
*221 goofy.gov delivering mail*

## *3.3 Timestamp and mail route*

The SMTP protocol adds the time when the mail is sent to each message. The format for the timestamp may vary but newer implementations reports the time as local time followed by the offset from the Greenwich time. The SMTP protocol keeps track on each mail transfer agent that the message was relayed through. Each message transfer agent adds its own timestamp to the header of the message. The timestamp contains the host that sent the message and the host that received the message (current MTA) and the time when the mail was received. These timestamps provide the exact route from the sender to the receiver and this information can be used for E.G. tracing e-mail delivery problems. The timestamps may give strange time sequences but normally this is due to the fact that the computer clocks are inaccurate. [Fei96]

## *3.4 Extensions: MIME*

One shortage in the SMTP protocol is that it cannot send other types of messages but only simple text messages. This problem is solved with either an extended SMTP message transfer agents or with a standard SMTP by converting the file so that it looks like the ordinary NVT text.

The Extended MTA needs one more command to the SMTP (e-MTA uses then extended SMTP). The e-MTA sends EHLO command in the place of a HELO. If the server replies to EHLO, the client will assume that the server is able to handle the MIME (Multipurpose Internet Mail Extensions) messages. If the reply is the error message, the MTA can convert to SMTP and send HELO. [RFC1425]

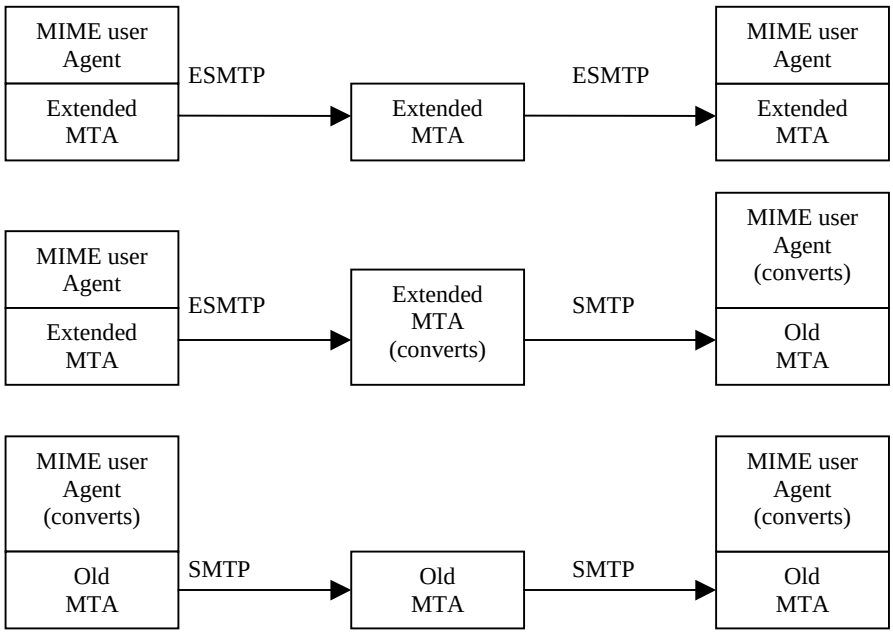


**Figure 5. MIME coding solutions**

The protocol dialogue differs a little from what was presented in the chapter 3.2.2. Aside with the change from HELO to EHLO, the MAIL FROM field contains an additional BODY parameter E.G. BODY=8BITMIME. The format of MIME message sent after DATA command contains some additional headers from the normal internet text message's normal headers (from:, to: and subject, date). The MIME message contains additional sc. introductory headers. These introductory headers contain information about the structure and the content of the message. E.g. if there is several parts in the message, the boundary between parts is announced so that the receiver can distinct the separate parts of the message. [RFC1521]

Actually MIME contain 5 other header fields (+ the BOUNDARY)

- MIME-version :
- Content-Type:
- Content-Transfer-Encoding:
- Content-ID:
- Content-Description:

The content-types presented in Table 6. The content type describes the content of the message. The content type describes the supertype of the message. The subtype describes the actual contents of the message by describing the message format. As seen in the Table 6 for example the content type image indicates that the content is a picture of some format, where the subtype describes the exact format of the picture.

| Content-Type | Subtype | Description |
|---|---|---|
| Text | Plain<br>Richtext<br>Enriched | Unformatted text<br>Text with simple formatting<br>Refinement of richtext |
| Multipart | Mixed<br>Parallel<br>Digest<br>Alternative<br>Appledouble<br>Header-set | Multiple body parts processed sequentially<br>Multiple body parts processed parallel<br>An electronic mail digest (each part is message itself)<br>Several renditions (postscript or text for example) |
| Message | Rfc822<br>Partial<br>External-body | Content is RFC822 mail message<br>Part of message<br>Pointer to actual message |
| Application | Octet-stream<br>Postscript<br>(several others) | Arbitrary binary data<br>Formatted postscript file |
| Image | Jpeg<br>Gif<br>Ief<br>Tiff | Jpeg file<br>Gif file |
| Audio | Basic | Encoded using 8-bit ISDN u-law format |
| Video | Mpeg<br>QuickTime | ISO 11172 format<br>QuickTime format |

**Table 6. MIME content types**

The content type table shows just current view to supported content types and subtypes. The amount of content types and subtypes are expected to increase over time to contain e.g. new file formats as subtypes. The new content types and subtypes are registered through Internet Assigned Number Authority (IANA).

The Content-Transfer-Encoding field describes the encoding formats used for coding the contents into message. These formats are:

- 7-bit, NVT ASCII (default value for coding)
- Quoted-printable, Is mostly 7-bit ASCII but characters from eight bit set can be used by putting the equal mark (=) as a first letter followed by the two hexadecimal digits. The hexadecimal digit is code's value in 8-bit ASCII.
- Base64, coding where the entire content (binary or whatever) is mapped to a representation, which looks like a text.
- 8-bit, A sequence of lines including the 8-bit characters.
- Binary, 8-bit data that need not contain lines.

The Content-ID and the Content-Description can be used for further describe the contents of the message body.

# 4 NNTP

The Network News Transport Protocol (NNTP) is used in transferring news articles between news clients and news servers, where news articles are stored. The NNTP was developed because still widely used mailing list doesn't scale easily for large amount users, for separate copy are sent to everybody's mailbox and this demands network bandwidth and disk space.

The NNTP is based on RFC977 [977], which is dated on February 1986. The news article format is defined in the RFC 1036 [1036] which also describe the USENET news system. The news article format is based closely to the general Internet message format defined in the RFC 822 [822].

In the news system articles belong to the newsgroup(s) hierarchy e.g. comp.protocols.snmp. These newsgroups form subnet hierarchy. Currently there exist about 560 main level Usenet hierarchy groups [wmc] e.g. alt, comp or sfnet. These are further divided to the second level groups and hierarchy like comp.protocols or sfnet.tietoliikenne. Currently in the Usenet news system there exists over 20 000 different newsgroups. There exist also local news servers, which can contain the hierarchy of their own. The news articles are separated with the unique message ID, which include ASCII string followed by full domain name of the news server, which originally received the article, e.g. 1a2b3@news.server.com. However in the newsserver newsfeed articles are arranged according to the newsgroup hierarchy. Articles have also server internal article number, which is used for client to identify the articles.

The Usenet is rather a logical network than a physical network. Domains, newsgroup hierarchy, newsfeed and different physical networks form this logical network. The usenet is directed graph, however newsfeed can be bi-directional.

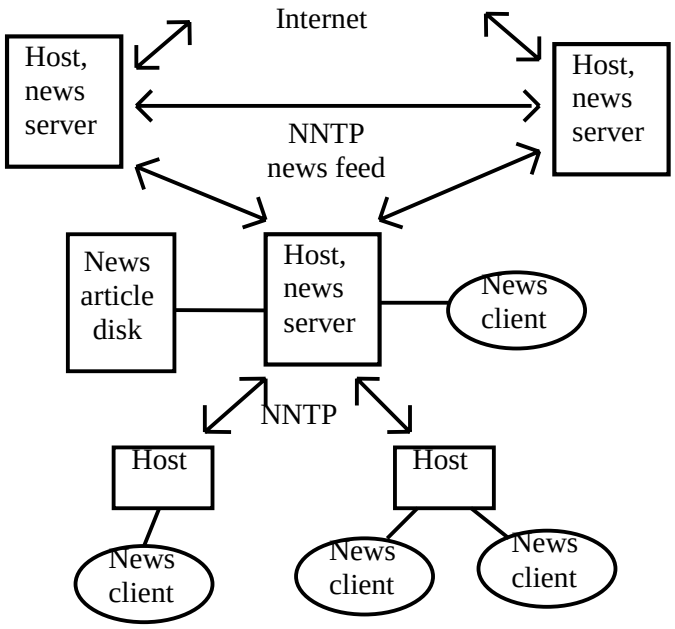


**Figure 6 News system**

## *4.1 Protocol function*

The NNTP is typical server-client protocol, where a news client initiate connection over TCP and send ASCII commands to a server which responds with a numeric response followed by optional (depending on the command) ASCII data. The TCP port number for NNTP is 119. The news server functions as interface between news database and news client just like other common Internet applications. Server does not perform any presentation level functions, these are left to the client programs.

### 4.1.1 Command and response message format

The NNTP has SMTP-like ASCII based commands and responses. Client sent command words (case insensitive) can include in some case parameters, which are separated from command by space or tab characters. The NNTP define two different responses: textual and status. The text responses are meant to be displayed by the news client. Textual responses are sent after numeric status response, which indicates that text response follows.

Status responses are sent by the server to response the last command received from the client. The status response starts with three digits and can be followed by parameter(s) or text, separated by space or tab character. Responses are not normally displayed, however the news client may show certain informative, unusual or error condition status responses e.g. "no such newsgroup" or "access permission denied". The numeric response codes have quite a same meaning as with other TCP/IP applications.

**Table 7 Response code allocation for NNTP**

| | |
|---|---|
| 1xx | Informative message |
| 2xx | Command ok |
| 3xx | Command ok so far, send rest of it |
| 4xx | Command was correct, but couldn't be performed for some reason |
| 5xx | Command unimplemented, incorrect or serious program error |
| x0x | Connection, set-up and miscellaneous messages |
| x1x | Newsgroup selection |
| x2x | Article selection |
| x3x | Posting |
| x4x | Distribution functions |
| x8x | Non-standard extensions |
| x9x | Debugging output |

Commands can be 512 characters long and have to end with CR-LF (carriage return - line feed). With the 8 bit transport channel 7 bit ASCII characters are transferred right justified and high order bit set to zero. Possible command words [and parameters] are:

- *Article, body, head* or *stat* [message number or message-id] fetch entire news article, header or body of article. The STAT command set "current active pointer" for that client.
- *Group* [group name] specify current default newsgroup of that client, the response include number of articles of the group and also the first and last number of articles stored. The last subtracted from the first are not necessarily same as number of articles due to different expiration times or cancelled articles. The client program can fetch all subscribed groups automatically and show all of them or only unread articles including groups.
- *List* command response include all (e.g. all 20000) groups of the server with name of the group, first and last index number of the articles and are posting allowed for that group.

- *Newgroups* [date, time] command is used by client to check are there any new newsgroups created after the date and time specified.
- *Newnews* [newsgroup, date, time] returns new articles (message-id) posted or received to that group after specified time.
- *Next* and *last* is used to set "current article pointer" to the next or last article of the group.
- *Post* command is used to post articles, the article format should fulfil RFC 1036. The response tells is article posted successfully, posting not allowed or posting failed.
- *Quit* command is responded by the server and the connection is closed.
- *Slave* command indicates that the client connection is to slave server not a normal newsreader. This is used to separate the subsidiary servers from single user connection.

There are extensions to these basic commands, e.g. with *xover* command can be fetched several article headers by one command.

### 4.1.2 Control messages

When article includes control field in the header, the body is interpreted as a control message. The first word is the name of the control message and subsequent words are parameters. Possible control messages are:

- *Cancel* [message-id] is used in cancelling news article. This message can be created by the article original sender or administrator. This is checked from- or sender-field of the header.
- *Ihave / sendme* [message-id] are used in distributing articles. When host A tells with ihave message to host B that it has the article with message-id, host B can ask to send it or reject it with sendme command.
- *Newgroup* command is used by the news server or hierarchy administrator to create a new newsgroup.
- *Rmrgp* is used correspondingly to remove a newsgroup.
- *Sendsys* command is used by a server (or anyone) to get sysfile from other server. Sysfile lists all neighbours and newsgroups sent to these neighbours.
- *Version* of the software running in local system are mailed back to the originator of the version command.

In the following table is presented a command line example for message interaction between server (S) and news client (C):

**Table 8 NNTP example**

| | | |
|---|---|---|
| C | "connect request to port 119" | TCP/IP connection to server's port 119 |
| S | 200 news.server ready posting ok | connection succeed |
| C | list | client ask to list all newsgroups |
| S | 215 list of newsgroups follows<br>alt.all 1234 1567 y<br>… | server accept command and sends all newsgroups subscribed to server in format: name of the group; first and last article number; posting allowed y/n |
| C | group comp.protocols.snmp | client selects group comp.protocols.snmp |
| S | 211 78 2143 2233 comp.protocols.snmp | command successful code: 211; total number of articles: 78; first article index, last article index |
| C | head 2143 | client asks to send header of article 2143 |

| | | |
|---|---|---|
| S | 220 2143 <6789@news.domain><br>article retrieved header follows<br><header> | server accept request and send article header |
| C | body 2143 | client asks to send body of article |
| S | 220 2143 <6789@news.domain><br>article retrieved header follows<br><body> | server accept request and send body of the article |
| C | Quit | client quit the connection |
| S | 205 news.server closing connection. Goodbye | server acknowledged closing |

### *4.2 News article format*

The USENET article format is defined in RFC 1036. Article format is based on mail message format defined in RFC822. Required headers for the news article are:

- *Relay-version* defines the software version of the last relay server.
- *Posting -version* specifies the software of news client.
- *From* field tells originator of the article in normal Internet address format.
- *Date* specifies the date and time when article is sent to the news system by news client in format: Wdy DD Mon YY HH:MM:SS TIMEZONE
- *Newsgroups* field specifies what newsgroup(s) article is posted to. Note that wildcards are no allowed with this header.
- *Subject* of the article.
- *Message-ID* defines unique identifier of the article. This is generated by the host that first received the article.
- *Path* specifies route that the article has propagated. The last system is marked the first in header line.

Correspondingly optional headers are:

- *Follwup-to* specifies the newsgroup(s) to article are send, when it is a response to the other article.
- *Expires,* with this header user can define then article are deleted from the news database e.g. in case that article contain the information that obsolete in certain time. Normally news servers have their own expiring policy based on disk space and newsgroup importance defined by administrator.
- *Reply-To* defines the mail address if it is different from defined in from field.
- *Sender* field is present if user manually enters the from field. This is generated by host software. However currently this field may also entered manually.
- *References* field includes message-id(s) which article is follow-up and response.
- *Control* field defines that body of the article containing this header is a control message.
- *Distribution* field alters distribution scope of the article defined in newsgroups field. The receiving site must subscribe specified newsgroups AND belong to one of the specified distribution.
- *Lines* field specifies amount of the article lines.
- *Organisation* field defines with a short phrase the organisation that sender belongs.
- *Approved* field is required for the article posted to moderate newsgroups. This is added by the group moderator and it contains normally his/her mail address.

Also MIME type coding is coming more popular in Usenet environment, this have to be present also with optional header fields.

### *4.3 Client programs*

The NNTP related standards do not define anything about the presentation layer so this all is left to the client programs. A news clients normally stores subscribed newsgroups and information about already read news articles for that group in a .newsrc or corresponding file. So then connected again only subscribed groups with unread articles can be fetched. A news client can also contain some features like killfile. This file defines which articles are not fetched or shown based on some header field like from, sender or subject.

Currently there exist dozen of different client programs i.e. newsreaders for different operating systems. Most of them are separate programs but currently one of the most popular way is to integrate news client to Web browsers e.g. Microsoft IE and Netscape. In the UNIX environment GNUS with emacs and rn (ReadNews) and its variants (Trn/Tin, Xrn) are popular client programs. In PC environment popular news clients are e.g. NewsXpress, Agent or Outlook Express.

### *4.4 Server programs*

In the UNIX environment most popular and in fact de facto server program for handling client request and the Usenet newsfeed is INN (*InterNetNews*)[INN], which is provided by the ISC. The INN provides both NNRP (Network News Reading Protocol) and newsfeed (INNDaemon) servers. This software can be also compiled for the Linux. Only newsfeed functionality including Diablo and Cyclone (from Highwind SW) are also popular newsfeed server programs [Dia], [High]. NNTPcache, Leafnode and Newscache provide newsreader server functionality for small implementation. There exist also server programs for other operating systems e.g. Microsoft provide this kind a feature.

Expiring of the articles is important parameter for the news server, it defines when (amount or time limit) articles of that group are deleted from the news database of that server. Normally this is compromise from disk space and administrator policy. For storing older articles have been created the system called *Dejanews*. The newsfeed parameters for timing and newsgroups can also effect the service quality of the NNTP server. If articles are transferred at nights or other minor load time, "real time" news discussion is not possible.

Currently the Web and HTTP based news systems and user interfaces are developed. In this case news articles are retrieved by HTTP server from news database and user interface is provided with server formatted HTML link page and with normal browser.

#### 4.4.1 Statistics from news

Table 9 presents a server newsfeed statistics found from Web [cnic]. These statistics are from cyclone.news.indirect.com newsfeed server during one day (20.10.98). This East Coast of US located server is connected to 60 other newsfeed servers.

**Table 9 Newsfeed statistics in articles** [**cnic**]

| In | Connects | Offered articles | Accepted articles | Bytes |
|---|---|---|---|---|
| | 18395 | 21,8 millions | 782 000 | 20,5GByt |
| Out | Connects | Offered | Accept | Bytes |
| | 2954 | 29,2 millions | 9,4 millions | 75GByt |

This table shows one example about level of the traffic in Usenet. However it shows that the level of transferred bytes are quite a high, for with link of 34Mbit/s transmission of 75 GBytes lasts 37 minutes.

# 5 SNMP

The *Simple network management protocol* (SNMP) is a standard defined by IETF for network management of the TCP/IP based networks. The network management system is consists of *network management stations* (NMS) and *network elements* (NE). The network management stations (also referred as *managers*) are normally workstations used by network administrator. Network elements can be anything that run TCP/IP suite e.g. host, switches, routers, X terminals, printers and so on. The NE running management software is normally called as *agent* [Ste96].

Whole network management system of TCP/IP networks consist of three entities:

- *Management Information Base* (MIB) defines variables what network element maintains and network manager can query or set.
- *Structure management information* (SMI) defines a set of common structures and an identification scheme used to reference the variables in MIB.
- *Simple network management protocol* itself defines protocol for communication between network manager and agents. This protocol is simply request-reply protocol, which rely on UDP as transport protocol.

## *5.1 Standardization*

The SNMP has historic background in development of CMIP/CMIS (Common Management Information Protocol/Services), which are defined by ISO. In the late 80's when the Internet started grow, it was found that this ISO/OSI based management system is too heavy and complex for the TCP/IP based networks. So in the begin of 90's in guidance of the IETF was developed simpler version from this quite complex network management protocol, which was mainly intended for traditional telephone networks. Currently in telephone networks widely used Q-interfaces are however based on CMIP/CMIS.

For the TCP/IP networks management there exist currently two version: SNMP(v1) and SNMPv2. The SNMPv1 is defined in RFC 1157 [1157]. The community based SNMPv2 is defined in RFC 1901 [1901] and RFCs 1902-1909 contain more accurate definition for SNMPv2 (also referred SNMPv2c). The user based security model for SNMPv2 is defined in RFC 1910 [1910]. The SNMPv1 is in standard state and SNMPv2 still in draft or proposed state. In the January 1998 released also proposed RFCs (2271-2275) for version three (SNMPv3) [2271].

The basic structure of the MIB is defined in RFC 1212. In RFC 1213 are defined second and more extended version called MIB-II [1213]. Both of these are currently in standard state. There exist also extension RFCs for MIBs. These are often specific for some protocol, which are released after MIB-II.

The original version of SMI [1155] is defined in RFC 1155. The SMIv2 is defined in various different RFCs. These definitions have often followed some new protocol or feature.

This presentation is mainly based on version 1 and version 2 is referenced when major differences exist. More efficient security model containing SNMPv3 is presented shortly.

## *5.2 Protocol function*

The SNMP is a message based simply request-reply type protocol, which uses normally UDP as transport protocol. There exist two basic type of message interaction between the manager and the agent: the manager sent get and set messages and the agent sent trap messages. The SNMP manager sends set and query messages to port 161 and SNMP agent response to this message. Agent-initiated trap messages include information from some event and are sent to manager port 162. Trap messages are not acknowledged. Because of two different port numbers are used, one system can easily be the manager or agent [Ste96].

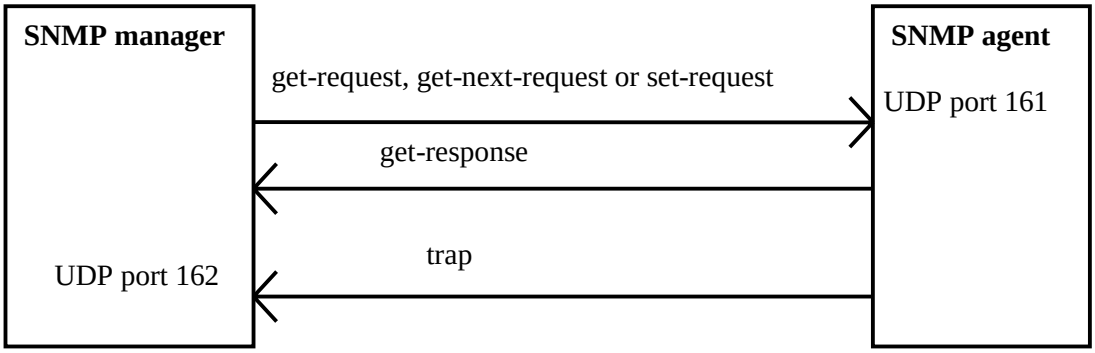


**Figure 7 SNMP basic function**

Because the SNMP normally rely on UDP as the transport layer, message or response can disappear or corrupt. So normally timers and retransmission scheme are needed in the manager side. The error check for SNMP message is handled at UDP level, i.e. optional checksum field in UDP header is used. The SNMPv2 defines also usage of other transport protocol than UDP e.g. AppleTalk, OSI and TCP.

The SNMPv1 (and community-based version of SNMPv2c) do not include any security options, like authentication. Whole definition of SNMPv2 includes support for authentication, encryption, authorisation and access control. Development of SNMPv3 is based mainly on SNMPv2 defined security models. Security is one key issue, because with SNMP are transferred important information for network function.

## *5.3 Message format*

SNMP messages use formal specification called *Abstract Syntax Notation number 1* (ASN.1). Actual encoding of the bits of the SNMP message uses correspondingly *Basic Encoding Rules* (BER). Because of this coding general bit level structure of SNMP message cannot present, only message field positions can be drawn, (see Figure 8). In practise coding one integer byte demand three bytes, one byte defining that value is integer (tag), one specifies number of integer bytes and last byte is the real value.

In the Figure 8 are presented basic SNMPv1 message formats and encapsulation to IP and UDP datagram. As seen get/set and trap messages have different SNMP header structure.

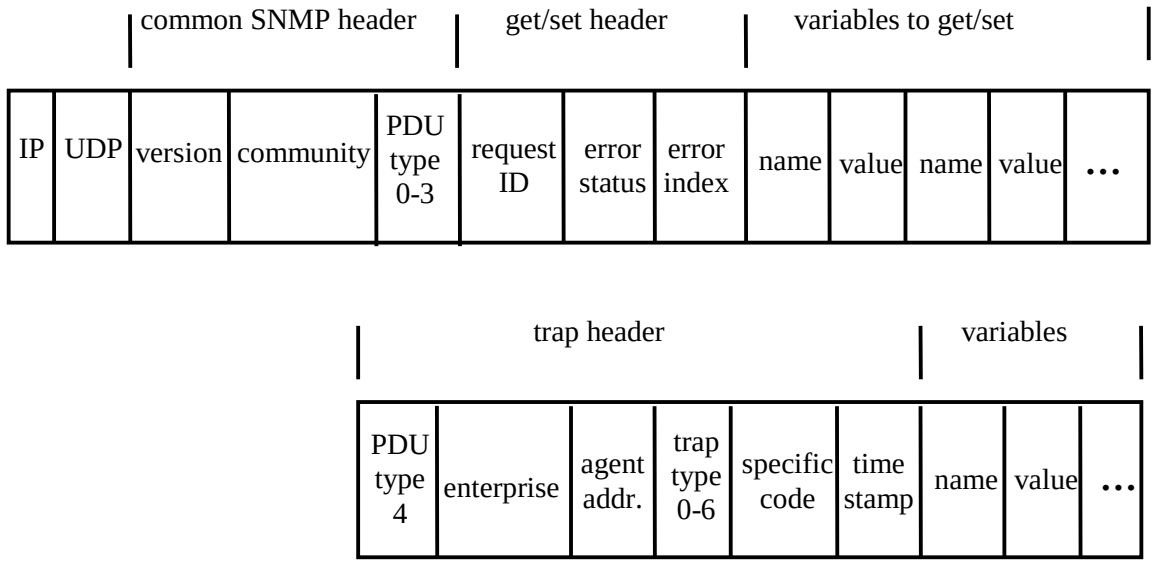


**Figure 8 SNMPv1 message formats**

The SNMP header includes common part for both types of messages. It consists of version number, community information and PDU type. The SNMP *version* type numbering is started from zero i.e. SNMPv1 has version number 0 in SNMP message header. The *community* is character string, which can be used as plaintext password (SNMPv1, SNMPv2c) between the manager and agent. This one of the main weakness of the SNMPv1 because this plaintext password is quite a easy to capture and misuse. The *PDU type* defines SNMP message type. In the SNMPv1 there exist five different messages:

- *get-request* command are used by network manager to fetch management data from the SNMP agent of the network element. The data object to fetch are specified by the name in message field, the value field are set to null in this query message. The get-request operator can fetch only one variable in time e.g. one address of whole routing table.
- *get-next-request*, because data objects in MIBs are arranged in sequential order with get-next-request can be fetched next variables by giving previously fetched variable name as parameter. The exact name (and value) is included in the response.
- *set-request* command is used by the manager in setting management variables of the network element.
- *get-response* is the agent response to manager's get-, get-next- and set-request operators. The response includes variable(s) field corresponding the command message.
- *trap* message is sent by the agent to the manager to inform that some event has happened.

And in SNMPv2 are defined two new message:

- *get-bulk-request*, with this message can be fetched larger variable groups. This was developed because get and get-next operators are not very efficient in fetching large amount of data e.g. routing tables.
- *inform-request* is used to change information between managers.

Get/set request and response messages include request ID, error status and error index. The *request ID* is generated by the manager for identification of certain message. The similar identification is used in e.g. DNS messages. The *error status* is used in responses and it specifies error that set/get message caused. Possible errors codes are e.g. too big message, no such variable name exist, wrong value or wrong syntax for variable. The *error index* defines correspondingly what variable was in error.

Variable field(s) (variable-bindings) of the message contain a name and a value. The name field specifies object identifier for accessed variable of the network element and the value field contains information. Variable names and data types of the values are defined in RFCs for MIB and SMI.

*Trap* messages are sent to port 162 by the SNMP agent to inform manager from some event. Trap messages are only informative so the manager does not response to them. The manager can handle these traps or just ignore them. The SNMPv1 trap message header include following fields:

- *Enterprise* field specifies exact type of the network element (vendor and model) with the object identifier defined in the MIB enterprise group.
- *Agent address* is the IP address of the network element, which sent the trap message.
- *Trap type*, in the SNMPv1 are defined e.g. following different type of traps:
    - cold start
    - warm start
    - link down
    - link up
    - authentication failure, message is received from invalid SNMP manager
    - EGP neighbours loss
    - enterprise specific trap
- *Specific code* are used with enterprise type traps
- *Timestamp* is time in hundredths of second (10 ms) since the agent last time initialised.

### *5.4 SMI and MIB*

The *Structure Management Information* (SMI) is a set of common structures and an identification scheme used to reference the variables in the MIB. In the (Figure 9) is presented basic structure of management information defined by ISO and internet SMI structure. The SMI-I define with ASN.1 basic data types for defining MIB variables [Ste94]:

- Integer
- Octet string
- DisplayString, 8 bit bytes from NVT ASCII character set.
- OBJECT IDENTIFIER , which specifies group, simple variable or table identifier of MIB tree
- NULL
- IpAddress, specific data type for IP addresses
- PhyAddress, specific data type for physical addresses
- Counter, SNMPv1 define 32 bit counter (SNMPv2 has also 64 bit counter)
- Gauge, counter which is not wraparound
- TimeTicks is time counter in hundredths of second (10 ms)
- SEQUENCE or SEQUENCE OF are used specify more complex variable structures like tables

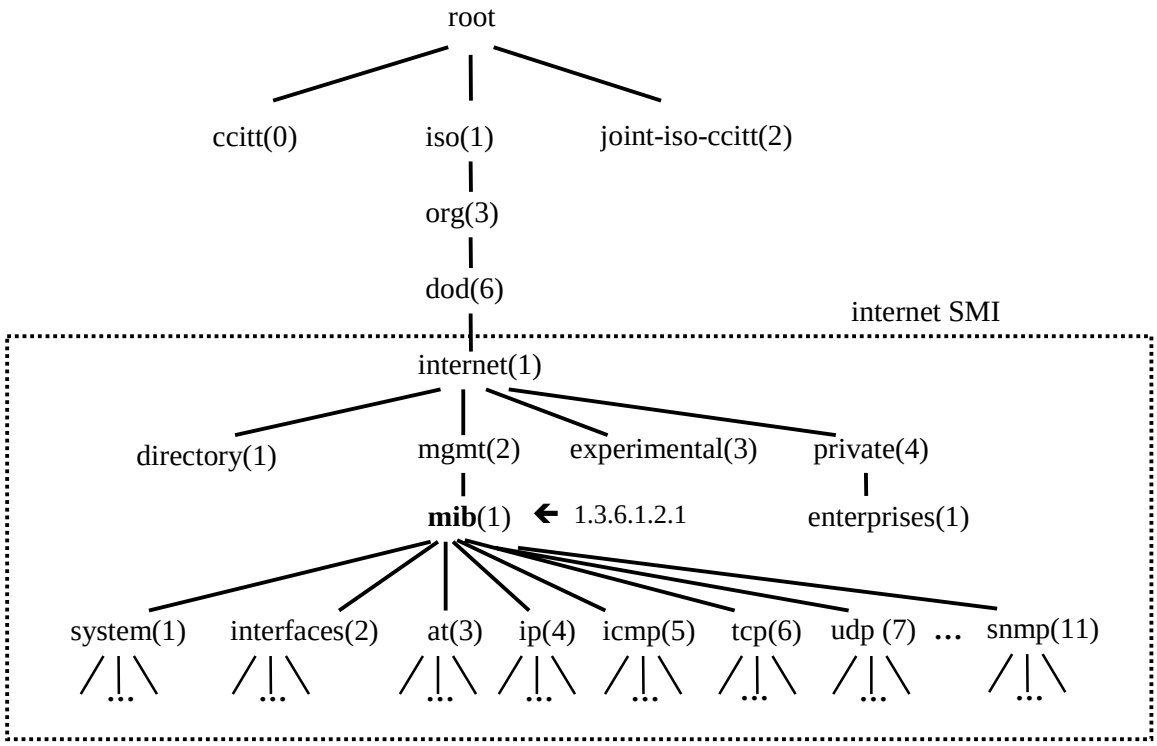


**Figure 9 SMI and MIB-I**

The *Management Information Base* (MIB) defines the exact structure and formats of the management data, which a network element stores and maintains. The MIB is a tree like structure, just like the DNS or UNIX file system. The MIB structure is based on ISO definition of CMIP and CMIS (common management information protocol/services) so it is part of ISO/OSI defined management structure. The MIB can be understood as a virtual database, for real implementation inside network element are not defined or restricted by standards. Often same information is part of several variables, so relational database can be used. In the SMI defined Object identifier is a data type specifying an authoritatively named object and it consist of sequence of integers separated by decimal points. These *object identifier*s specify nodes of the management tree. All variables in the Internet MIB start with object identifier 1.3.6.1.2.1 because of the ISO background. For human readability all nodes have also the textual name, e.g. object identifier 1.3.6.1.2.1.4 have name iso.org.dod.internet.mgmt.mib.ip. The SNMP manager accesses to nodes or leafs of the tree by these object identifiers in variable name field of the SNMP message. [wcun]

With the SNMPv2 is defined new MIBs: *MIB-II* and *manager to manager* MIB. In the MIB-II some groups are extended to have more specific subgroups. The SNMPv2-SMI defines also 64 bits counters. Manager to manager MIB makes possible to use more hierarchical structure for the network management.

Under Internet node there exist *enterprise* group, which contains a vendor specific groups and leaves. This tree structure can include the specific management information for that vendor or its some specific product. Currently there is registered about 3700 different enterprises [fie]. The *Internet Assigned Number Authority* (IANA) handles this enterprise group name registration as normally in the Internet community.

Under Internet node located MIB is divided into the groups named: system, interfaces, at (address translation), ip, icmp, tcp, udp, egp, and snmp. These groups include a simple *variables* and/or *tables*, which correspondingly include variables. In the MIB each variable contain object-type, syntax, access, status and textual description clauses. Also the index clause can be included. The *object type* defines the name of the object identifier. The *syntax* defines the data type of the variable according to the SMI. The *access* right in the SNMPv1 can be read-only or read-write. In the MIB-II the access type is extended to the *max-access* field mainly for the get-bulk-request operator. The *status* can be mandatory or optional (MIB-I). The *index* clause defines table indexing. Variables are referenced with .0 after the variable textual name or the object identifier. The simple variables store normally a counter values or integer type configuration information. For example a simple variable in the MIB-I snmp group can be presented:

```
snmpInGetRequests      OBJECT-TYPE
        SYNTAX         counter
        ACCESS         read-only
        STATUS         mandatory
        DESCRIPTION
                " The total number of SNMP Get-Request PDUs which have been
                  accepted and processed by the SNMP protocol entity"
        : : = { snmp 15}
```

Table entries have one or more indexes depending on the container, e.g. the address translation table is indexed with the internal physical interface number (not address) and net address. In the definition of tables are used SEQUENCE and SEQUENCE OF data types. All entries of the tables are arranged in the lexicographic order. This mean that the manager is accessed tables in the column-row order, so with tables get-next operator has as a response e.g. all physical addresses of the address translation table entries before accessing corresponding net (IP) addresses.

In the Figure 10 is presented the UDP group structure as for example. The UDP group is very simple and it has four simple variables and one table, e.g. the IP group (MIB-I) contains 20 variables and three tables.

The *UDP* group contains a UDP listener table (udpTable) and a counters for transmitted (udpOutDatagrams) and received (udpInDatagrams) datagrams, received datagrams with no application process at the destination port (udpNoPorts) and undeliverable datagrams for some other reason (udpInErrors). The UDP table contains under node entry (udpEntry) UDP address (udpLocalAddress) and port (udpLocalPort). The local address specifies listened address and the local port defines the listened port for that address.

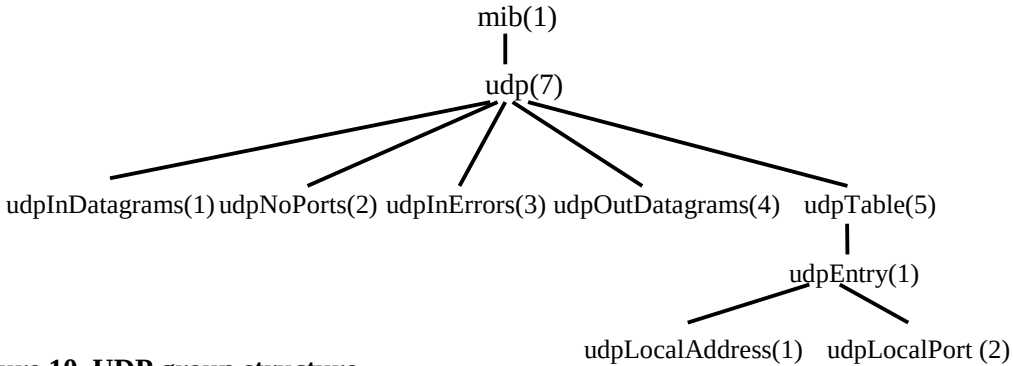


**Figure 10 UDP group structure**

The *system* group (see Figure 9) includes general variables like vendor's ID in enterprise subtree, system textual name, system full domain name, services node provide (OSI model based) and system up time from the last reboot.

The *Interface* group contains for each interface a internal index, interface type (e.g. Ethernet), interface MTU (maximum transmission unit), speed (bits/s), physical address and state of interface (up, down, testing). Various counters are also specified e.g. received and transmitted packets and bytes, error in and discarded packets and number of packets in output queue.

In the MIB-I used *at* group includes address translation tables, where physical address (e.g. Ethernet), net address (IP) and internal index of the interface is mapped. In the MIB-II at-group is divided into several separate groups based on the used protocol.

The *IP* group (in MIB-I used) includes three tables: IP address, IP routing and IP address translation tables. The *IP address table* contains one row for each IP address on the system and includes IP address, interface index, subnet mask and maximum datagram can be reassembled. The *IP routing table* contains e.g. IP destination address, interface index, routing metrics, IP address of next-hop router (gateway) and routing protocol (e.g. OSPF, BGP). The IP group contains also variables for network element IP forwarding state, default TTL (time-to-live) and various counters e.g. received and transmitted datagrams, fragmented and reassembled datagrams and discarded packets due to routing, protocol, fragmenting or reassembling error.

The *ICMP* group contains counters for transmitted and received ICMP messages and errored messages. Also counters for transmitted and received different (11) message types exist.

The *TCP* group includes one table for TCP connections and several protocols related variables and counters. The TCP connection table includes for each connection state (e.g. syn_sent, established or close_wait), local IP address, local TCP port number, remote IP address and remote port number information. Variables contain information of used RTO (retransmission time-out) algorithm, minimum and maximum RTO value and counters e.g. for active and passive connections, transmitted and received segments and errors.

The *EGP* group contain the EGP neighbors table, which include e.g. state, IP address, autonomous system information, polling mode (active or passive), interval for polling (for hello message) and also counters for transmitted and received messages and errors.

The *SNMP* group includes counters for transmitted and received SNMP messages (total and separated by PDU type) and counters for messages with different type of errors.

MIB-II structure is basically the same as MIB-I. However different protocols are normally defined in own subgroups in the address translation.

### *5.5 Security*

In the reality the SNMPv1 does not provide any security option and the SNMPv2 definition has not clear enough security features. For the SNMPv2 are proposed RFCs for security model which provide authentication, privacy (encryption), authorisation and access control (referred also SNMPv2u) but currently RFCs are in the proposed or the draft state and are not widely in use. In the RFC 2271 was defined framework developing of *SNMP version three* [2271], which main intention is to clear the standard situation and to improve the security models of the SNMP. In the user based security model (USM) the authentication is based on the digest authentication protocol (MD5 algorithm) with the private keys. The timestamping

of messages is also provided. The DES (data encryption standard) can optionally be used for encryption [wsc].

It is possible to use the *HTTP* (hypertext transfer protocol) and the *SSL* (secure socket layer) in the transport of SNMP messages. In this case the SNMP network element has to implement basic functionality of the HTTP-server. This method has authentication and ciphered messages, so security is better than with the community based versions.

### *5.6 Implementations*

Many network element manufactures and vendors have however own specific implementation for managing their devices. These are often based on the SNMP and MIB definition, but do not fulfil the entire standard. In the SNMPv1 based systems it is often allowed only to query variables of the network element. The setting and configuration of the network element are done with some other method e.g. also quite an unsure telnet is used in some implementations.

Basic SNMP and MIB with object identifier notation is not very user friendly so it is obvious that there have been developed graphical user interfaces. Quite a popular way to implement SNMP manager is to use the HP OpenView as platform [hpov]. The SNMPv2 makes possible to use any Web browser as the manager interface. In this case agent has to implement basic HTTP server functionality.

## References


[Fei96] Feit, S., TCP/IP, second edition, 1996, McGraw-Hill.

[RFC821] Postel, J.B., Simple Mail Transfer Protocol, Request for comments 821, August 1982

[RFC854] Postel, J., Reynolds, J., TELNET protocol specification, Request for comments 854, May 1983.

[RFC959] Postel, J., Reynolds, J., File Transfer Protocol (FTP), Request for comments: 959, October 1985.

[RFC1350] Sollins, K., The TFTP protocol (revision2), Request for comments 1350, July 1992.

[RFC1425] Klensin, J., et al, SMTP service extensions, Request for comments 1425, 1993

[RFC1521] Borenstein, Freed, Multipurpose Internet Mail Extensions, Request for comments 1521, 1993.

[Ste94] Stevens, R., TCP/IP illustrated volume 1 the protocols, 1994, Addison-Wesley.

[822] D. Crocker. Standard for the Format of ARPA Internet Text Messages. RFC 822, IETF, 1982.

[977] B. Kantor, P Lapsley. Network News Transfer Protocol. RFC 977, IETF, 1986.

[1036] M. Horton, R. Adams. Standard for Interchange of USENET Messages. RFC 1036, IETF, 1986.

[wmc] http://www.magmacom.com/~leisen/master_list.html, Oct. 1998

[INN] http://www.isc.com/inn/, Oct. 1998

[Dia] http://www.backplane.com/diablo/, Oct. 1998

[high] http://www.highwind.com/, Oct. 1998

[cnic] http://cyclone.news.idirect.com/stats/archive/, Oct. 1998

[1155] K. McCloghrie, M. Rose. Structure and Identification of Management Information for TCP/IP-based Internets, RFC 1155, IETF, 1990.

[1157] J. Case et al., A Simple Network Management Protocol, RFC 1155, IETF, 1990

[1213] K. McCloghrie, M. Rose. Management Information Base for Network Management of TCP/IP-based Internets: MIB-II , RFC 1213, IETF, 1991

[1901] J. Case, K. McCloghrie, M. Rose. Introduction to Community-based SNMPv2, RFC 1901, IETF, 1996.

[1910] G. Waters, User-Based Security Model for SNMPv2, RFC 1910, IETF,1996

[2273] P. Presuhn, B. Wijnen. An Architecture for Describing SNMP Management Frameworks, RFC 2271, IETF, 1998.

[wsc] http://www.snmp.com/, Oct. 1998

[wcun] http://wwwsnmp.cs.utwente.nl/, Oct. 1998

[fie] ftp://ftp.isi.edu/in-notes/iana/assignments/enterprise-numbers, Oct. 1998

[Ste96] W.R. Stevens. TCP/IP Illustrated, Volume 3, TCP/IP for Transactions HTTP, NNTP and the UNIX Domain Protocols. Addison-Wesley, 1996.

[hpov] http://www.hp.com/openview/, Oct. 1998