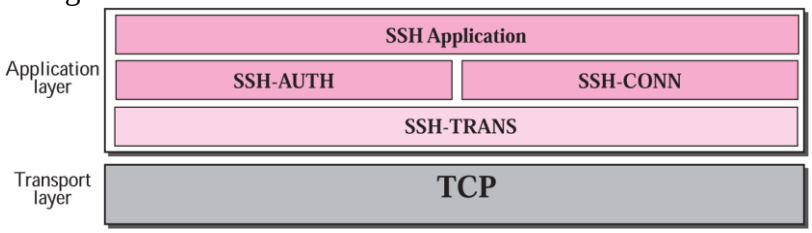
Unit - IV

UNIT – IV: Remote Login

TELNET and SSH- TELNET - Secure shell (SSH) - File transfer - FTP and FTP - FTP - TFTP

SECURE SHELL (SSH)

- ✓ Remote login application program is Secure Shell (SSH). SSH, like TELNET, uses TCP as the underlying transport protocol, but SSH is more secure and provides more services than TELNET.
- ✓ **Versions:**
  - There are two versions of SSH: SSH-1 and SSH-2, which are totally incompatible.
- ✓ **SSH Components:**
  - SSH is a proposed application-layer protocol with four components, as shown in Figure.



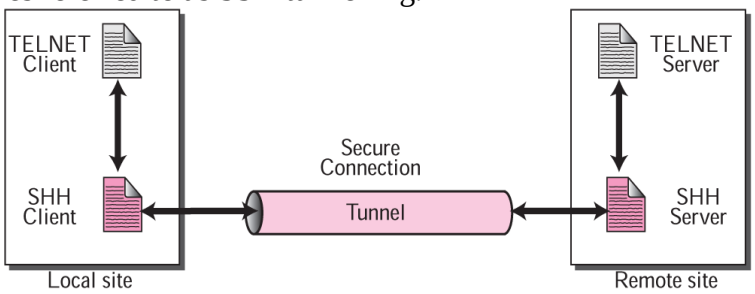
  - **SSH Transport-Layer Protocol (SSH-TRANS):**
    - TCP is not a secured transport layer protocol, SSH first uses a protocol that creates a secured channel on the top of TCP.
    - This new layer is an independent protocol referred to as SSH-TRANS.
    - When the software implementing this protocol is called, the client and server first use the TCP protocol to establish an insecure pro-connection. Then they exchange several security parameters to establish a secure channel on the top of the TCP.
    - Security Services:
      - Privacy or confidentiality of the message exchanged.
      - Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder.
      - Server authentication, which means that the client is now sure that the server is the one that it claims to be.
      - Compression of the messages that improve the efficiency of the system and make attack more difficult.
  - **SSH Authentication Protocol (SSH-AUTH)**
    - After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another software that can authenticate the client for the server.
  - **SSH Connection Protocol (SSH-CONN)**
    - After the secured channel is established and both server and client are authenticated for each other, SHH can call a piece of software that implements the third protocol, SSHCONN.
    - One of the services provided by the SSH-CONN protocol is to do multiplexing. SSH-CONN takes the secure channel established by the

two previous protocols and lets the client create multiple logical channels over it.

- **SSH Applications**
  - After the connection phase is completed, SSH allows several application programs to use the connection.
  - Each application can create a logical channel as described above and then benefit from the secured connection.
  - In other words, remote login is one of the services that can use the SSH-CONN protocols; other applications, such as a file transfer application can use one of the logical channels for this purpose. In the next chapter, we show how SSH can be used for secure file transfer.

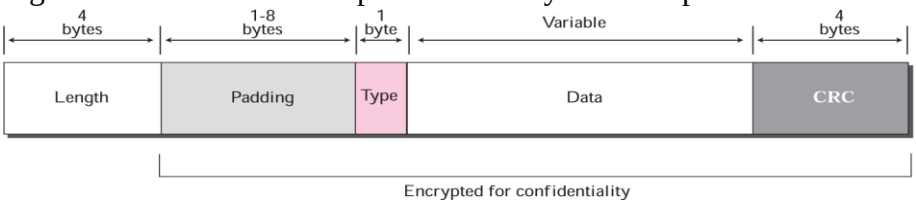
✓ **Port Forwarding:**

- One of the interesting services provided by the SSH protocol is to provide port forwarding.
- SSH port forwarding mechanism creates a tunnel through which the messages belonging to other protocol can travel. For this reason, this mechanism is sometimes referred to as SSH tunnelling.



✓ **Format of the SSH Packets:**

- Figure shows the format of packets used by the SSH protocols.

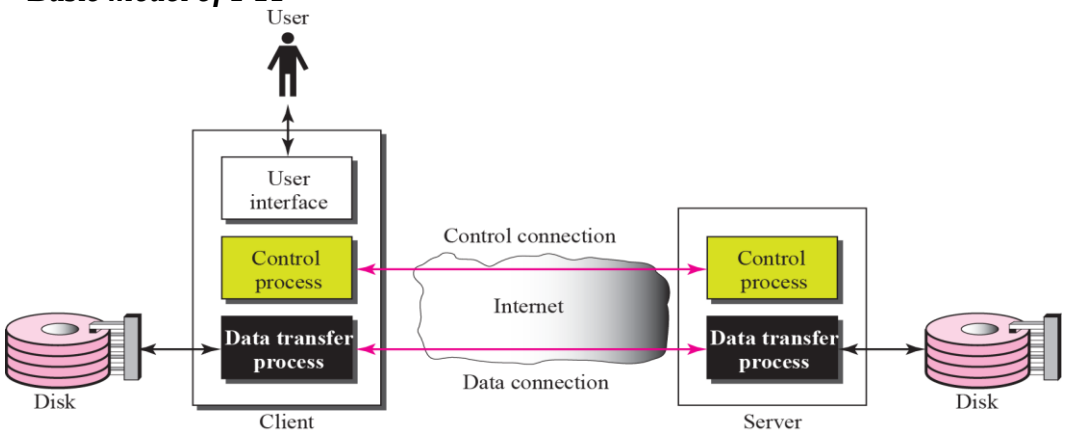


- The following is the brief description of each field:
  - **Length**: This 4-byte field defines the length of the packet including the type, the data, and the CRC field, but not the padding and the length field.
  - **Padding**: One to eight bytes of padding is added to the packet to make the attack on the security provision more difficult.
  - **Type**: This one-byte field defines the type of the packet used by SSH protocols.
  - **Data**: This field is of variable length. The length of the data can be found by deducting the five bytes from the value of the length field.
  - **CRC**: The cyclic redundancy check filed is used for error detection.

FTP

- ✓ File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another.
- ✓ Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first.
- ✓ For example,
  - Two systems may use different file name conventions. Two systems may have different ways to represent text and data.
  - Two systems may have different directory structures.
  - **All of these problems have been solved by FTP in a very simple and elegant approach.**
- ✓ FTP differs from other client-server applications in that it establishes two connections between the hosts.
  - One connection is used for data transfer
  - The other for control information (commands and responses).
- ✓ Separation of commands and data transfer makes FTP more efficient.
  - The control connection uses very simple rules of communication.
  - The data connection, on the other hand, needs more complex rules due to the variety of data types transferred.
- ✓ FTP uses two well-known TCP ports:
  - Port 21 is used for the control connection, and
  - Port 20 is used for the data connection.

- ✓ ***Basic model of FTP***

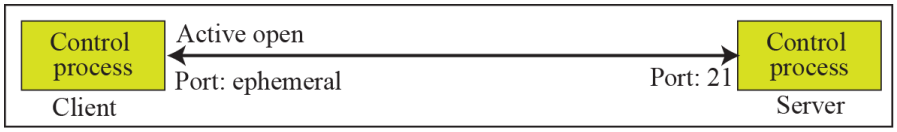


  - The client has three components:
    - User interface, client control process, and the client data transfer process.
  - The server has two components:
    - Server control process and the server data transfer process.
  - The control connection is made between the control processes.
  - The data connection is made between the data transfer processes.
    - The control connection remains connected during the entire interactive FTP session.
    - The data connection is opened and then closed for each file transferred.
- ✓ ***Control Connection***
  - The control connection is created in the same way as other application programs. There are two steps:

- 1. The server issues a passive open on the well-known port 21 and waits for a client.
- 2. The client uses an ephemeral port and issues an active open.

- The connection remains open during the entire process.

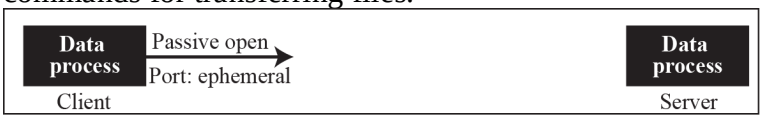
a. First, passive open by server



b. Later, active open by client

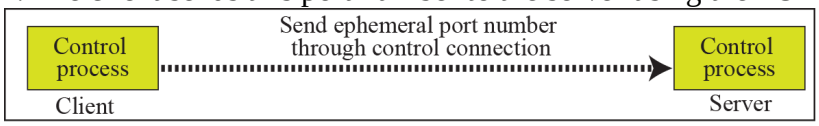
✓ ***Data Connection***

- 1. The client, not the server, issues a passive open using an ephemeral port. This must be done by the client because it is the client that issues the commands for transferring files.



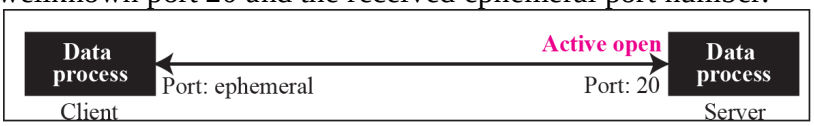
a. First, passive open by client

- 2. The client sends this port number to the server using the PORT command.



b. Second, sending of ephemeral port

- 3. The server receives the port number and issues an active open using the wellknown port 20 and the received ephemeral port number.

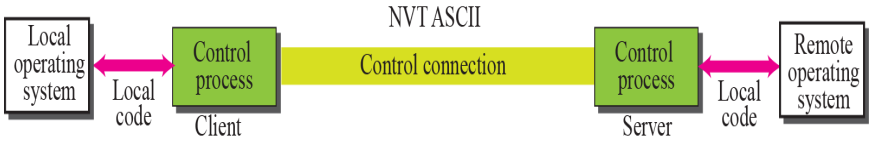


c. Third, active open by server

✓ ***Communication:***

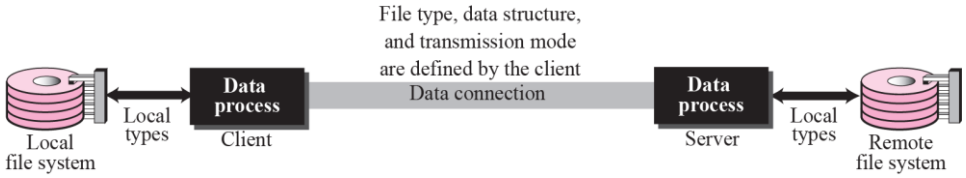
- The FTP client and server, which run on different computers, must communicate with each other.
- These two computers may use different operating systems, different character sets, different file structures, and different file formats.
- FTP must make this heterogeneity compatible.
- FTP has two different approaches, one for the control connection and one for the data connection.

- ***Communication over Control Connection***



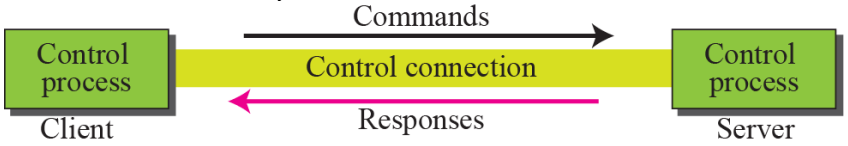
- FTP uses the same approach as TELNET or SMTP to communicate across the control connection.
- It uses the NVT ASCII character set.
- Communication is achieved through commands and responses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each command or response is only one short line so we need not worry about file format or file structure. Each line is terminated with a two-character (carriage return and line feed) end-of-line token.

- ***Communication over Data Connection***



- The purpose and implementation of the data connection are different from that of the control connection.
- It used to transfer files through the data connection.
- The client must define the type of file to be transferred, the structure of the data, and the transmission mode.
- Before sending the file through the data connection, first prepare for transmission through the control connection.
- The heterogeneity problem is resolved by defining three attributes of communication: file type, data structure, and transmission mode.
- File Type:
  - ASCII file, EBCDIC file and Image file
- Data Structure:
  - File structure (default), Record structure and Page structure.
- Transmission Mode:
  - Stream mode, Block mode and compressed mode.

- ***Command processing***
  - FTP uses the control connection to establish a communication between the client control process and the server control process.
  - During this communication, the commands are sent from the client to the server and the responses are sent from the server to the client



✓ ***Commands:***

- Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument.
- Divide the commands into six groups: access commands, file management commands, data formatting commands, port defining commands, file transferring commands, and miscellaneous commands.

- **Access Commands:**

Command	*Argument(s)*	*Description*
USER	User id	User information
PASS	User password	Password
ACCT	Account to be charged	Account information
REIN		Reinitialize
QUIT		Log out of the system
ABOR		Abort the previous command

- **File Manageme nt Commands:**

Command	*Argument(s)*	*Description*
CWD	Directory name	Change to another directory
CDUP		Change to parent directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
NLIST	Directory name	List subdirectories or files without attributes
MKD	Directory name	Create a new directory
PWD		Display name of current directory
RMD	Directory name	Delete a directory
RNFR	File name (old)	Identify a file to be renamed
RNTO	File name (new)	Rename the file
SMNT	File system name	Mount a file system

- **Data Formatting Commands:**

Command	*Argument(s)*	*Description*
TYPE	**A** (ASCII), **E** (EBCDIC), **I** (Image), **N** (Nonprint), or **T** (TELNET)	Define file type
STRU	**F** (File), **R** (Record), or **P** (Page)	Define organization of data
MODE	**S** (Stream), **B** (Block), or **C** (Compressed)	Define transmission mode

- **Port Defining Commands**

Command	*Argument(s)*	*Description*
PORT	6-digit identifier	Client chooses a port
PASV		Server chooses a port

- **File Transferring Commands:**

Command	*Argument(s)*	*Description*
RETR	File name(s)	Retrieve files; file(s) are transferred from server to client
STOR	File name(s)	Store files; file(s) are transferred from client to server
APPE	File name(s)	Similar to STOR, but if file exists, data must be appended to it
STOU	File name(s)	Same as STOR, but file name will be unique in the directory
ALLO	File name(s)	Allocate storage space for files at the server
REST	File name(s)	Position file marker at a specified data point
STAT	File name(s)	Return status of files

- **Miscellaneous Commands:**

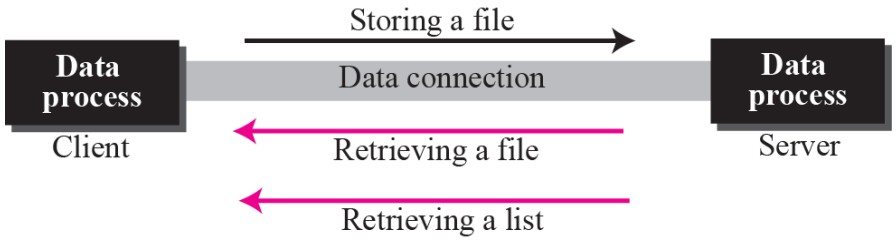
Command	*Argument(s)*	*Description*
HELP		Ask information about the server
NOOP		Check if server is alive
SITE	Commands	Specify the site-specific commands
SYST		Ask about operating system used by the server

✓ ***Responses:***

Code	*Description*
Positive Preliminary Reply	
120	Service will be ready shortly
125	Data connection open; data transfer will start shortly
150	File status is OK; data connection will be open shortly
Positive Completion Reply	
200	Command OK
211	System status or help reply
212	Directory status
213	File status
214	Help message
215	Naming the system type (operating system)
220	Service ready
221	Service closing
225	Data connection open
226	Closing data connection
227	Entering passive mode; server sends its IP address and port number
230	User login OK
250	Request file action OK
Positive Intermediate Reply	
331	User name OK; password is needed
332	Need account for logging
350	The file action is pending; more information needed

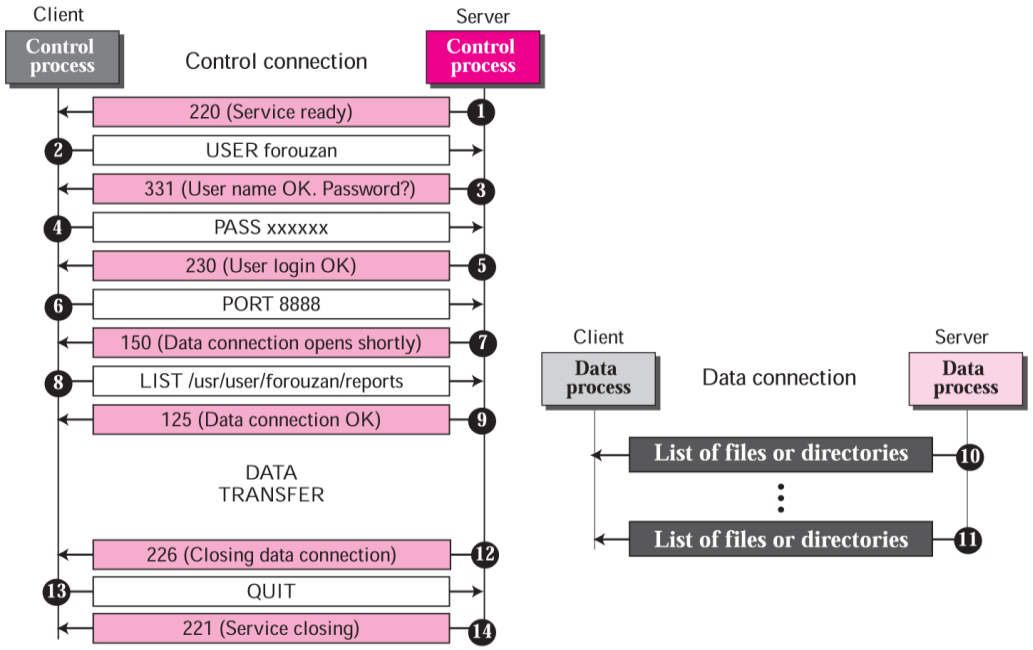
Transient Negative Completion Reply	
425	Cannot open data connection
426	Connection closed; transfer aborted
450	File action not taken; file not available
451	Action aborted; local error
452	Action aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
530	User not logged in
532	Need account for storing file
550	Action is not done; file unavailable
552	Requested action aborted; exceeded storage allocation
553	Requested action not taken; file name not allowed

✓ ***File transfer:***



- File transfer occurs over the data connection under the control of the commands sent over the control connection. However, we should remember that file transfer in FTP means one of three things:
  - A file is to be copied from the server to the client (download). This is called retrieving a file. It is done under the supervision of the RETR command.
  - A file is to be copied from the client to the server (upload). This is called storing a file. It is done under the supervision of the STOR command.
  - A list of directory or file names is to be sent from the server to the client. This is done under the supervision of the LIST command. Note that FTP treats a list of directory or file names as a file. It is sent over the data connection.

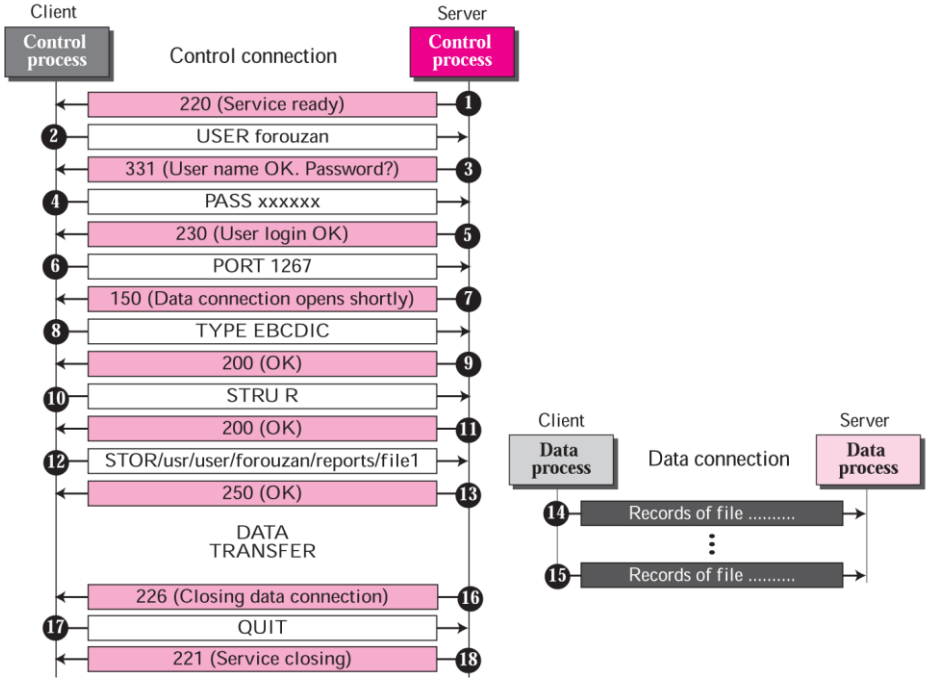
✓ ***Figure shows an example of using FTP for retrieving a list of items in a directory:***



1. After the control connection to port 21 is created, the FTP server sends the 220 (service ready) response on the control connection.
2. The client sends the USER command.
3. The server responds with 331 (user name is OK, password is required).
4. The client sends the PASS command.
5. The server responds with 230 (user login is OK).

6. The client issues a passive open on an ephemeral port for the data connection and sends the PORT command (over the control connection) to give this port number to the server.
7. The server does not open the connection at this time, but it prepares itself for issuing an active open on the data connection between port 20 (server side) and the ephemeral port received from the client. It sends response 150 (data connection will open shortly).
8. The client sends the LIST message.
9. Now the server responds with 125 and opens the data connection.
10. The server then sends the list of the files or directories (as a file) on the data connection. When the whole list (file) is sent, the server responds with 226 (closing data connection) over the control connection.
11. The client now has two choices. It can use the QUIT command to request the closing of the control connection or it can send another command to start another activity (and eventually open another data connection). In our example, the client sends a QUIT command.
12. After receiving the QUIT command, the server responds with 221 (service closing) and then closes the control connection.

✓ ***Figure shows an example of how an image (binary) file is stored:***



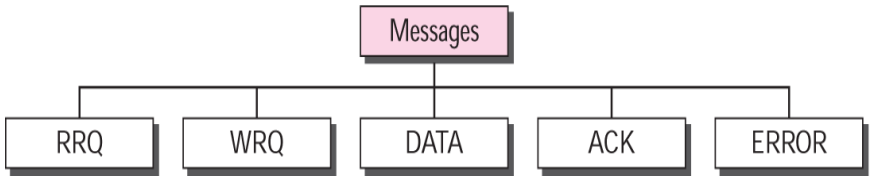
1. After the control connection to port 21 is created, the FTP server sends the 220 (service ready) response on the control connection.
2. The client sends the USER command.
3. The server responds with 331 (user name is OK, a password is required).
4. The client sends the PASS command.
5. The server responds with 230 (user login is OK).
6. The client issues a passive open on an ephemeral port for the data connection and sends the PORT command (over the control connection) to give this port number to the server.
7. The server does not open the connection at this time, but prepares itself for issuing an active open on the data connection between port 20 (server side)

and the ephemeral port received from the client. It sends the response 150 (data connection will open shortly).

8. The client sends the TYPE command.
9. The server responds with the response 200 (command OK).
10. The client sends the STRU command.
11. The server responds with 200 (command OK).
12. The client sends the STOR command.
13. The server opens the data connection and sends the response 250.
14. The client sends the file on the data connection. After the entire file is sent, the data connection is closed. Closing the data connection means end-of-file.
15. The server sends the response 226 on the control connection.
16. The client sends the QUIT command or uses other commands to open another data connection for transferring another file. In our example, the QUIT command is sent.
17. The server responds with 221 (service closing) and it closes the control connection.

TFTP (Trivial File Transfer Protocol)

- ✓ TFTP, or Trivial File Transfer Protocol, is a simple high-level protocol for transferring data servers use to boot diskless workstations, X-terminals, and routers by using User Data Protocol (UDP).
- ✓ TFTP can read or write a file for the client. Reading means copying a file from the server site to the client site. Writing means copying a file from the client site to the server site.
- ✓ TFTP uses the services of UDP on the well-known port 69.
- ✓ **Messages**:
  - There are five types of TFTP messages: RRQ, WRQ, DATA, ACK, and ERROR.



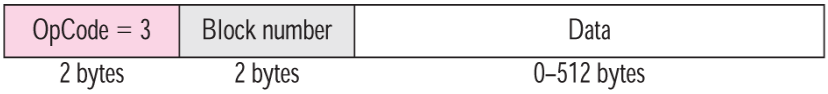
  - **RRQ**
    - The read request (RRQ) message is used by the client to establish a connection for reading data from the server.
    - RRQ format:

    - The RRQ message fields are as follows:
    - **OpCode:** The first field is a 2-byte operation code. The value is 1 for the RRQ message.
    - **File name:** The next field is a variable-size string (encoded in ASCII) that defines the name of the file. Since the file name varies in length, termination is signaled by a 1-byte field of 0s.
    - **Mode:** The next field is another variable-size string defining the transfer mode. The mode field is terminated by another 1-byte field of

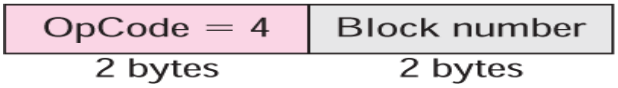
0s. The mode can be one of two strings: "netascii" (for an ASCII file) or "octet" (for a binary file).

- **WRQ**
  - The write request (WRQ) message is used by the client to establish a connection for writing data to the server.
  - The format is the same as RRQ except that the OpCode is 2.

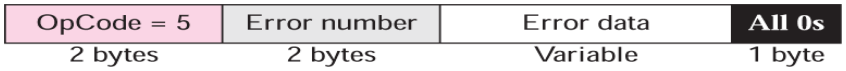
- **DATA**
  - The data (DATA) message is used by the client or the server to send blocks of data.
  - DATA format:



  - **Block number:** This is a 2-byte field containing the block number. The sender of the data (client or server) uses this field for sequencing. All blocks are numbered sequentially starting with 1.
  - **Data**. This block must be exactly 512 bytes in all DATA messages except the last block, which must be between 0 and 511 bytes. A non-512 byte block is used as a signal that the sender has sent all the data. In other words, it is used as an end-of-file indicator. The sender must send one extra block of zero bytes to show the end of transmission.
- **ACK**
  - The acknowledge (ACK) message is used by the client or server to acknowledge the receipt of a data block. The message is only 4 bytes long.
  - ACK format:



    - **OpCode :** The first field is a 2-byte operation code. The value is 4 for the ACK message.
    - **Block number:** The next field is a 2-byte field containing the number of the block received.
  - The ACK message can also be a response to a WRQ. It is sent by the server to indicate that it is ready to receive data from the client. In this case the value of the block number field is 0.
- **ERROR**
  - The ERROR message is used by the client or the server when a connection cannot be established or when there is a problem during data transmission.
  - It can be sent as a negative response to RRQ or WRQ.
  - It can also be used if the next block cannot be transferred during the actual data transfer phase.
  - The error message is not used to declare a damaged or duplicated message.



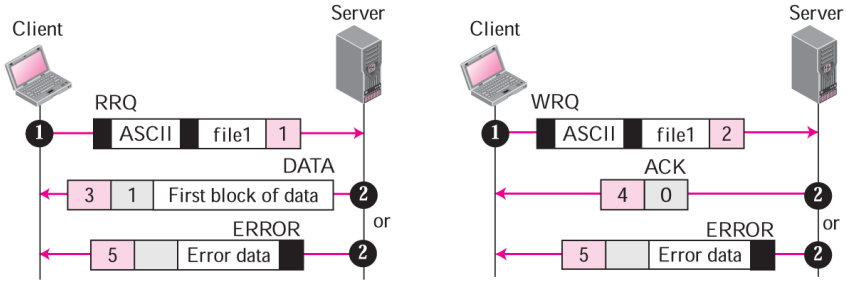
    - **OpCode**: The first field is a 2-byte operation code. The value is 5 for the ERROR message.

- **Error number:** This 2-byte field defines the type of error. Table shows the error numbers and their corresponding meanings.

Number	*Meaning*	*Number*	*Meaning*
0	Not defined	5	Unknown port number
1	File not found	6	File already exists
2	Access violation	7	No such user
3	Disk full or quota exceeded		
4	Illegal operation		

✓ **Connection**

- TFTP uses RRQ, WRQ, ACK, and ERROR messages to establish connection.
- It uses the DATA message with a block of data of fewer than 512 bytes (0–511) to terminate connection.
- **Connection Establishment**
  - Connection establishment for reading files is different from connection establishment for writing files.



  - **Reading**: To establish a connection for reading, the TFTP client sends the RRQ message. The name of the file and the transmission mode is defined in this message. If the server can transfer the file, it responds positively with a DATA message containing the first block of data. If there is a problem, such as difficulty in opening the file or permission restriction, the server responds negatively by sending an ERROR message.
  - **Writing**: To establish a connection for writing, the TFTP client uses the WRQ message. The name of the file and the transmission mode is defined in this message. If the server can accept a copy of the file, it responds positively with an ACK message using a value of 0 for the block number. If there is any problem, the server responds negatively by sending an ERROR message.
- **Connection Termination:**
  - After the entire file is transferred, the connection must be terminated.
  - TFTP does not have a special message for termination. Termination is accomplished by sending the last block of data, which is less than 512 bytes.

✓ **Data Transfer**

- The data transfer phase occurs between connection establishment and termination.
- TFTP uses the services of UDP, which is unreliable.
- The file is divided into blocks of data, in which each block except the last one is exactly 512 bytes. The last block must be between 0 and 511 bytes. TFTP can transfer data in ASCII or binary format.

- UDP does not have any mechanism for flow and error control. TFTP has to create a flow- and error-control mechanism to transfer a file made of continuous blocks of data.

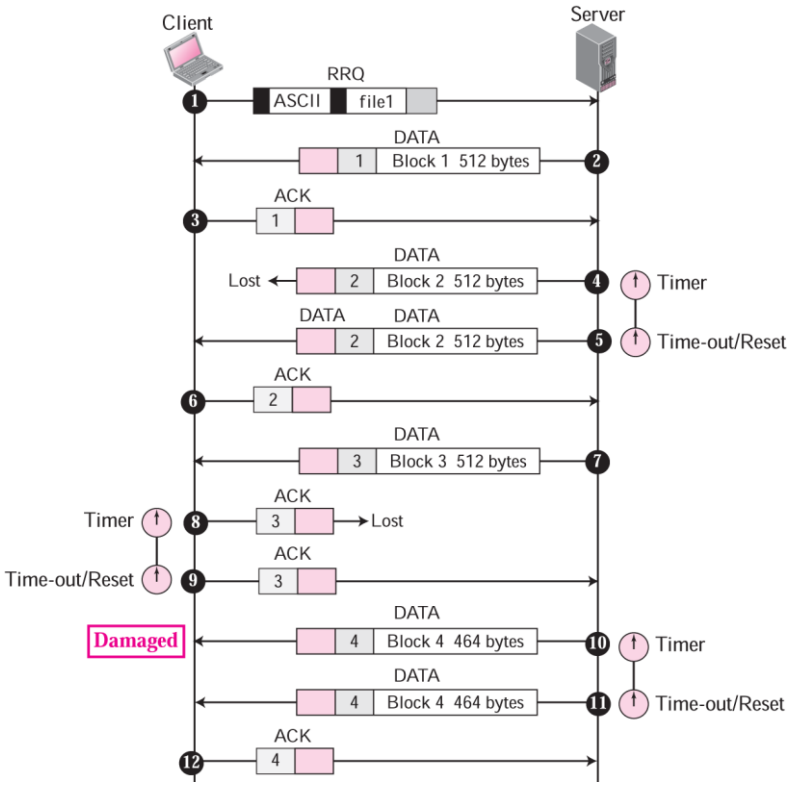
  - **Flow Control:** TFTP sends a block of data using the DATA message and waits for an ACK message. If the sender receives an acknowledgment before the time-out, it sends the next block.

    - **Retrieve a File:** When the client wants to retrieve (read) a file, it sends the RRQ message. The server responds with a DATA message sending the first block of data with a block number of 1.

    - **Store a File**: When the client wants to store (write) a file, it sends the WRQ message. The server responds with an ACK message (if there is no problem) using 0 for the block number. After receiving this acknowledgment, the client sends the first data block with a block number of 1.

  - **Error Control**: The TFTP error-control mechanism is different from those of other protocols. It is symmetric, which means that the sender and the receiver both use time-outs. The sender uses a time-out for data messages; the receiver uses a time-out for acknowledgment messages. If a data message is lost, the sender retransmits it after time-out expiration. If an acknowledgment is lost, the receiver retransmits it after time-out expiration. This guarantees a smooth operation.
    - **Damaged Message:** There is no negative acknowledgment. If a block of data is damaged, it is detected by the receiver and the block is discarded. The sender waits for the acknowledgment and does not receive it within the time-out period. The block is then sent again.
    - **Lost Message:** If a block is lost, it never reaches the receiver and no acknowledgment is sent. The sender resends the block after the time-out.
    - **Lost Acknowledgment:** If an acknowledgment is lost, we can have two situations. If the timer of the receiver matures before the timer of the sender, the receiver retransmits the acknowledgment; otherwise, the sender retransmits the data.
    - **Duplicate Message**: Duplication of blocks can be detected by the receiver through block number. If a block is duplicated, it is simply discarded by the receiver.

✓ **<u>TFTP Example</u>**


Client
Server
RRQ
ASCII
file1
DATA
1
Block 1 512 bytes
ACK
1
DATA
Lost
2
Block 2 512 bytes
Timer
DATA
DATA
2
Block 2 512 bytes
Time-out/Reset
ACK
2
DATA
3
Block 3 512 bytes
ACK
Timer
3
Lost
Time-out/Reset
ACK
3
DATA
Damaged
4
Block 4 464 bytes
Timer
DATA
4
Block 4 464 bytes
Time-out/Reset
ACK
4