FTP e TFTP

File transfer protocols

On-Line File Sharing

- Always a popular application
- Two basic paradigms
  - Whole-file copying
  - Piecewise file access
- Piecewise access mechanism
  - Opaque: application uses special facilities to access remote file
  - Transparent: application uses same facilities to access local and remote files

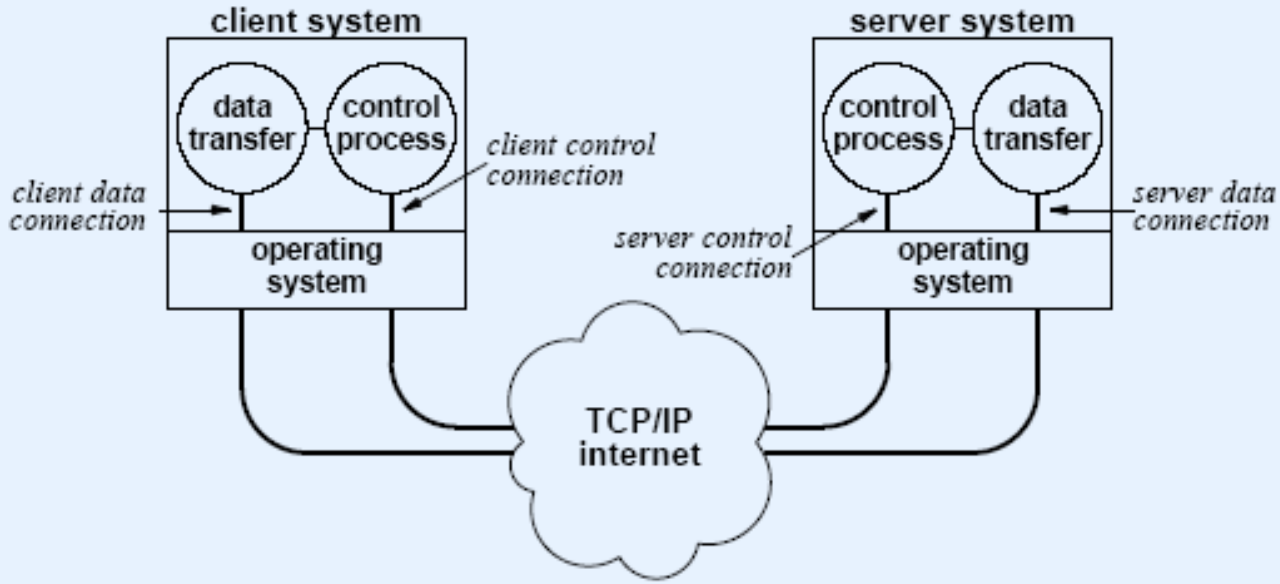
File Transfer

- Whole file copying
- Client
  - Contacts server
  - Specifies file
  - Specifies transfer direction
- Server
  - Maintains set of files on local disk
  - Waits for contact
  - Honors request from client

File Transfer Protocol (FTP)

- Major TCP/IP protocol for whole-file copying
- Uses TCP for transport
- Features
  - Interactive access
  - Format specification (ASCII or EBCDIC)
  - Authentication control (login and password)

FTP Process Model



- Separate processes handle
  - Interaction with user
  - Individual transfer requests

FTP's Use of TCP Connections

Data transfer connections and the data transfer processes that use them can be created dynamically when needed, but the control connection persists throughout a session. Once the control connection disappears, the session is terminated and the software at both ends terminates all data transfer processes.

Control Connection Vs. Data Connection

- For data transfer, client side becomes server and server side becomes client
- Client
  - Creates process to handle data transfer
  - Allocates port and sends number to server over control connection
  - Process waits for contact
- Server
  - Receives request
  - Creates process to handle data transfer
  - Process contacts client-side

Interactive Use Of FTP

- Initially a command-line interface
  - User invokes client and specifies remote server
  - User logs in and enters password
  - User issues series of requests
  - User closes connection
- Currently
  - Most FTP initiated through browser
  - User enters URL or clicks on link
  - Browser uses FTP to contact remote server and obtain list of files
  - User selects file for download

Interactive FTP Commands

!	cr	macdef	proxy	send
$	delete	mdelete	sendport	status
account	debug	mdir	put	struct
append	dir	mget	pwd	sunique
ascii	disconnect	mkdir	quit	tenex
bell	form	mls	quote	trace
binary	get	mode	recv	type
bye	glob	mput	remotehelp	user
case	hash	nmap	rename	verbose
cd	help	ntrans	reset	?
cdup	lcd	open	rmdir	
close	ls	prompt	runique	

Anonymous FTP

- Login *anonymous*
- Password *guest*
- Used for ‘‘open’’ FTP site (where all files are publicly available

Firewall problems with FTP

- Client-side Firewalls
  - the client is behind a firewall and cannot be reached directly from the outside
- NATs
  - the client is behind a NAT device and cannot be reached from the outside

Client-side Firewalls: Solution 1

- **Solution 1**: The **client user** should configure their FTP client program to use PASV rather than PORT. (Using passive mode may not solve the problem if there is a similar restrictive firewall on the server side.)

The Two Types of Data Transfers - *Active* (PORT) and *Passive* (PASV)

- The client program can specify ***active*** mode by sending the "PORT" command to instruct that the server should connect back to a specified IP address and port number and then send the data.
- Or, a client program can choose ***passive*** mode by using the "PASV" command to ask that the server tell the client an IP address and port number that the client can connect to and receive the data.

The Two Types of Data Transfers - *Active* (PORT) and *Passive* (PASV)

- In a nutshell, PORT is used to have the server connect to the client, and PASV is used to have the client connect to the server. Since the client connects to the server to establish the control connection, it would seem logical that the client should connect to the server to establish the data connection, which would imply that PASV would be preferred (and at the same time eliminate the single biggest problem with FTP and firewalls).
- Mysteriously, the implementers chose to specify in the FTP specification that PORT should be preferred and PASV need not be implemented at all by FTP client programs.

Example Sessions Using Active Data Transfers

- Client: USER anonymous
- Server: 331 Guest login ok, send your e-mail address as password.
- Client: PASS NcFTP@
- Server: 230 Logged in anonymously.
- Client: PORT 192,168,1,2,7,138 The client wants the server to send to port number 1930 on IP address 192.168.1.2.
- Server: 200 PORT command successful.
- Client: LIST
- Server: 150 Opening ASCII mode data connection for /bin/ls. The server now connects out from port 21 to port 1930 on 192.168.1.2.
- Server: 226 Listing completed. That succeeded, so the data is now sent over the established data connection.
- Client: QUIT
- Server: 221 Goodbye.

Example Sessions Using Passive Data Transfers

- Client: USER anonymous
- Server: 331 Guest login ok, send your e-mail address as password.
- Client: PASS NcFTP@
- Server: 230 Logged in anonymously.
- Client: PASV The client is asking where he should connect.
- Server: 227 Entering Passive Mode (172,16,3,4,204,173) The server replies with port 52397 on IP address 172.16.3.4.
- Client: LIST
- Server: 150 Data connection accepted from 172.16.3.4:52397; transfer starting. The client has now connected to the server at port 52397 on IP address 172.16.3.4.
- Server: 226 Listing completed. That succeeded, so the data is now sent over the established data connection.
- Client: QUIT
- Server: 221 Goodbye.

Client-side Firewalls: Solution 2

- **Solution 2**: A better solution is for the **network administrator of the client network** to use high-quality network address translation software. Devices can keep track of FTP data connections, and when a client on a private network uses "PORT" with an internal network address, the device should dynamically rewrite the packet containing the PORT and IP address and change the address so that it refers to the external IP address of the routing device.

Solution 2, continued

- The device would then have to route the connection incoming from the remote FTP server back to the internal network address of the client. I.e., from the example above we had:
- **Client:** PORT 192,168,1,2,7,138

- When the packet containing this PORT reaches the routing device, it should be rewritten like this, assuming the external address is 17.254.0.26:
- **Client:** PORT 17,254,0,26,7,138
- The remote server would then attempt to connect to 17.254.0.26:1930. The routing device in this example would then forward all traffic for this connection to and from the client address at 192.168.1.2:1930.

More info

- http://www.ncftp.com/ncftpd/doc/misc/ftp_and_firewalls.html

TFTP – Trivial FTP

Trivial File Transfer Protocol (TFTP)

- Alternative to FTP
- Whole-file copying
- Not as much functionality as FTP
- Code is much smaller
- Intended for use on Local Area Network
- Runs over UDP
- Diskless machine can use to obtain image at bootstrap

TFTP Retransmission

- Symmetric (both sides implement timeout and retransmission)
- Data block is request for ACK
- ACK is request for next data block