TFTP & FTP

CSE 156

slides are modified from **Dave Hollinger** and **Michael mgunes**

Overview

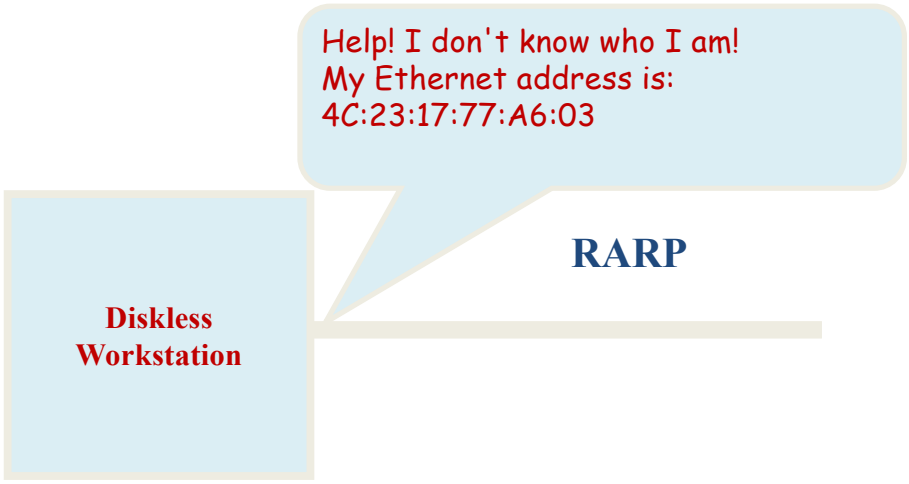
- Trivial File Transfer Protocol (RFC 1350)
  - TFTP and TFTP's message formats
- File Transfer Protocol (RFC 959)
  - Why FTP?
  - FTP's connections
  - FTP in action
  - FTP commands/responses
- TFTP and FTP compared
- Worth exploring as foundational protocols of the Internet
- These protocols are still actively used today

Trivial FTP (TFTP)

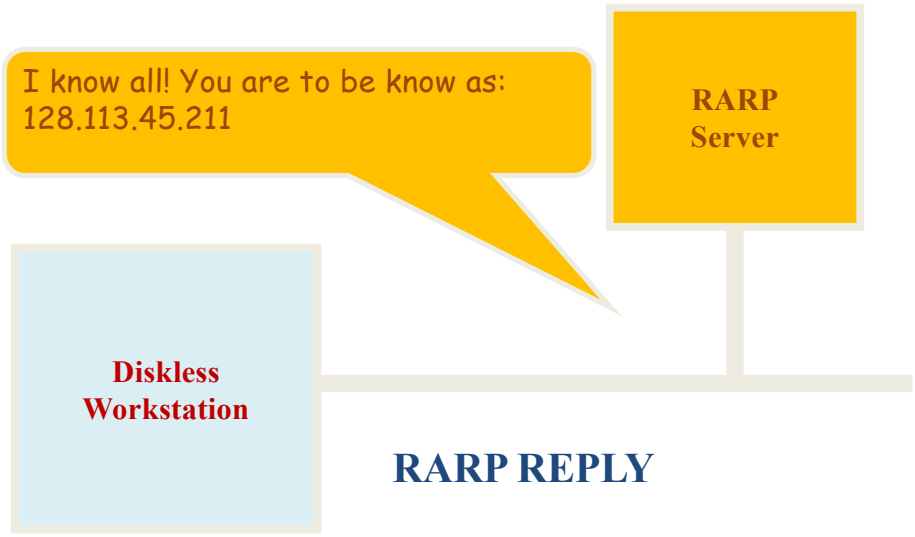
- Revision 2 defined in RFC 1350
- Used only to read and write files from/to a remote server
  - Cannot list directories etc
- Useful for bootstrapping diskless systems
  - PCs, VMs, workstations, X terminals
- Simple and small:
  - 5 message formats
  - Runs on UDP
  - Designed to fit in ROM
  - Uses a "stop and wait" protocol
  - No built-in security features (login)

Diskless Workstation Boot Up

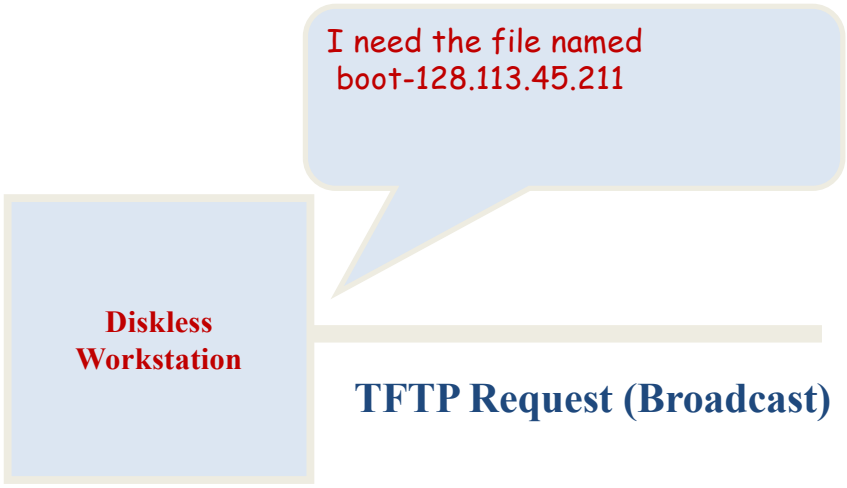
The call for help



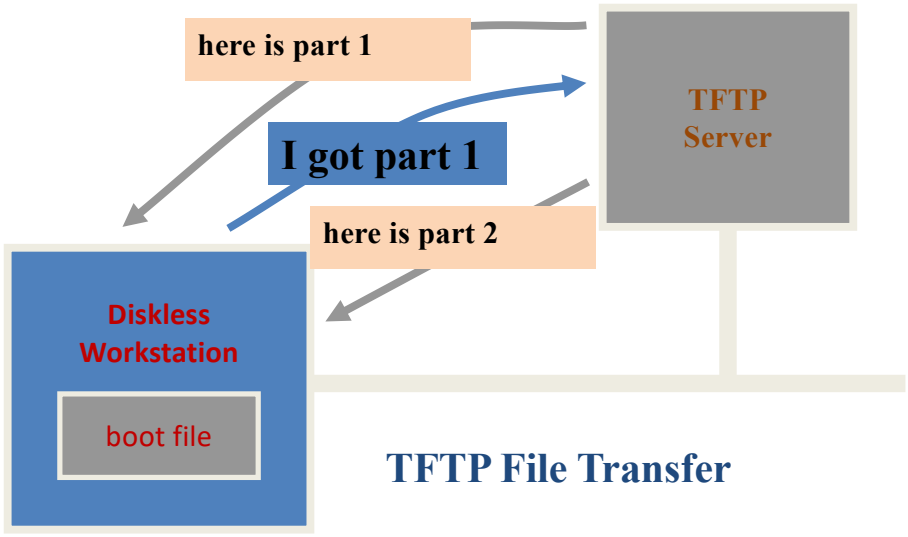
The answer from the all-knowing



The request for instructions



The dialog



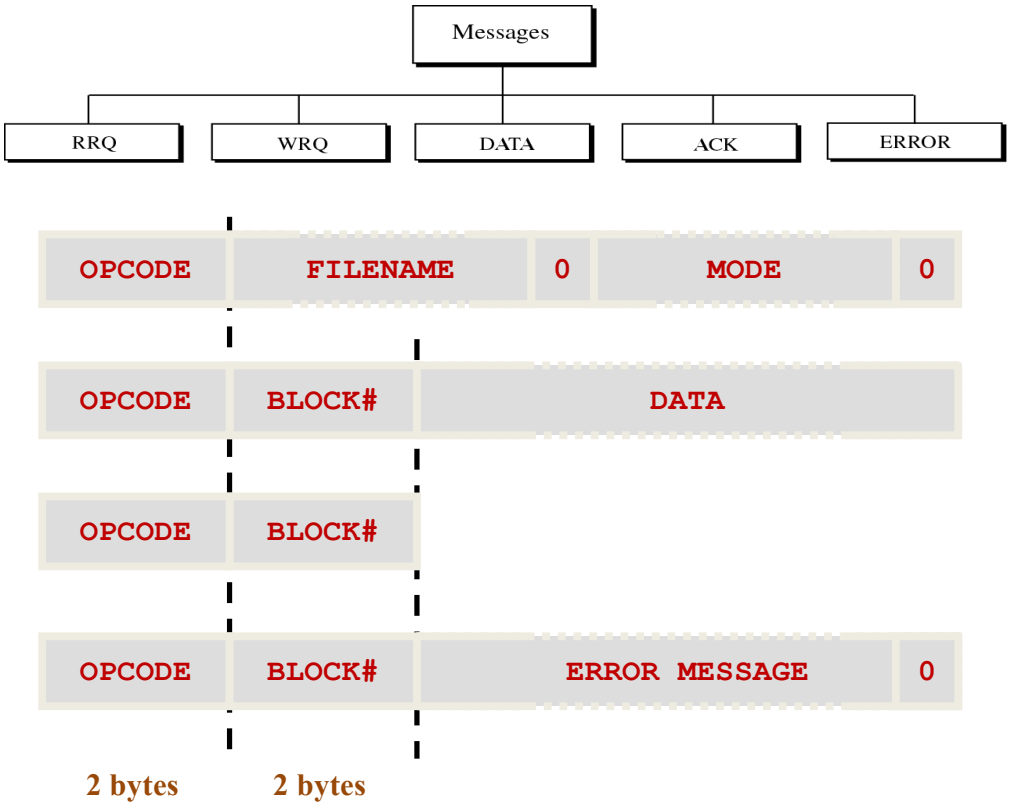
TFTP Protocol

5 message types:

- Read request
- Write request
- Data
- ACK (acknowledgment)
- Error

- Each is an independent UDP Datagram
- Each has a 2-byte opcode (1st 2 bytes)
- The structure of the rest of the datagram depends on the opcode

TFTP Message Formats



TFTP Transfer Modes

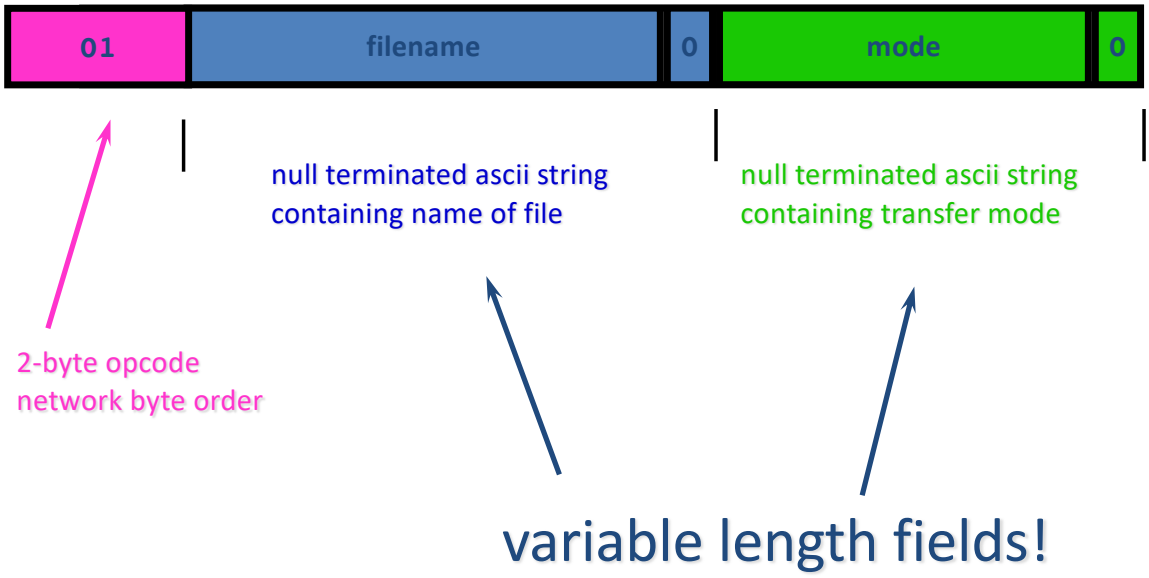
- **octet**: for transferring binary files
  - no translation done
- **netascii**: for transferring text files
  - all lines end with \r\n (CR,LF).
  - provides standard format for transferring text files
  - both ends responsible for converting to/from netascii format

NetAscii Transfer Mode

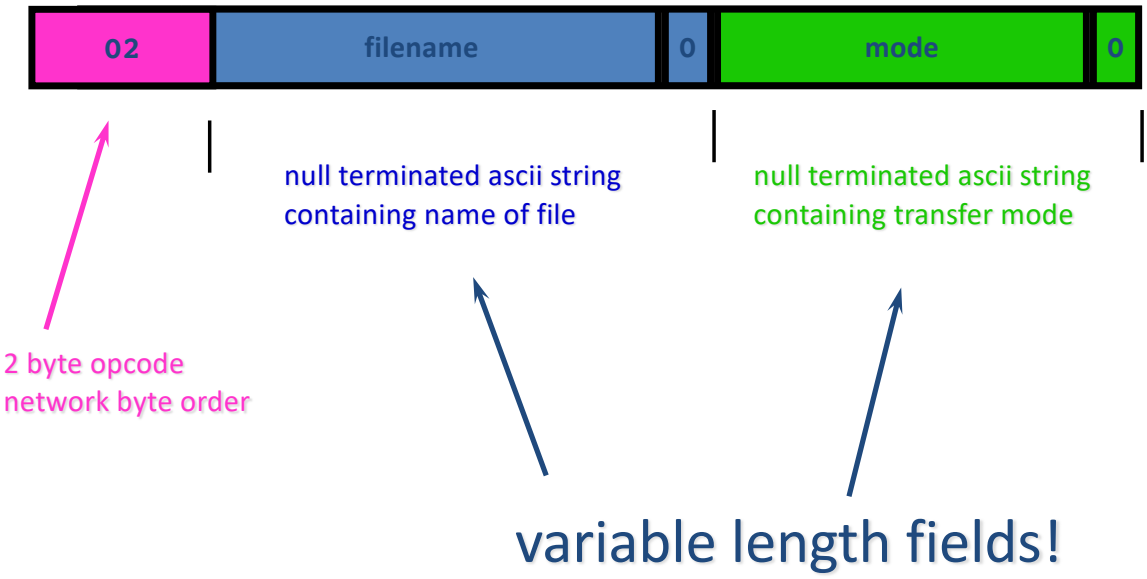
Unix - end of line marker is just '\n'

- Receiving a file
  - Need to remove '\r' before storing data.
- Sending a file
  - Need to replace every '\n' with "\r\n" before sending

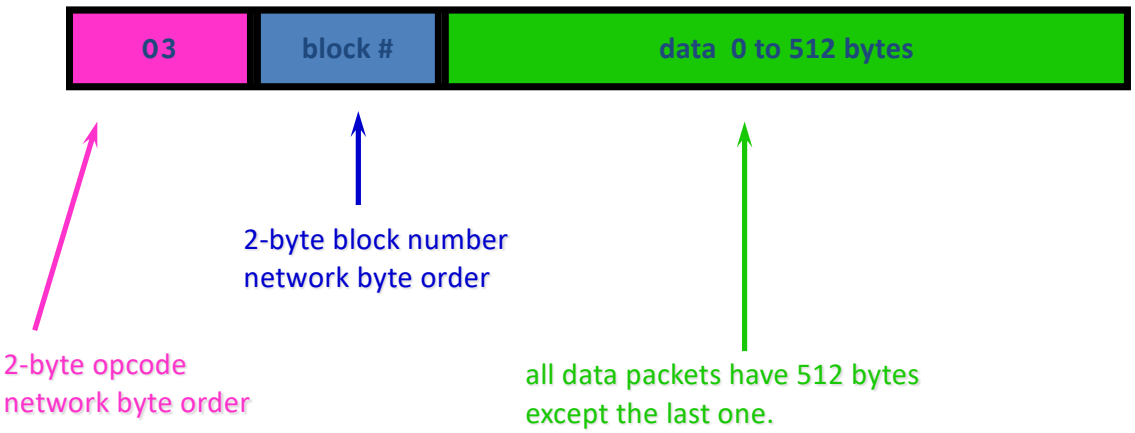
Read Request



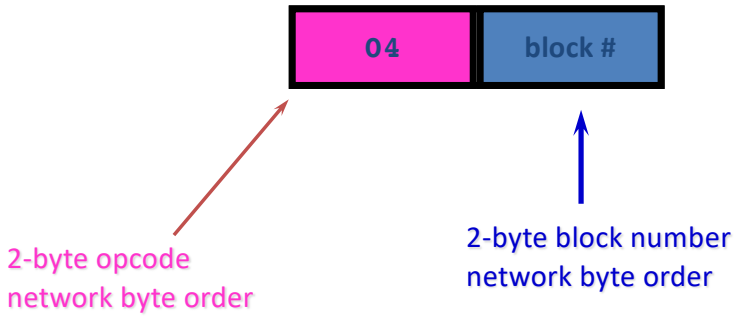
Write Request



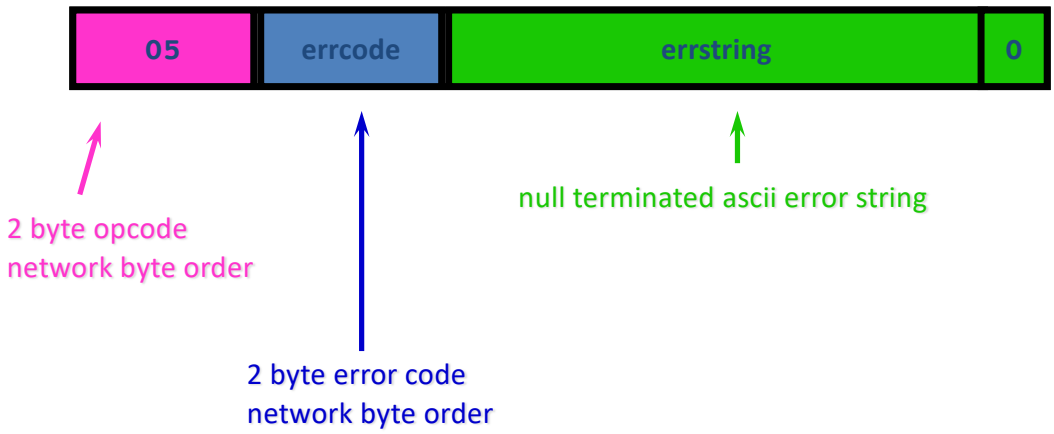
TFTP Data Packet



TFTP Acknowledgment



TFTP Error Packet

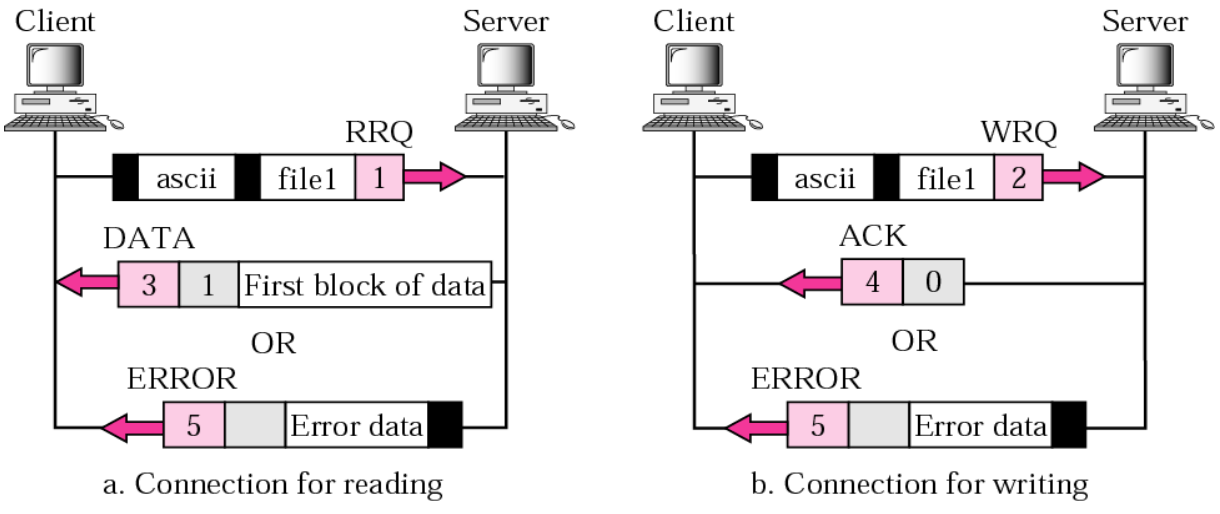


Not retransmitted or acked

TFTP Error Codes (16-bit)

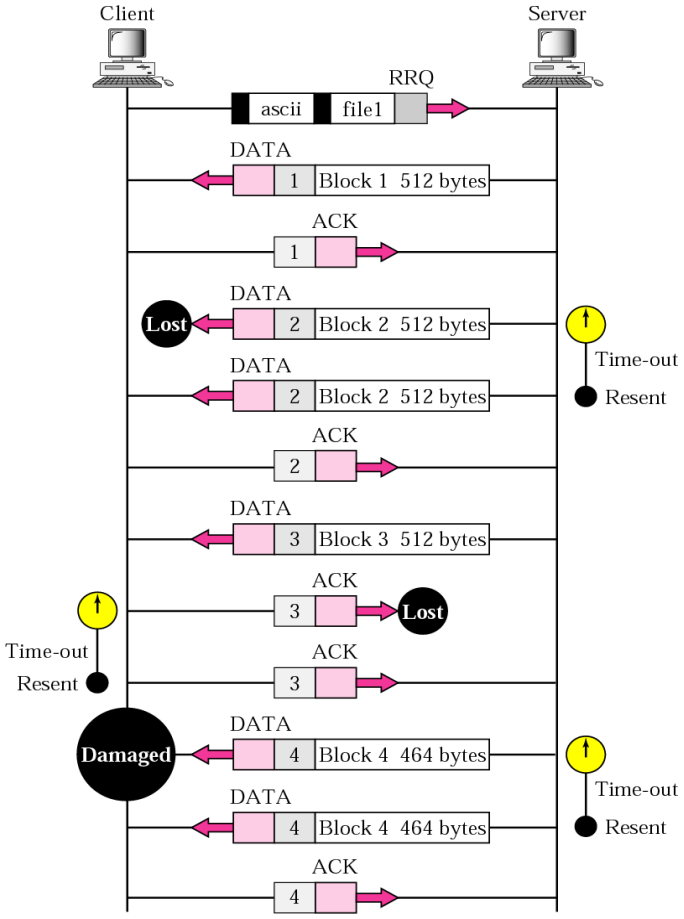
0 - Not defined
1 - File not found
2 - Access violation
3 - Disk full
4 - Illegal TFTP operation
5 - Unknown port
6 - File already exists
7 - No such user

TFTP Connection Establishment



a. Connection for reading

b. Connection for writing



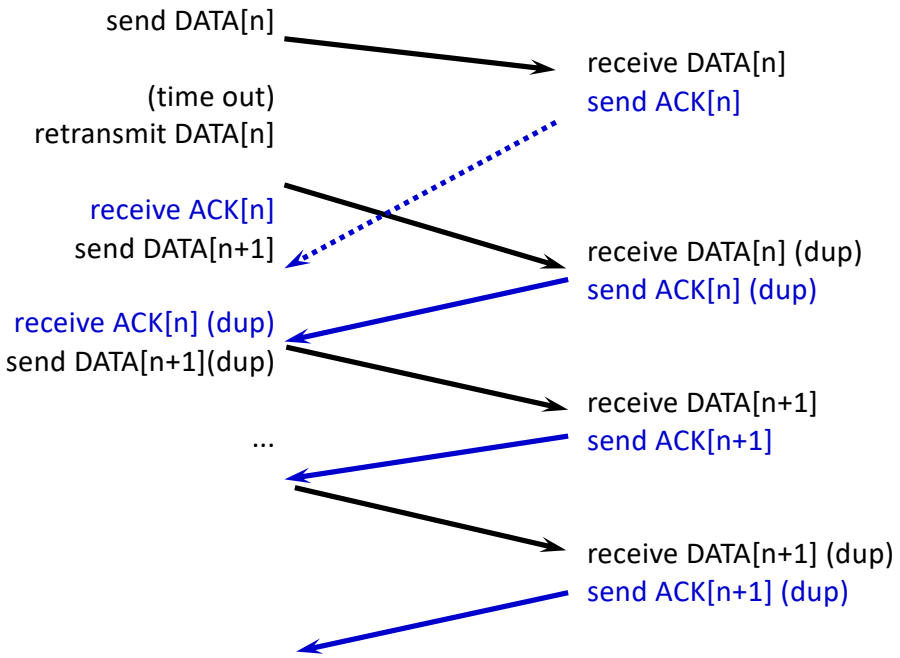
TFTP

Lost Data Packets

Original Protocol Specification

- Sender uses a timeout with retransmission
  - Sender could be client or server
- Duplicate data packets must be recognized, and ACK retransmitted
- This original protocol suffers from the "sorcerer's apprentice syndrome"

Sorcerer's Apprentice Syndrome



The Fix

- Sender should not resend a data packet in response to a duplicate ACK
- If sender receives ACK[n]
  - Don't send DATA[n+1] if the ACK was a duplicate

Concurrency

- TFTP servers use a "well-known" service, i.e., UDP port number 69
- How would you implement a concurrent server?
  - Fork/thread can be used
  - Can also be done without forking/threading, but it requires some bookkeeping
  - Need to deal with two different cases of request/reply exchanges

UDP Server Types

- Simple server using one request/reply
- Complex server requiring multiple request/reply exchanges, e.g., TFTP

TFTP Concurrency

- According to the protocol, the server may create a *new udp port* and send the initial response from this new port
- The client should recognize this, and send all subsequent messages to the new port

When is it over?

- There is no *length of file* field sent!
- All data messages *except the last one* contain 512 bytes of data.
  - Message length is 2 + 2 + 512 = 516
- The last data message might contain 0 bytes of data!

Max Transfer Size?

- What if more than 65535 chunks are sent?
  - 65536 blocks x 512 bytes/block = 33,554,432 bytes.
- The RFC does not address this issue!

Extensions

- Extensions defined in RFC 2347, 2348 and 2349
- Allow passing additional control parameters
  - Use keywords followed by numeric values
  - E.g., timeout for unacked data
  - E.g., blksize for setting other packet sizes

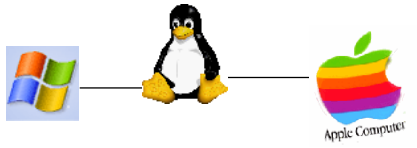
***

Why FTP Service?

- Purpose: to transfer files between two systems
- Goals of FTP service
  - Provide sharing of files (text, program and/or data)
  - Hide the complexity of how files are actually moved from one system to another
  - Handle the variations in file storage among systems, which can have different ways to represent data or text
  - Encourage indirect/implicit use of remote systems to keep a single or reliable copy

Problems of File Transfer

- At first, file transfer may seem simple
- But heterogeneous systems use different:
  - Operating Systems
  - Character Sets
  - Naming Conventions
  - Directory Structures
  - File Structures and Formats

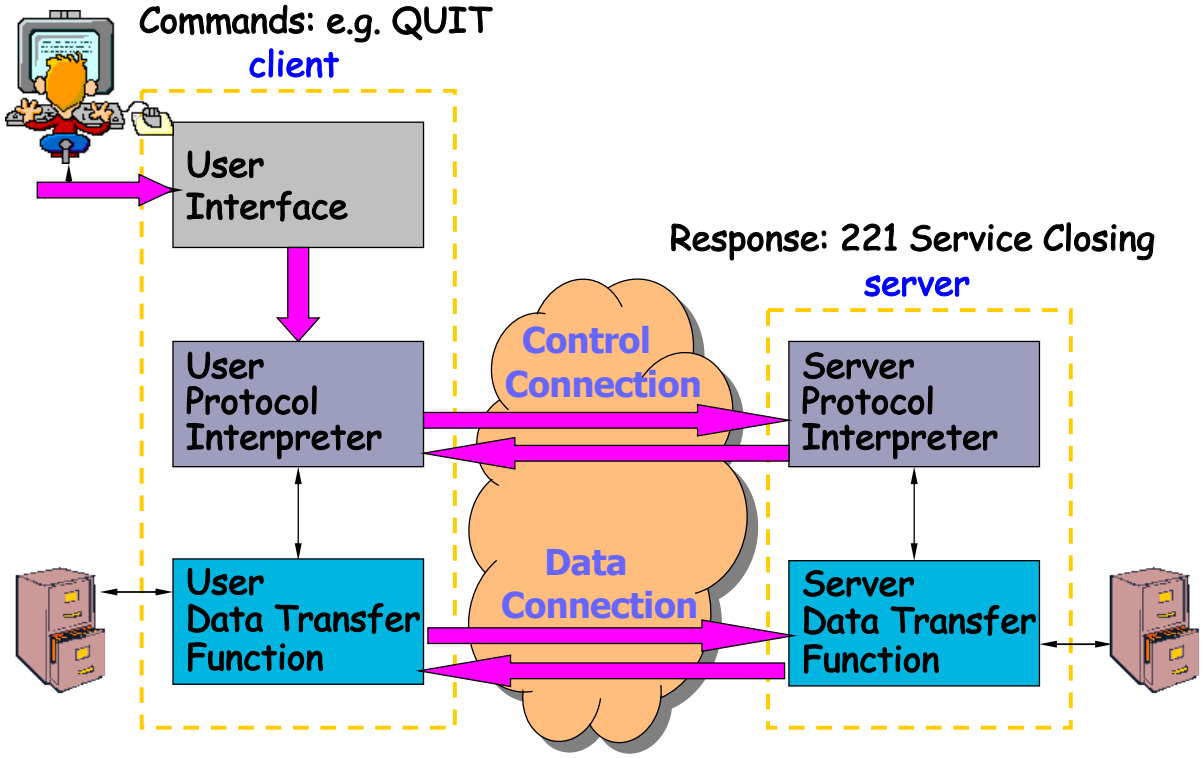


- FTP needs to address and resolve these problems

Control and Data Connections

- Control functions (commands) and reply codes are transferred over the control connection.
- All data transfer takes place over the data connection.
- The control connection must be "up" while data transfer takes place.

FTP's Model



Control Connection

- The control connection is the "well known" service: port 21
- The control connection similar to the TELNET protocol, using commands and responses
- Commands and replies are all line-oriented text (default is ASCII)
  - Two character '\r\n' end-of-line token

Access Control Commands

- To use, need an account or if the server allows anonymous user

USER	*specify user*
PASS	*specify password*
QUIT	*logout*

- For more security, other protocols need to be used, such as sftp

Data Transfer

- Client defines attributes about the file to be transferred
  - File type
  - Data structure
  - Transfer mode
- Default attributes defined
  - Ascii
  - 'File' structure (continuous seq. of bytes)
  - 'Stream' transfer mode

File Types

- Transferred over data connection
- ASCII file, default for sending text
- IMAGE file, default for sending binary files
  - E.g., executable programs
  - Continuous stream of bytes
  - No structure or encoding

Data Structure

- File (default): no structure, stream of bytes
- Record: used with text files to indicate the file is made up of records
- Page: indicates file is made up of independent indexed pages, with random storage or access

Data Transfer Mode

- Stream (default): file is transmitted as a stream of bytes
- Block: file is transmitted as a series of blocks preceded by 3-byte headers containing 1-byte descriptor code (EOF, EOR, restart marker) and 2-byte byte size
- Compressed: uses a simple compression scheme, compressed blocks are transmitted

Transfer Parameter Cmds (e.g.)

PORT	*publish local data port*
PASV	*server should listen*
TYPE	e*stablish data representation*
MODE	*establish transfer mode*
STRU	*establish file structure*

FTP Control Commands (e.g.)

RETR	*retrieve file from server*
STOR	*send file to server*
STOU	*send file and save as unique*
APPE	*send file and append*
ABOR	*abort prev. service command*
PWD	*print working directory*
LIST	*transfer list of files over data link*

User Interface Commands

- User commands trigger control commands
- Available commands may vary depending on the implementation (e.g., Linux vs Windows)

User command	Control command
put, mput	STOR
get,mget	RETR
ls	NLST
dir	LIST

FTP User Interface Commands

Command	Description
get *filename*	Retrieve file from server
mget *filename**	Retrieve multiple files from server*
put *filename*	Copy local file to server
mput *filename**	Copy multiple local files to server*
open *server*	Begin login to server
bye / close / exit	Logoff server
ls / dir	List files in current remote dir on server
lcd	Change local directory
cd	Change remote directory
rhelp / remotehelp	Lists commands the server accepts

* Sent to server as multiple command by User Proto Interpreter

FTP Replies

- All replies are sent over control connection
- Replies are a single line containing
  - 3 digit status code (sent as 3 numeric chars)
  - Ascii text format
- The FTP spec. includes support for multiline text replies

FTP Reply Status Code (1)

- First digit of status code indicates type of reply:
  - '1': Positive Preliminary Reply (got it, but wait)
  - '2': Positive Completion Reply (success)
  - '3': Positive Intermediate Reply (waiting for more information)
  - '4': Transient Negative Completion (error - try again)
  - '5': Permanent Negative Reply (error - can't do)

FTP Reply Status Code (2)

- 2nd digit indicates function groupings
  - '0' : Syntax (problem with command syntax)
  - '1' : Information (reply to help or status cmds)
  - '2' : Connections (problem with a connection)
  - '3' : Authentication (problem with login)
  - '4' : Unspecified
  - '5' : File system (related to file system)
- 3rd digit indicates specific problem within function group

Example FTP Responses

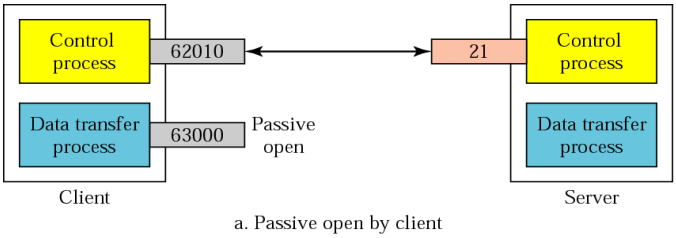
- **120** Service will be ready shortly
- **200** Command OK
- **230** User login OK
- **331** User name OK; password is needed
- **421** Service not available
- **530** User not logged in
- **552** Requested action aborted; exceeded storage allocation

Data Connection Management

- Data connection can be setup by either of two modes
  - Active mode
  - Passive mode
- Processing by middleboxes
  - E.g., firewall

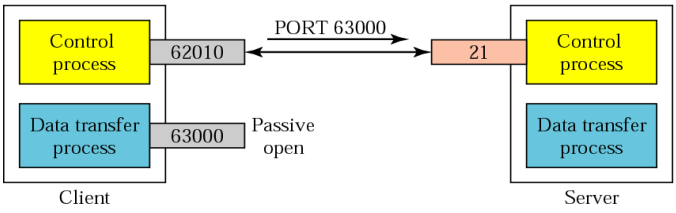
Active Mode

For data connection, client issues the passive open on local port.



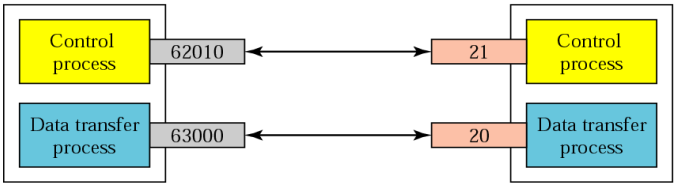
a. Passive open by client

Client sends the port number to server using PORT command.



b. Sending ephemeral port number to server

Server receives the port number and issues an active open using port number 20.



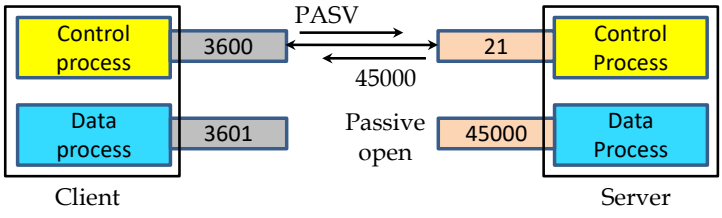
c. Active open by server

Passive Mode

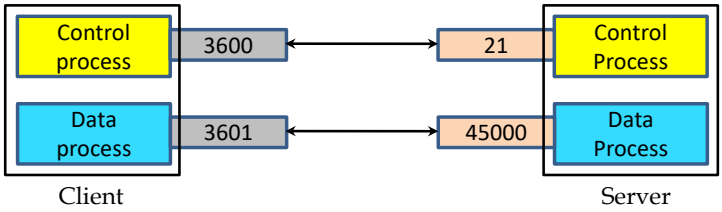
Firewall/NAT friendly mode

Client sends to server the PASV command.

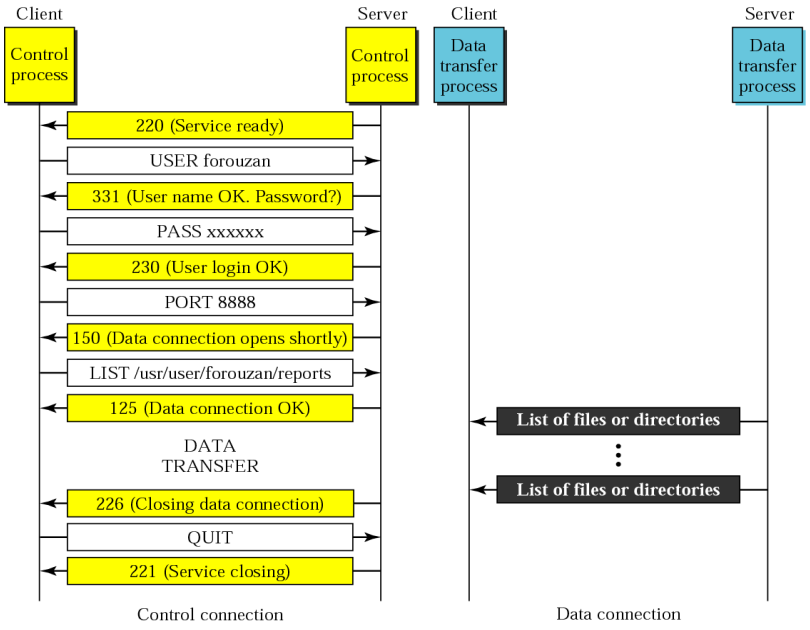
The Server dynamically picks a port and sends the port and the IP to the Client.



The Client issues an active open to the server port



FTP Exchange (e.g.)



RFC 959

- The RFC includes lots more information and many details including:
  - parameters for commands
  - lists of reply status codes
  - protocol state diagrams
  - support for a variety of file structures
  - sample sessions

TFTP vs. FTP

- FTP provides (minimal) security through login procedure
- TFTP has NO login procedure
- FTP Provides a reliable service through its use of TCP
- TFTP must handle its own retransmissions since it uses UDP
- FTP uses two connections
- TFTP uses one connection (stop and wait)
- FTP provides many commands
- TFTP can only read and write files