Mail Transfer

- SMTP (RFC2821) (was RFC821)
- Simple Command/Response Protocol
  - Client sends a command
  - Server responds
- Specified in partial (kind of) ABNF

```
The first step in the procedure is the MAIL command.
   MAIL FROM:<reverse-path> [SP <mail-parameters> ] <CRLF>

The second step in the procedure is the RCPT command.
   RCPT TO:<forward-path> [ SP <rcpt-parameters> ] <CRLF>
```

SMTP Specification

```
ehlo         = "EHLO" SP Domain CRLF
helo         = "HELO" SP Domain CRLF
ehlo-ok-rsp  =  ( "250"    domain [ SP ehlo-greet ] CRLF )
             /    "250-"   domain [ SP ehlo-greet ] CRLF
               *( "250-"   ehlo-line               CRLF )
                  "250"    SP ehlo-line            CRLF )
ehlo-greet   = 1*(%d0-9 / %d11-12 / %d14-127)
             ; string of any characters other than CR or LF
ehlo-line    = ehlo-keyword *( SP ehlo-param )
ehlo-keyword = (ALPHA / DIGIT) *(ALPHA / DIGIT / "-")
             ; additional syntax of ehlo-params depends on
             ; ehlo-keyword
ehlo-param   = 1*(%d33-127)
             ; any CHAR excluding <SP> and all
             ; control characters (US-ASCII 0-31 inclusive)
```

- Allows client to identify itself
  - And allow server to state its capabilities

SMTP basics

Client	Server
	Welcome Banı
HELO client-domain-name	OK
MAIL FROM:<address>	OK
RCPT TO:<address>	OK
DATA	Send it
E-mail message takes multiple lines so ends with a line that is just a dot .	OK
QUIT	OK

Commands and Responses

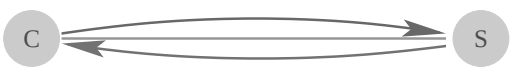
- Commands
  - 4 letters (by convention)
  - space
  - parameters
  - <CR><LF>
- Responses
  - Single Line
    - 3 digit code
    - space
    - human message
    - <CR><LF>
  - Multiple Lines
    - 3 digit code
    - hyphen "-"
    - human message
    - <CR><LF>
    - same 3 digit code
    - hyphen (if more lines to follow) space otherwise
    - human message
    - <CR><LF>

Response Codes

- First Digit gives response type
  - 1
    - Information
  - 2
    - All OK
  - 3
    - More input required
  - 4
    - Temporary Error
  - 5
    - Permanent Error
- Second digit gives specific response type
- Third digit gives extra information
  - Allows error codes to be distinguished

File Transfer Protocol (FTP)

- FTP uses an unusual communication model
- Traditional Client/Server (SMTP/DNS/...) :

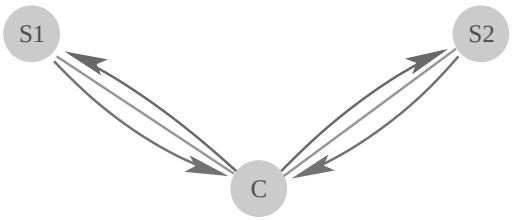


- Client and server
- Client opens TCP connection to server
- Client sends commands over connection
- Server sends responses back over same connection

FTP Model

- Client and two Servers

RFC959

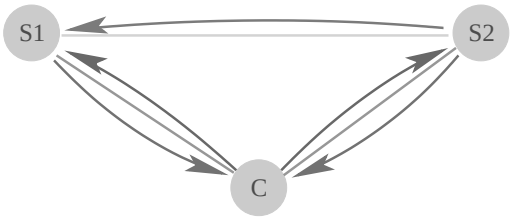


- Client opens TCP control connection
  - to first server
- And another TCP control connection
  - to second server
- Client sends commands to both servers
- Gets responses from servers

FTP model (2)

- Command to one server
  - causes it to open TCP data connection
  - to other server

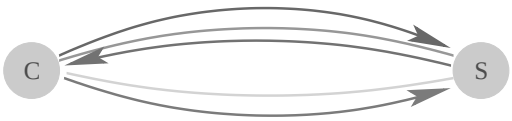
RFC959



- Command to server causes it to transfer data over data connection
- Data transfer can be in either direction

Common FTP Usage

- Client and one server in same system



- Client opens TCP control connection
- Client sends commands, gets responses
- Command causes server to open data connection to other server (ie: client)
- Data Connection opened by either end
  - Depends upon which server client shares
- Data transfer occurs over data connection
- Data transfer can be in either direction

FTP Control Connection

- Very similar to SMTP
  - SMTP was once part of FTP
- 4 letter commands, plus parameters
- 3 digit (+ message) response
  - Multi-line responses possible
  - Some responses contain information for software
- Authentication commands
- Commands to set options
- Commands to cause data connection setup
- Commands to cause data transfer to occur

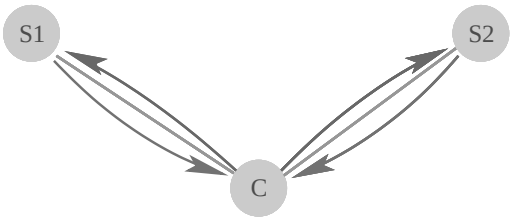
FTP Data Connection

- Used for "File Data" Only
  - File Data includes process output
    - directory listings
- Has MODE and TYPE
  - MODE - format of data transfer
    - STREAM
    - PAGE
    - BLOCK
  - TYPE - interpretation of data transferred
    - ASCII
    - IMAGE
    - Logical Bytes (with byte size)
- Designed to allow like systems to be able
  - to transfer files
  - without reformatting at either end
- But to also allow dissimilar systems
  - to exchange meaningful information

Creating Data Connections

- Client must tell server location of other server
  - Even if other server is the client
- And/or must tell other server to expect incoming connection
  - Easy if other server is client
- Done
  - By requesting address & port number
    - at server to be target of connection
  - By sending address & port number
    - to create a connection

Data connection method

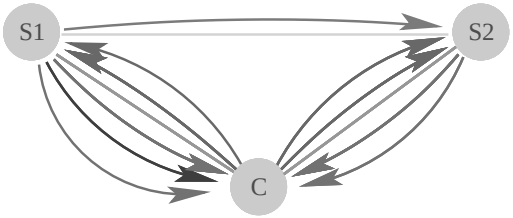


- Client command connections to 2 servers
- Client to server: prepare to receive data
- Server replies with its address and port
- Client to other server: use this address
- That server remembers value, says OK

PASV

PORT

Data connection method (2)



- Client tells one server to store a file
- Server says OK, and expects incoming data
- Client tells other server to send a file
- Server opens data connection as requested
  - sends file
  - tells client transaction in progress
- When complete, server tells client all done

Active vs Passive

- For 2 party FTP
  - Client is one of the servers
    - Data transfer to or from client
    - The normal case

PASSIVE FTP

- Client chooses to send PASV
  - Then open data connection to server
    - At address+port server specifies

ACTIVE FTP

- Client chooses to send PORT
  - Server opens data connection
    - To client at address+port specified
    - Always from PORT 20
- Active FTP used to be almost universal
  - Problems caused by firewalls
- Now Passive FTP used almost everywhere
    - Hard to classify FTP data transfer connections

Problem for IPv6

- Address is an IPv4 address
  - Protocol specifies 6 numbers each 0-255
    - 4 give IP address
    - 2 give port number
- How to extend to IPv6?
- FOOBAR
  - FTP Operation Over Big Address Records
    - New command LPRT (Long PoRT)
      replaces PORT
      - give length of address & port
      - any number of bytes
    - And LPSV
      replaces PASV
      - returns same info

Problem with NAT

- NAT translates addresses in headers
  - Body of message might be encrypted
    - FTP has an encryption extension
- How to handle addresses in data?
  - No way in simple NAT
  - Requires Application Level Gateway
    - FTP client to ALG in NAT box
    - ALG client of FTP server
      - All hidden from FTP client/server
  - Cannot do authentication/encryption
    - Hidden ALG could not remain hidden
- Problems caused by addresses
  - PORT command
  - PASV response
    - FOOBAR did not fix this

Alternative solution

- Note that most connections
  - have only client & server
- Client already knows server IP address
  - v4 or v6
  - It connected to server
- Server already knows client IP address
  - sees where connection came from
- No need to transmit IP addresses
  - in most cases
  - Make them optional

EPRT & EPSV

- New command EPRT (Extended PoRT)
  - Address type
  - Address
  - Port
  - separated by delimiters
    - No fixed lengths
    - No need to specify length
- Compatibility ?
  - If unknown at server
    - error code - client uses PORT
  - If address type unknown
    - server indicates which types it knows
- Also EPSV
  - Only returns port
    - No addresses
  - Require EPSV in 2 party FTP
    - Always use passive mode