Network Applications: File Transfer Protocol; HTTP/1.0

Qiao Xiang

https://qiaoxiang.me/courses/cnns-xmuf21/index.shtml

10/14/2021

This deck of slides are heavily based on CPSC 433/533 at Yale University, by courtesy of Dr. Y. Richard Yang.

Outline

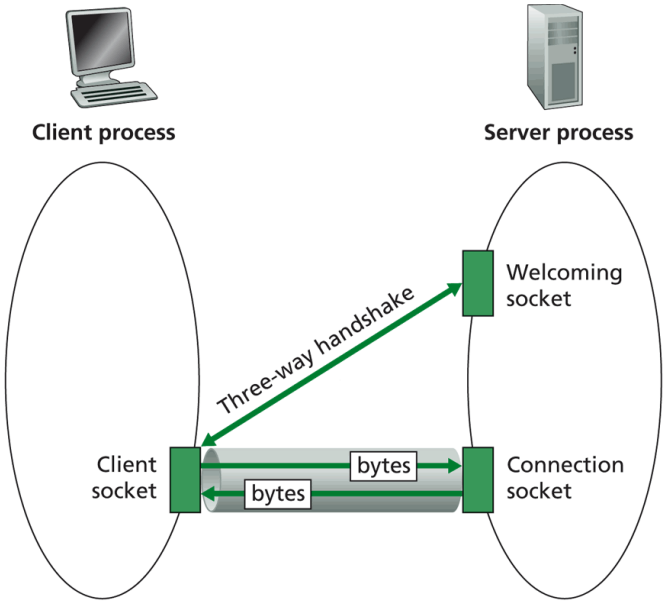
- Admin. and recap
- Network application programming
  - UDP sockets
  - TCP sockets
- Network applications (continue)
  - File transfer (FTP) and extension
  - HTTP
    - HTTP/1.0

Admin.

- Lab assignment 1 due today

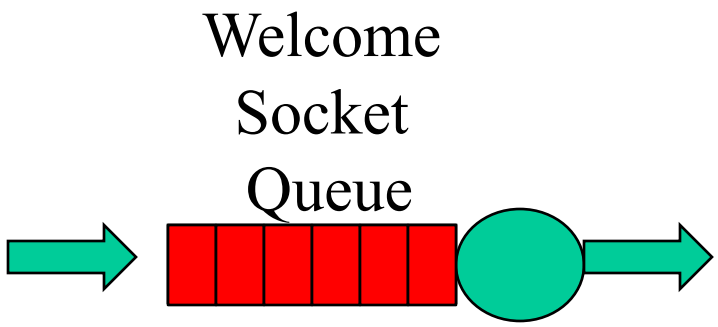
Recap: TCP Sockets

- TCP server socket demux by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number



Analysis

- Assume that client requests arrive at a rate of lambda/second
- Assume that each request takes $1/\mu$ seconds
- A basic question
  - How big is the backlog (welcome queue)



Analysis

- Is there any interop issue in the sample program?

Analysis

- Is there any interop issue in the sample program?
  - DataOutputStream writeBytes(String) truncates
    - http://docs.oracle.com/javase/1.4.2/docs/api/java/io/DataOutputStream.html#writeBytes(java.lang.String)

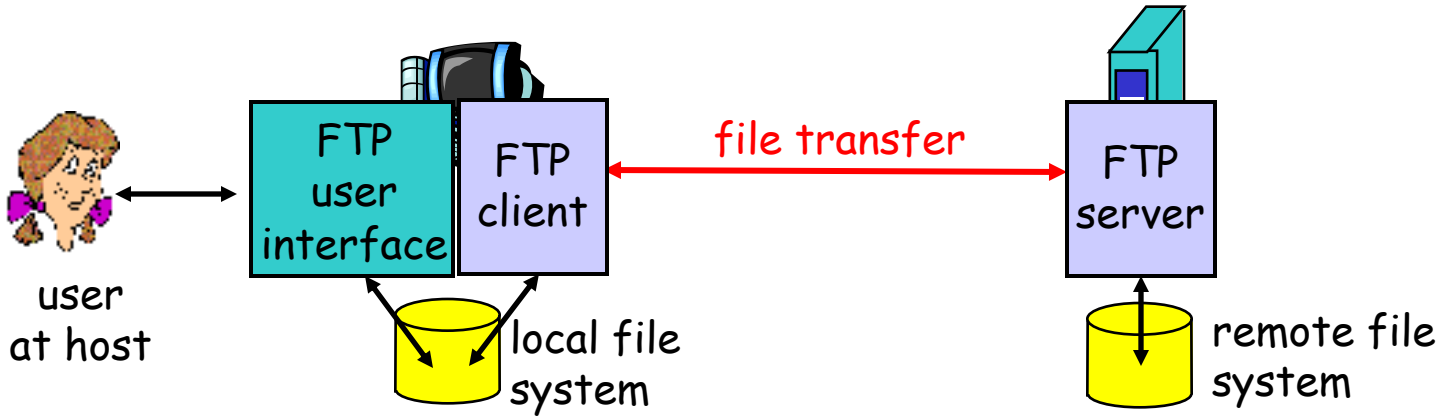
Summary: Basic Socket Programming

- They are relatively straightforward
  - UDP: DatagramSocket
  - TCP: ServerSocket, Socket
- The main function of socket is multiplexing/demultiplexing to application processes
  - UDP uses (dst IP, port)
  - TCP uses (src IP, src port, dst IP, dst port)
- Always pay attention to encoding/decoding

Outline

- Admin. and recap
- Network application programming
  - UDP sockets
  - TCP sockets
- Network applications (continue)
  - *File transfer (FTP) and extension*

FTP: the File Transfer Protocol



- Transfer files to/from remote host
- Client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 21/20 (smtp 25, http 80)

FTP Commands, Responses

Sample commands:

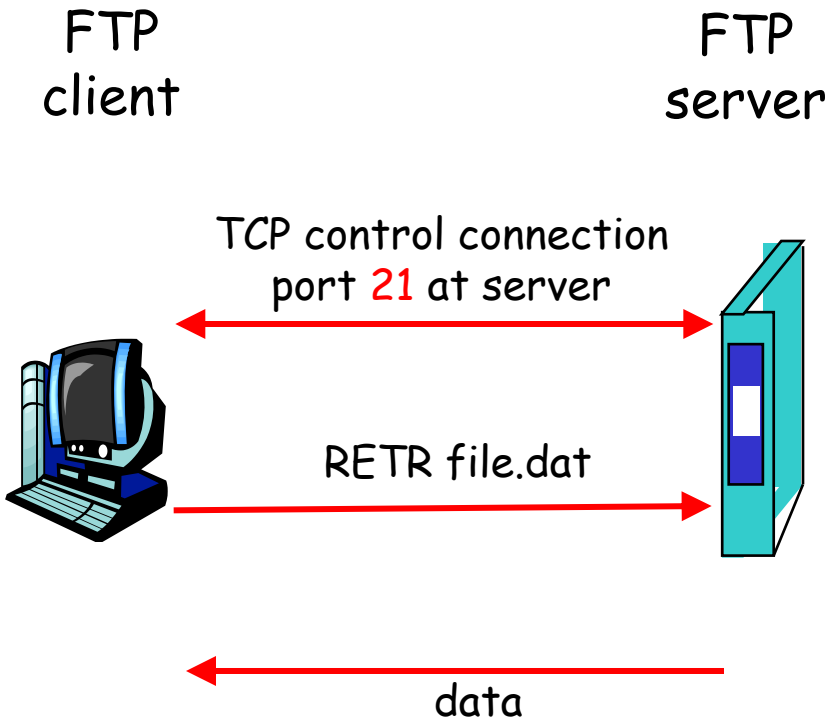
- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **PWD** returns current dir
- **STAT** shows server status
- **LIST** returns list of file in current directory
- **RETR filename** retrieves (gets) file
- **STOR filename** stores file

Sample return codes

- status code and phrase
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

FTP Protocol Design

- What is the simplest design of data transfer?



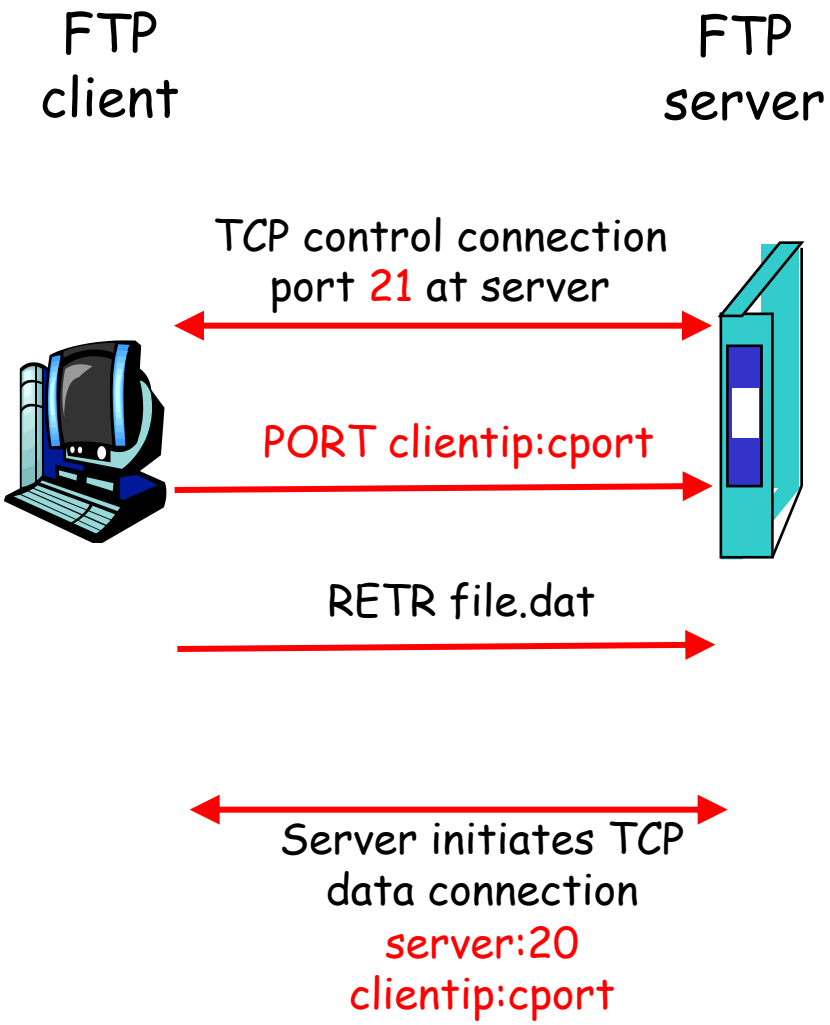
FTP: A Client-Server Application with Separate Control, Data Connections

- Two types of TCP connections opened:
  - A control connection: exchange commands, responses between client, server.
    "out of band control"
  - Data connections: each for file data to/from server

Discussion: why does FTP separate control/data connections?

Q: How to create a new data connection?

Traditional FTP: Client Specifies Port for Data Connection



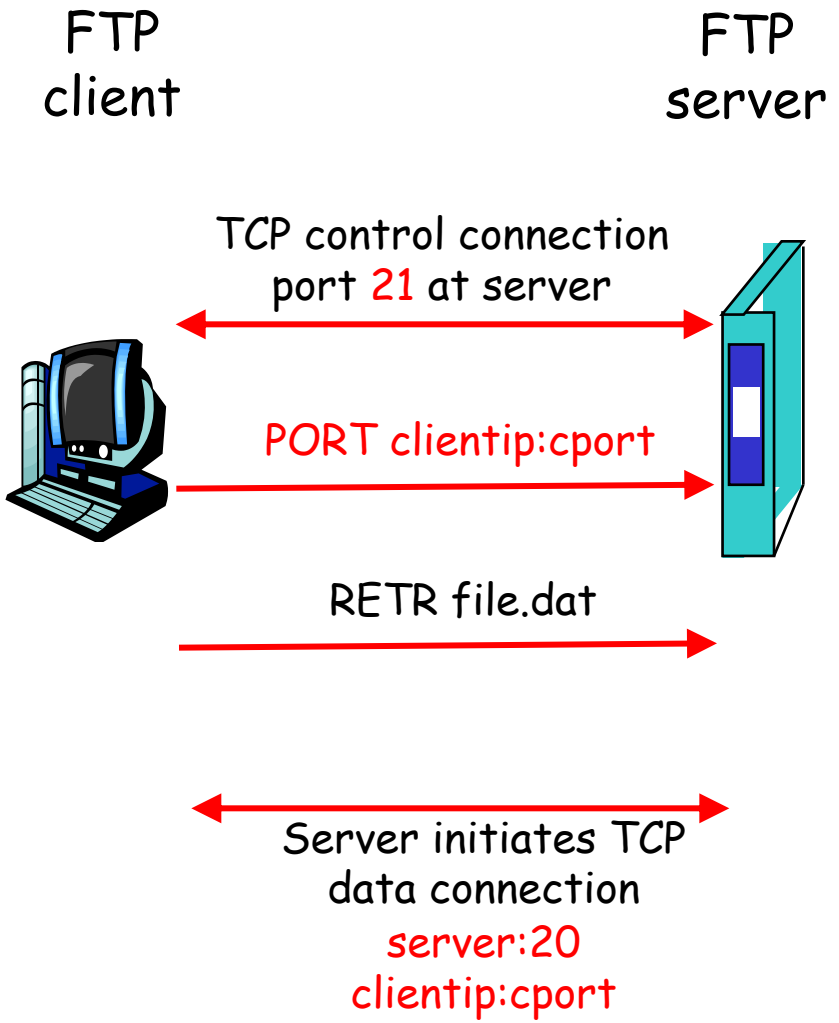
Example using telnet/nc

- Use telnet for the control channel
  - telnet ftp.ietf.org 21
  - user anonymous
  - pass your_email
  - port 10,90,61,172,4,1
  - list

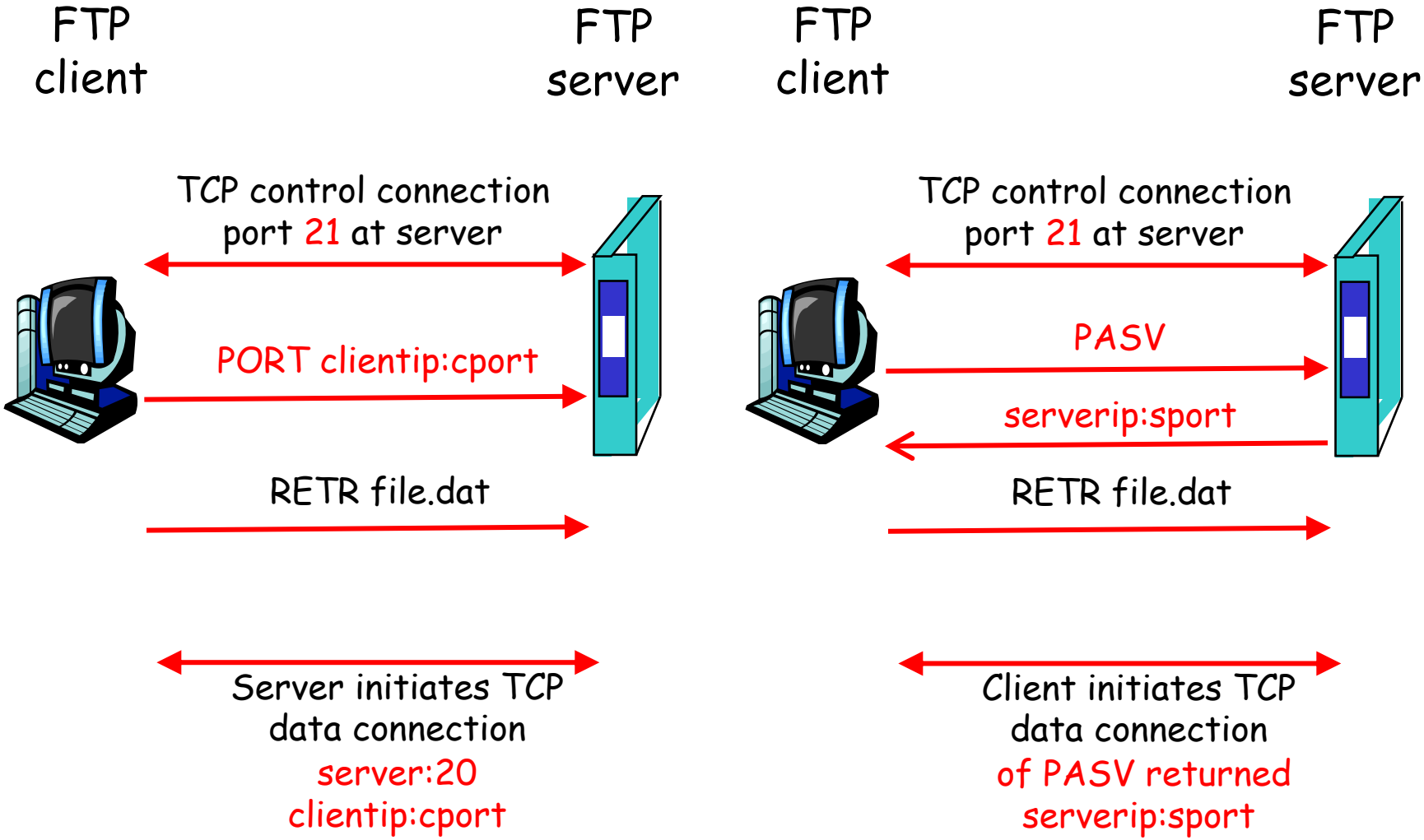
- use nc (NetCat) to receive/send data with server
  - nc –v –l 1025

Problem of the Client PORT Approach

- Many Internet hosts are behind NAT/firewalls that block connections initiated from outside



FTP PASV: Server Specifies Data Port, Client Initiates Connection

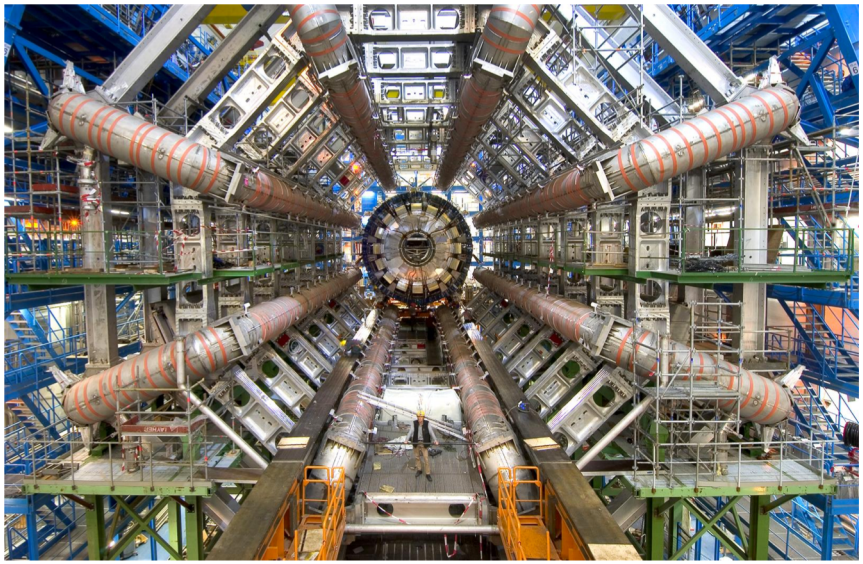


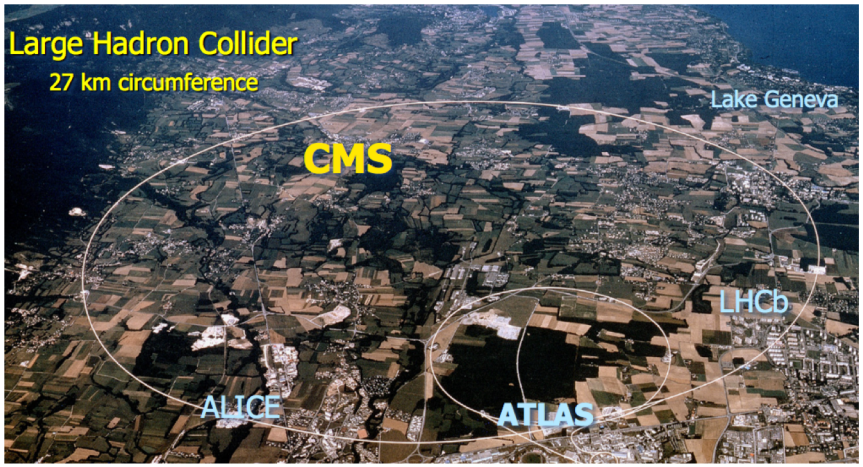
Example

- Use Wireshark to capture FTP traffic
  - Using chrome to visit
    ftp://ftp.freebsd.org

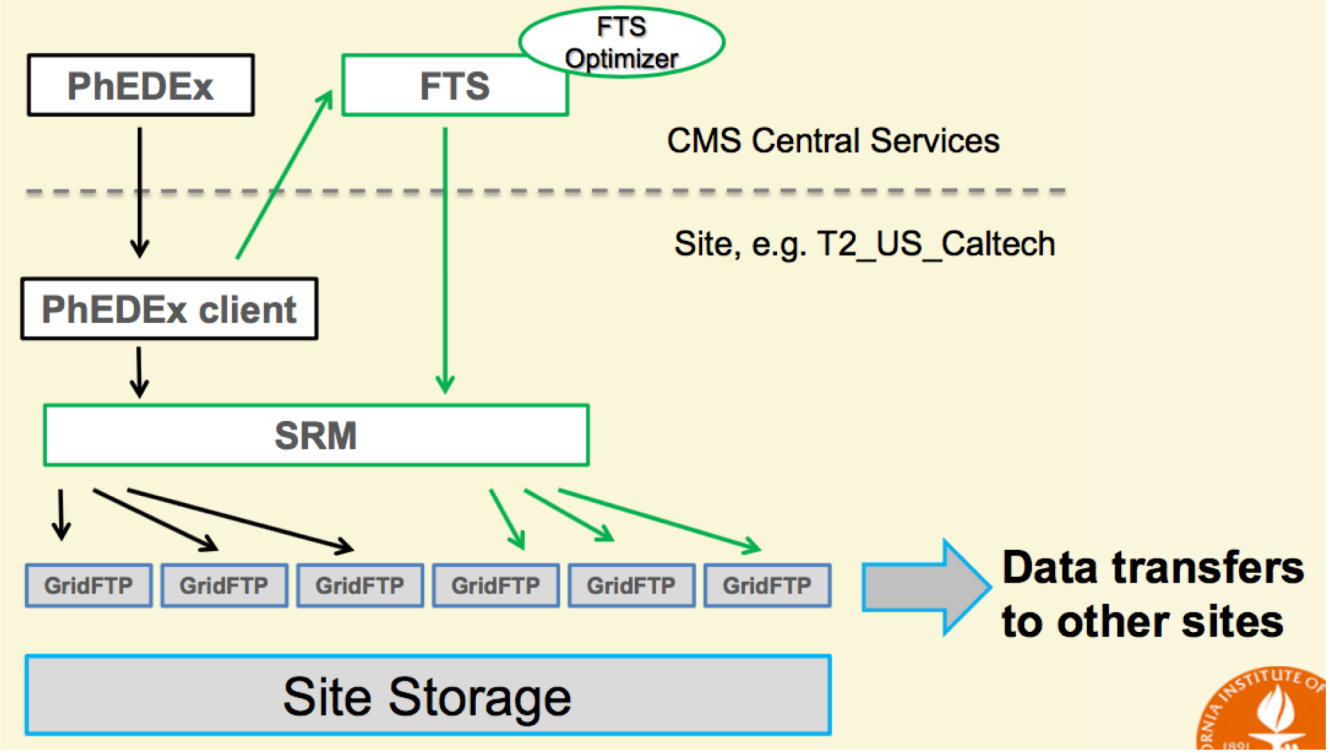
FTP Extensions

- FTP with extensions are being used extensively in large data set transfers (e.g., LHC)





Data Transfer Structure

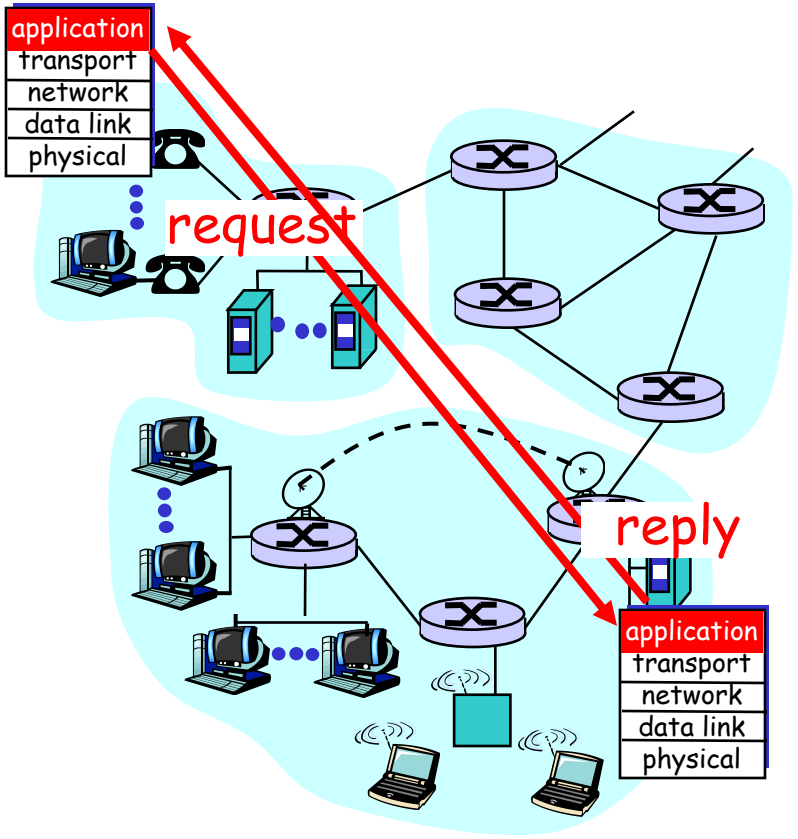


- See GridFTP to FTP extensions
  - https://www.ogf.org/documents/GFD.20.pd
- Goal of GridFTP: allow parallel, high-throughput data transfer
  - Discussion: What features do we need to add to FTP to allow parallel transfers?

FTP Evaluation

Key questions to ask about a C-S application

- **Is the application extensible?**
- **Is the application scalable?**
- **How does the application handle server failures (being robust)?**
- **How does the application provide security?**



What are some interesting design features of the FTP protocol?

Outline

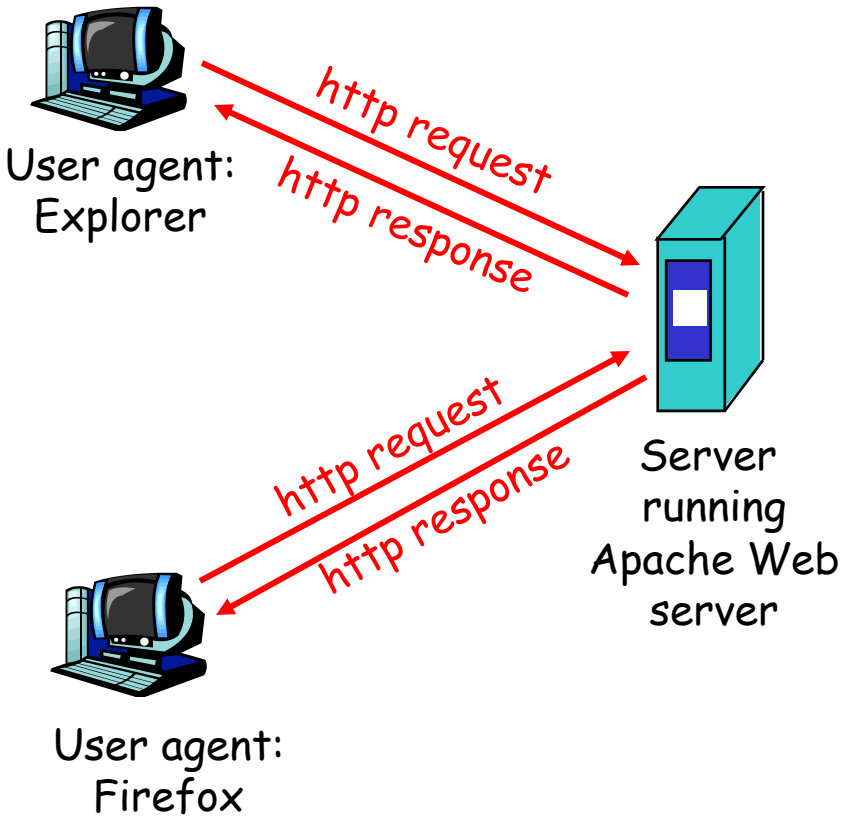
- Admin. and recap
- Network application programming
  - UDP sockets
  - TCP sockets
- Network applications (continue)
  - File transfer (FTP) and extension
  - *HTTP*

From Opaque Files to Web Pages

- Web page:
  - authored in HTML
  - addressed by a URL
    - URL has two components:
      - host name, port number and
      - path name

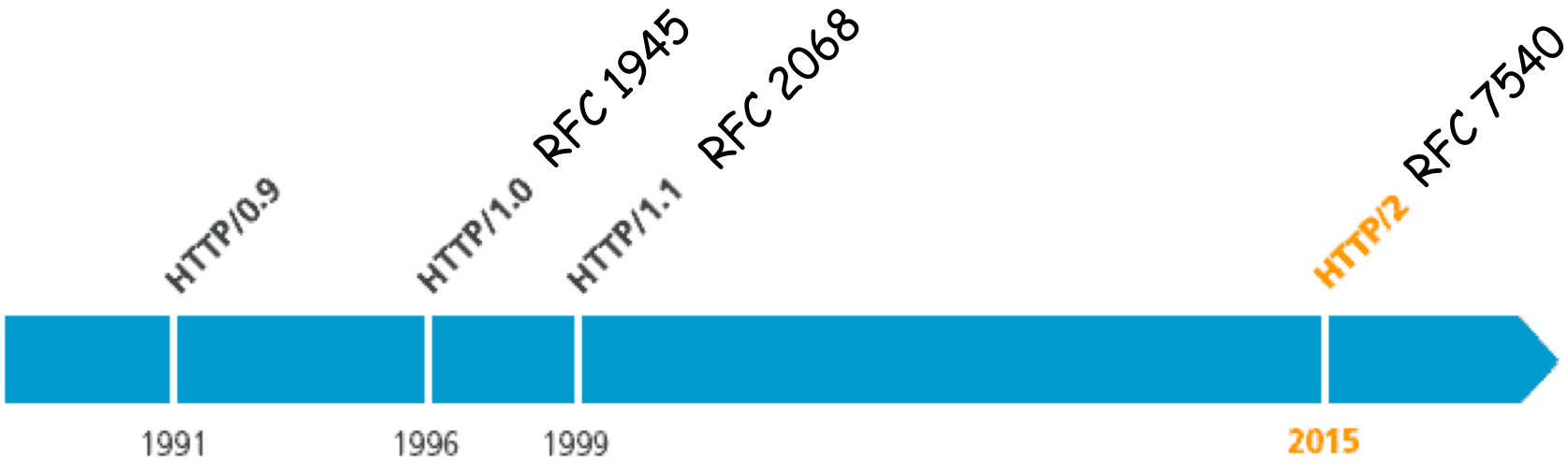
`http://qiaoxiang.me:80/index.html`

- Most Web pages consist of:
  - base HTML page, and
  - several referenced objects
    - E.g., image



The Web pages are requested through HTTP: hypertext transfer protocol

HTTP is Still Evolving



HTTP 1.0 Message Flow

- Server waits for requests from clients

- Client initiates TCP connection (creates socket) to server, port 80
- Client sends request for a document
- Web server sends back the document
- TCP connection closed

- Client parses the document to find embedded objects (images)
  - repeat above for each image

HTTP 1.0 Message Flow (more detail)

Suppose user enters URL
qiaoxiang.me/index.html

0. http server at host qiaoxiang.me waiting for TCP connection at port 80.

1a. http client initiates TCP connection to http server (process) at qiaoxiang.me. Port 80 is default for http server.

1b. server "accepts" connection, ack. client

2. http client sends http *request message* (containing URL) into TCP connection socket

3. http server receives request message, forms *response message* containing requested object (index.html), sends message into socket (the sending speed increases slowly, which is called slow-start)

time

HTTP 1.0 Message Flow (cont.)

4. http server closes TCP connection.

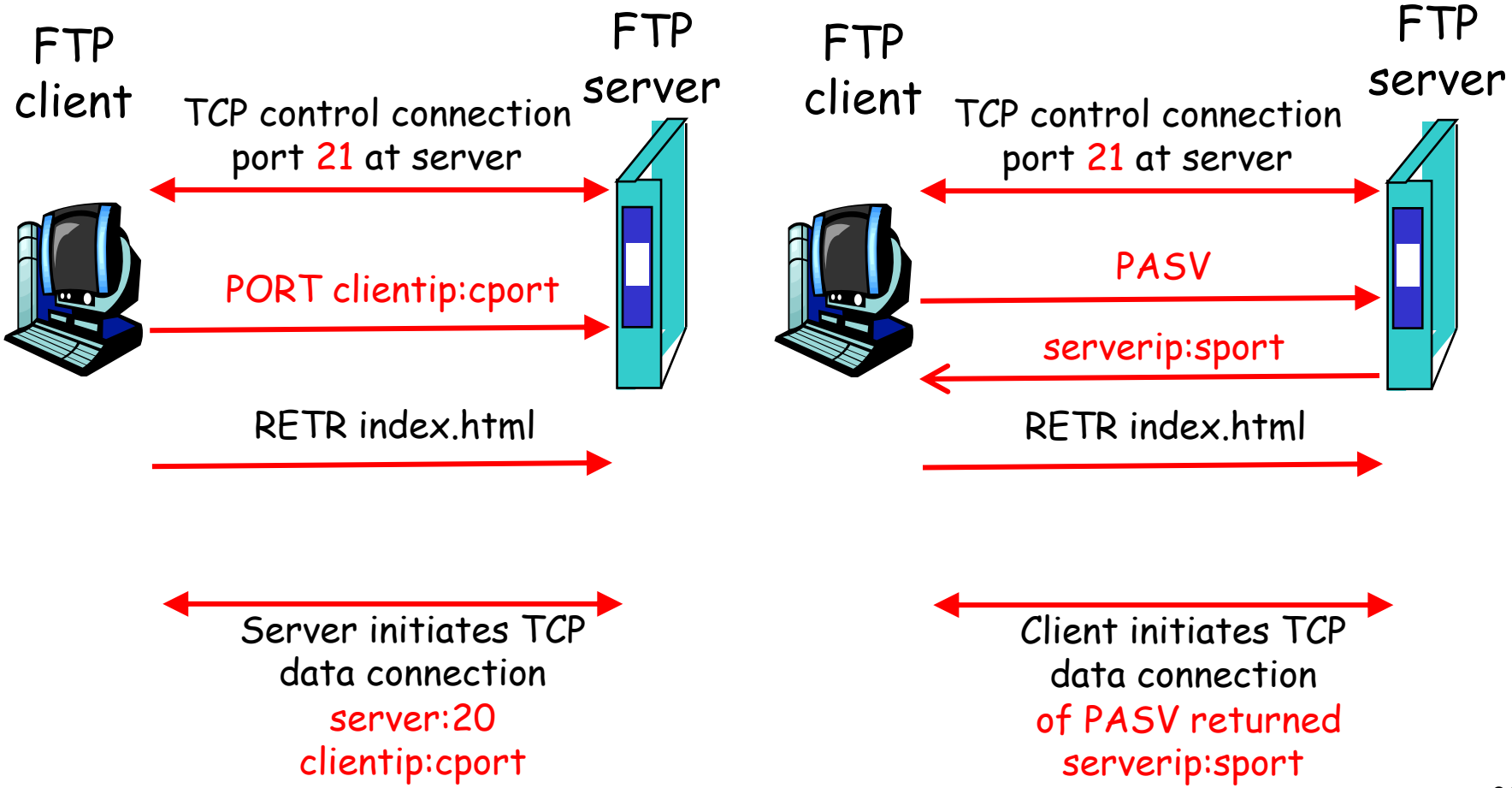
5. http client receives response message containing html file, parses html file, finds embedded image

time

6. Steps 1-5 repeated for each of the embedded images

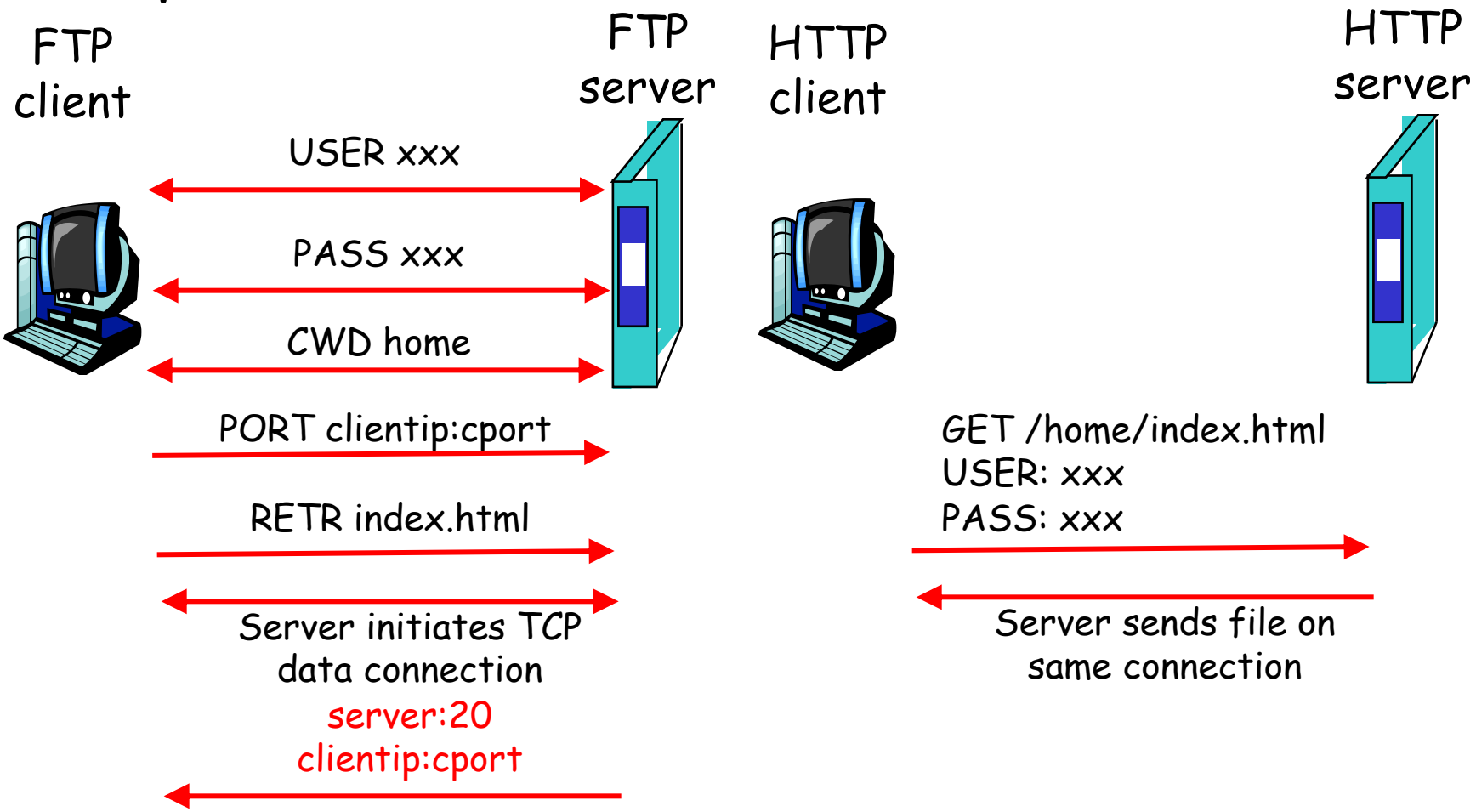
Discussion

- How about we use FTP as HTTP?



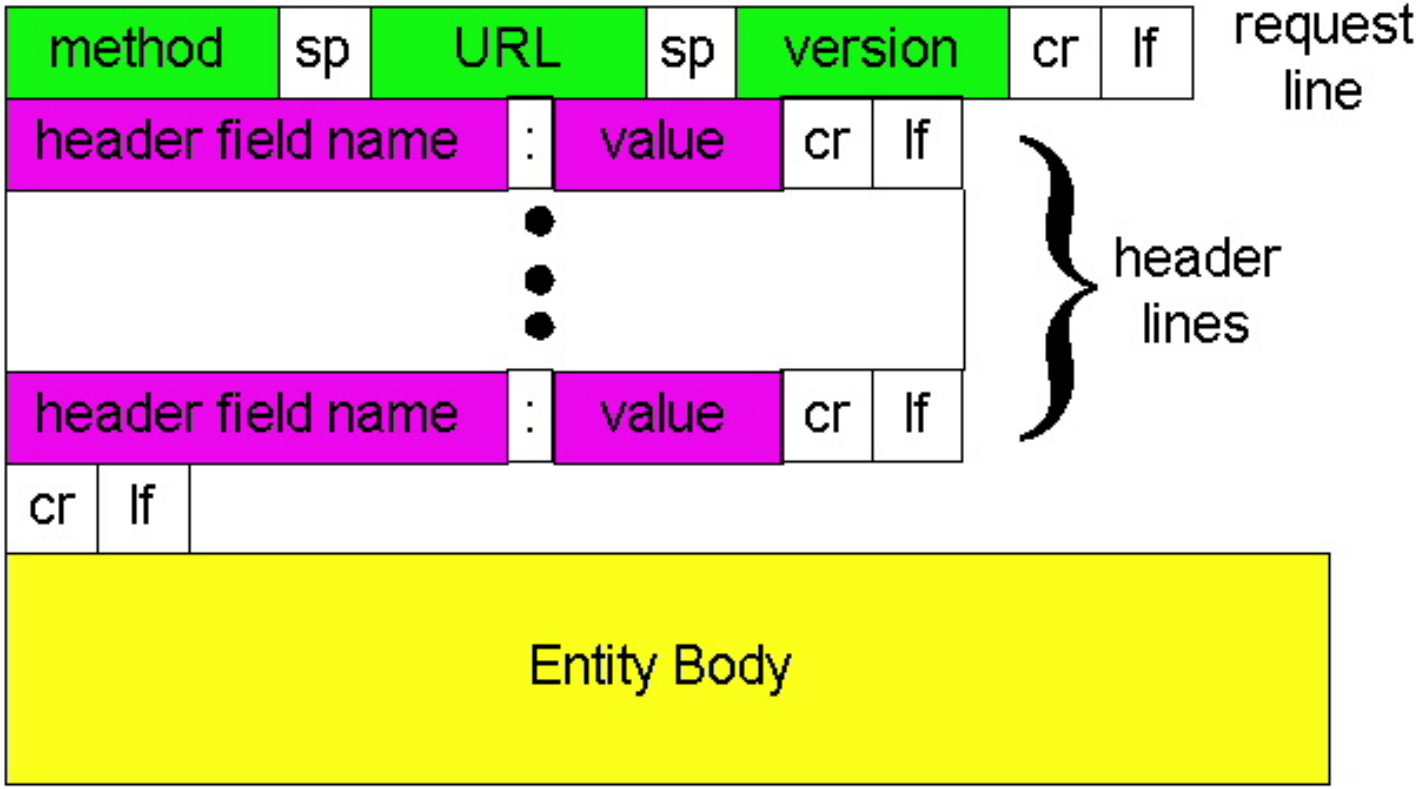
HTTP1.0 Message Flow

- HTTP1.0 servers are stateless servers: each request is self-contained



HTTP Request Message: General Format

- ASCII (human-readable format)



Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet qiaoxiang.me 80
```

Opens TCP connection to port 80 (default http server port) at qiaoxiang.me. Anything typed in sent to port 80 at qiaoxiang.me

2. Type in a GET http request:

```
GET /index.html HTTP/1.0
```

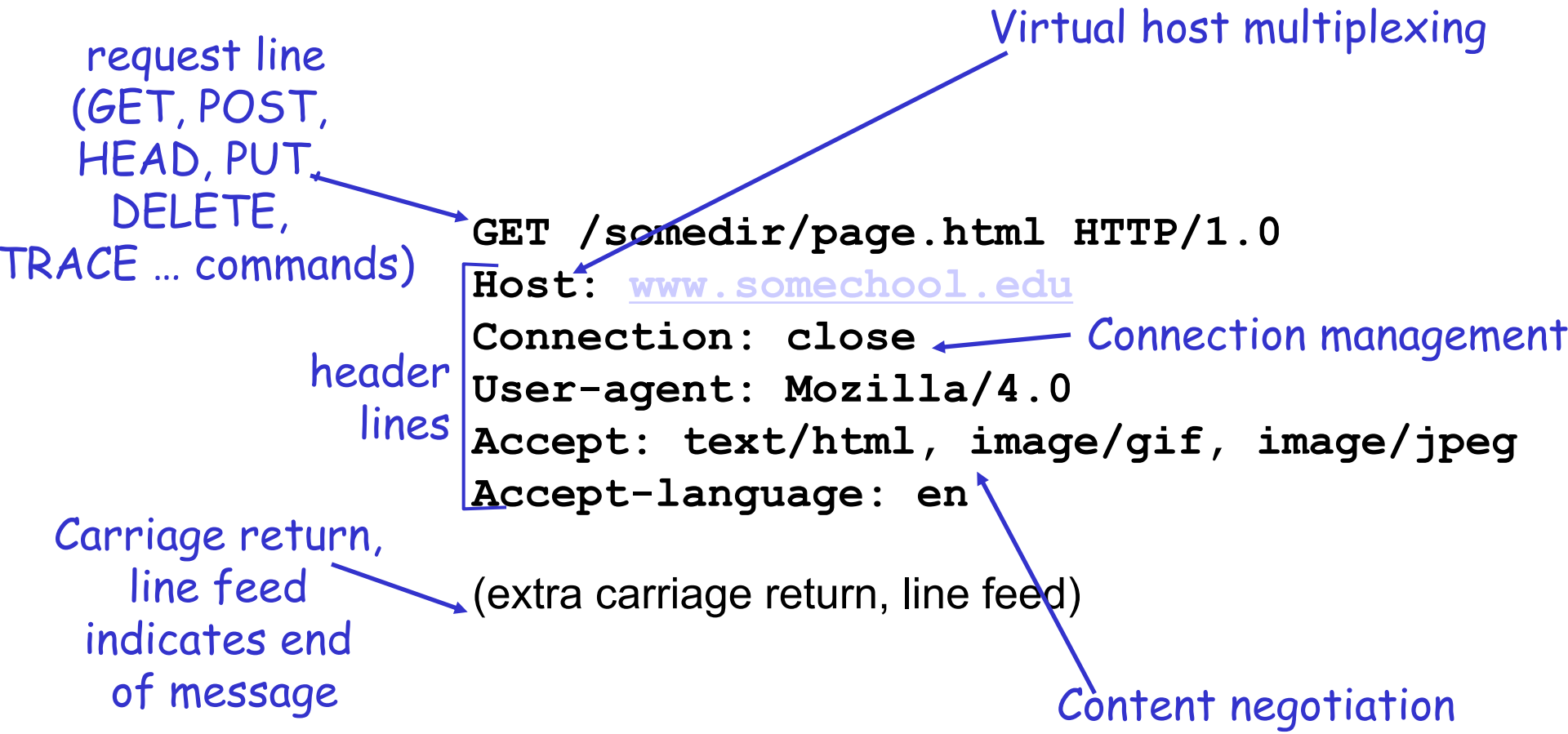
By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to http server

3. Look at response message sent by the http server.

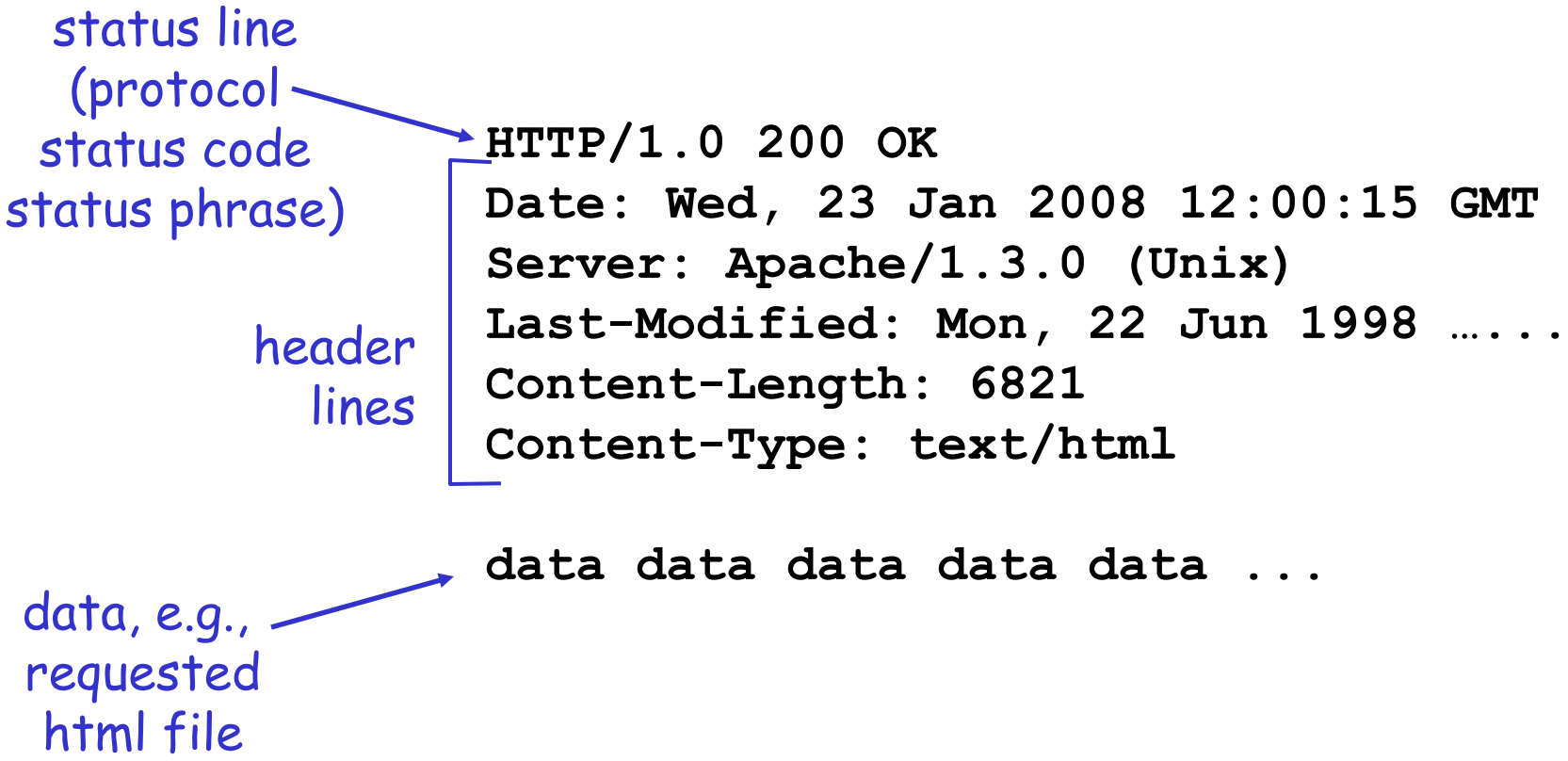
Trying out HTTP (client side) for yourself

- Try telnet GET on www.xmu.edu.cn

HTTP Request Message Example: GET



HTTP Response Message



HTTP Response Status Codes

In the first line of the server->client response message. A few sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

- requested object moved, new location specified later in this message (Location:)

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

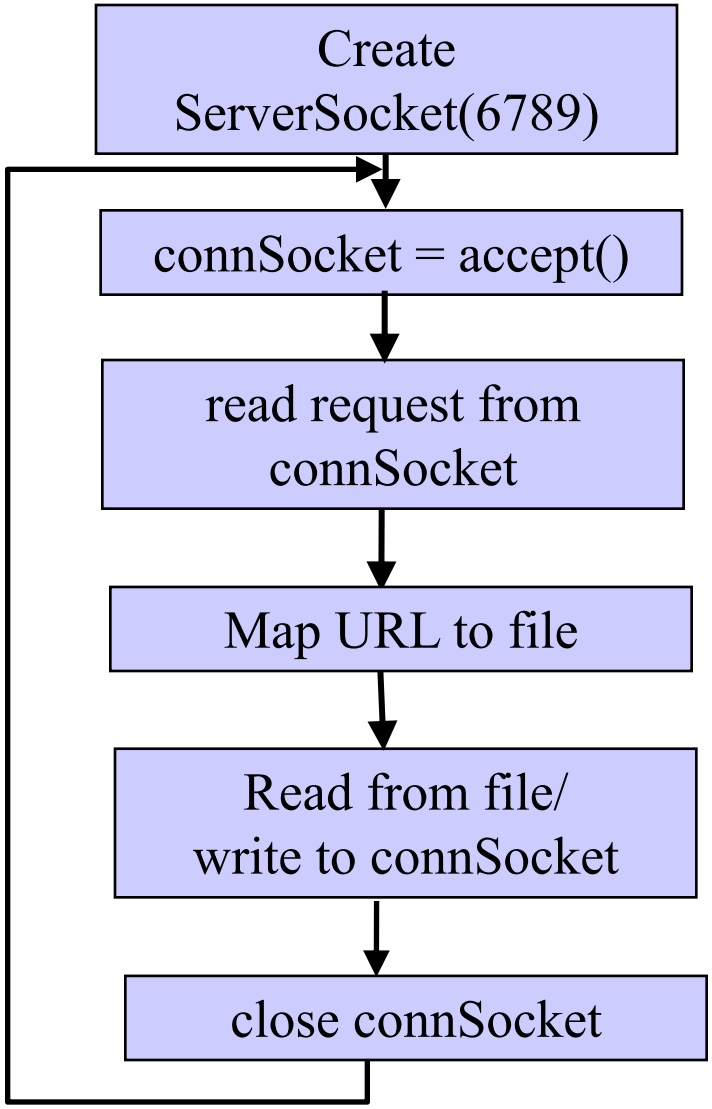
Trying Use Chrome to visit Course Page

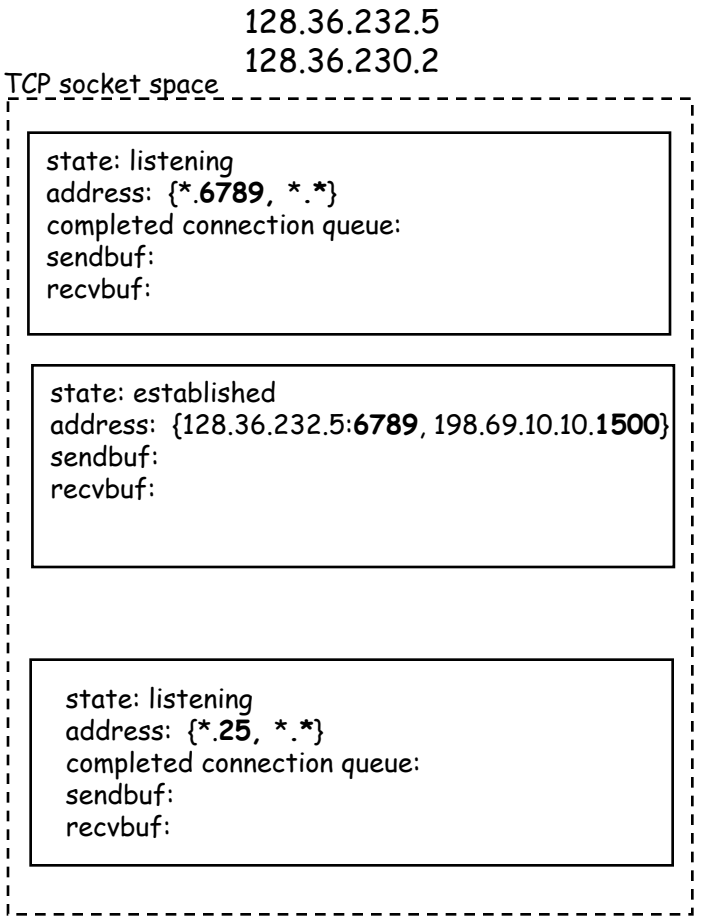
Design Exercise

- Workflow of an HTTP server processing a GET request that maps to a file:

```
GET /somedir/page.html HTTP/1.0
Host: www.somechool.edu
```

Basic HTTP Server Workflow





Example Code

- See BasicWebServer.java

- Try using telnet and real browser, and fetch
  - file1.html
  - index.html

  what difference in behavior?

Static -> Dynamic Content