Network Applications: File Transfer Protocol; HTTP/1.0

Qiao Xiang, Congming Gao, **Qiang Su**
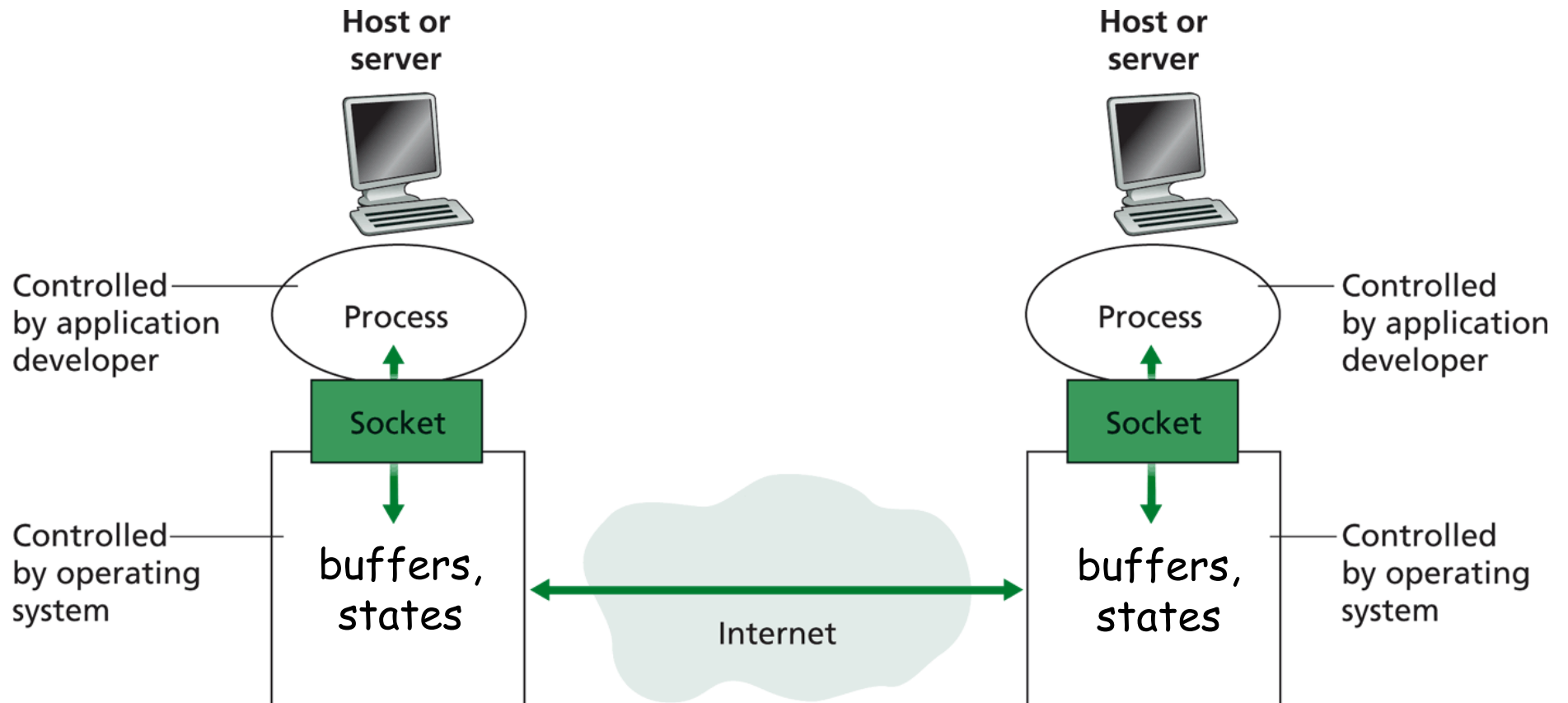
https://sngroup.org.cn/courses/cnns-xmuf25/index.shtml

09/28/2025

Outline
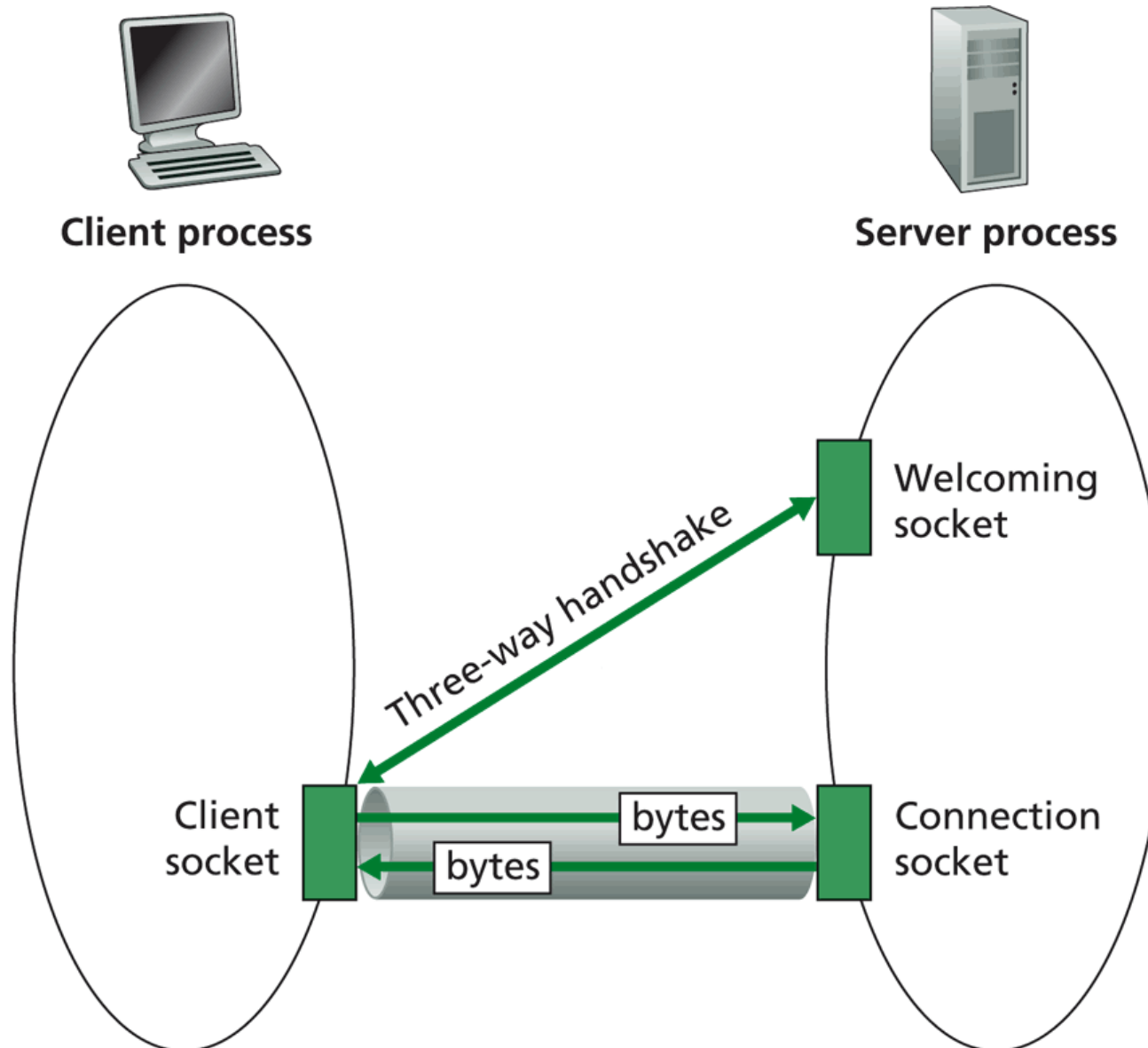
❑ Admin. and recap

❑ Network application programming

   o UDP sockets

   o TCP sockets

❑ Network applications (continue)

   ➢ *File transfer (FTP) and extension*

Big Picture: Socket

Recap: TCP Socket Big Picture

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**

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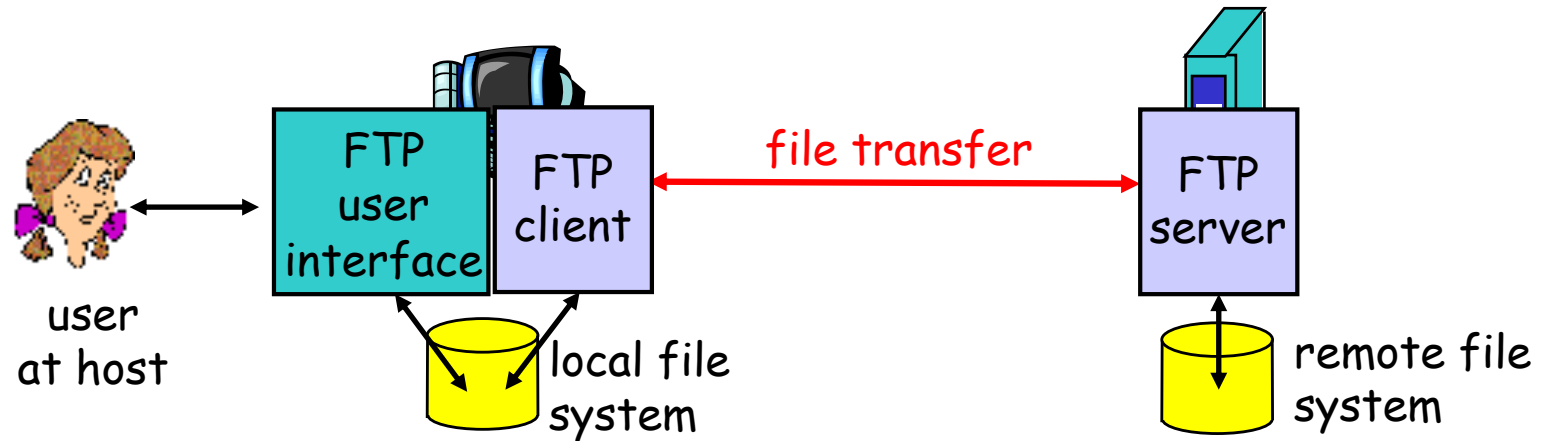
- ❑ Transfer files to/from remote host
- ❑ Client/server model
  - o *client:* side that initiates transfer (either to/from remote)
  - o *server:* remote host
- ❑ ftp: RFC 959
- ❑ ftp server: port 21/20 (smtp 25, http 80)

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 client

FTP server

TCP control connection
port **21** at server
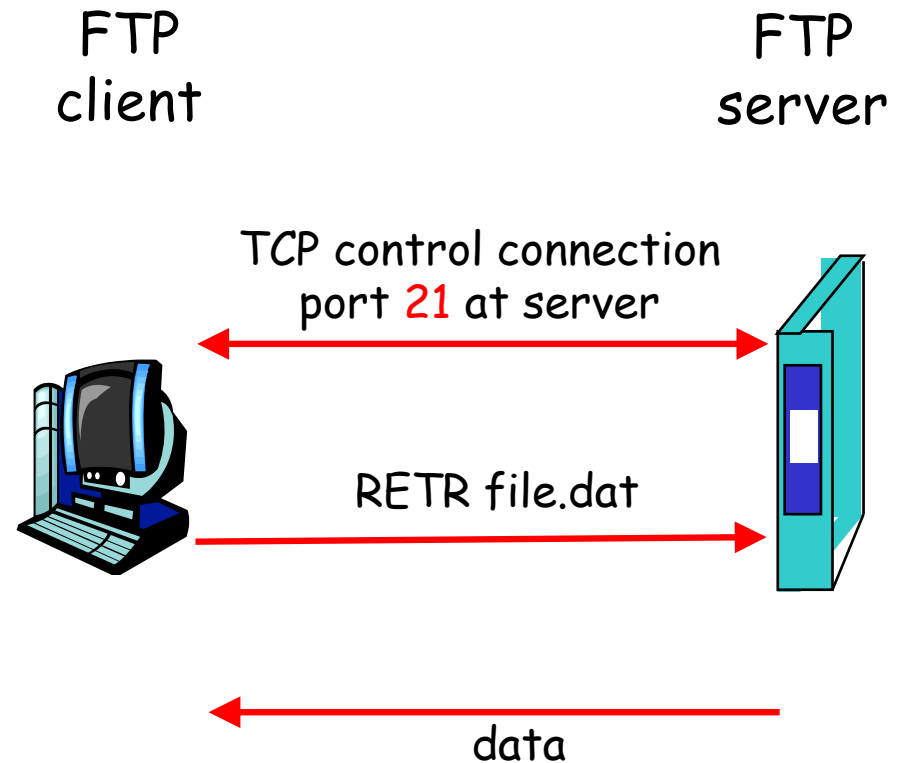
RETR file.dat

data

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

❑ **Two** types of TCP connections opened:

  o A **control connection:** exchange commands, responses between client, server.
     "out of band control"

  o **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

FTP
client

FTP
server

TCP control connection
port 21 at server

PORT clientip:cport

RETR file.dat

Server initiates TCP
data connection
server:20
clientip:cport
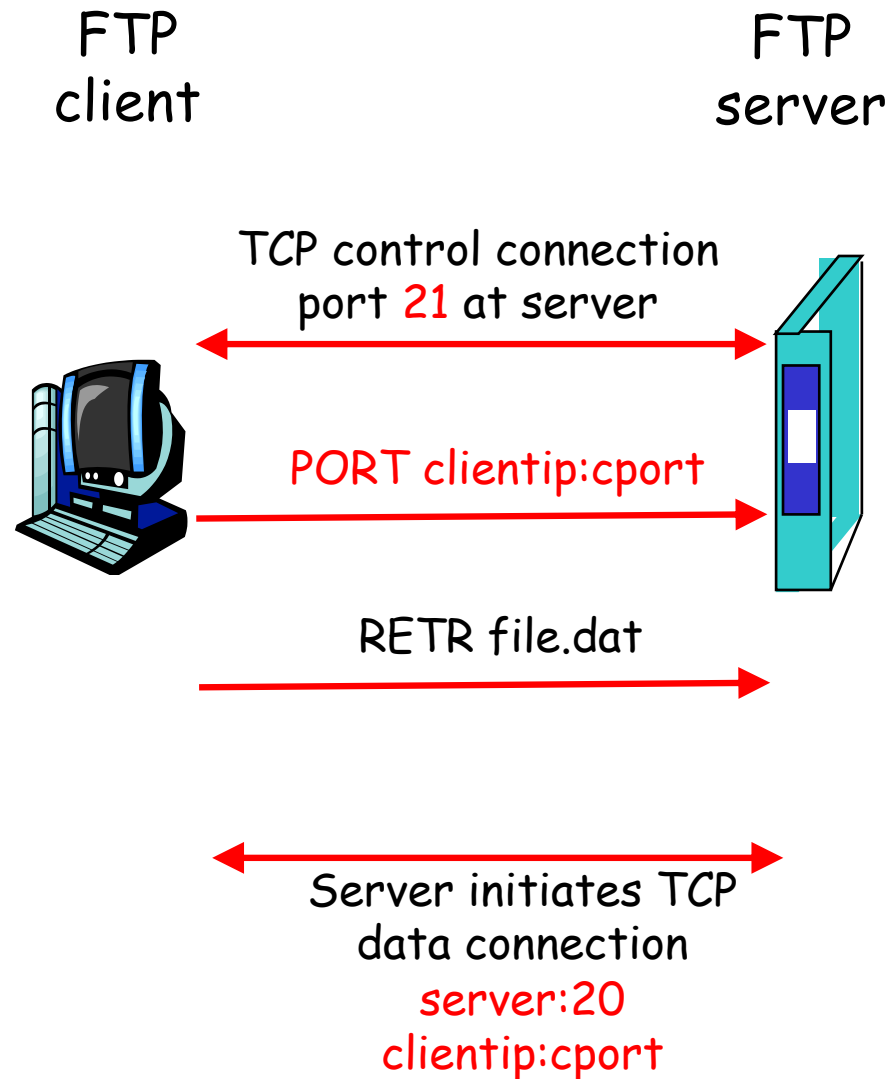
Example using telnet/nc

❑ Use telnet for the control channel

- ○ telnet ftp.ietf.org 21
- ○ user anonymous
- ○ pass your_email
- ○ port 10,90,61,172,4,1
- ○ list

         client   port
    IP address number

❑ 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 client

FTP server

TCP control connection
port 21 at server

PORT clientip:cport

RETR file.dat

Server initiates TCP
data connection
server:20
clientip:cport
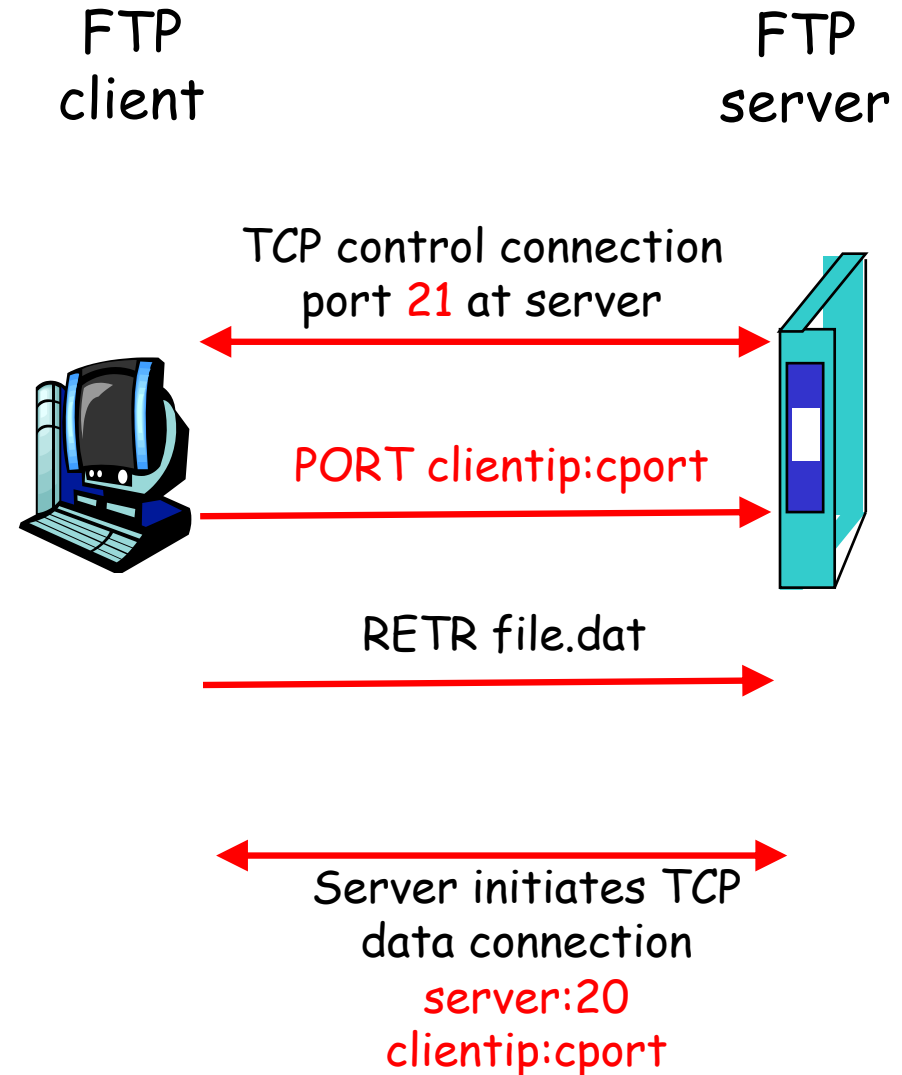
FTP PASV: Server Specifies Data Port, Client Initiates Connection

FTP
client

FTP
server

FTP
client

FTP
server

TCP control connection
port 21 at server

TCP control connection
port 21 at server

PORT clientip:cport

PASV

serverip:sport

RETR file.dat

RETR file.dat

Server initiates TCP
data connection
server:20
clientip:cport

Client initiates TCP
data connection
of PASV returned
serverip:sport
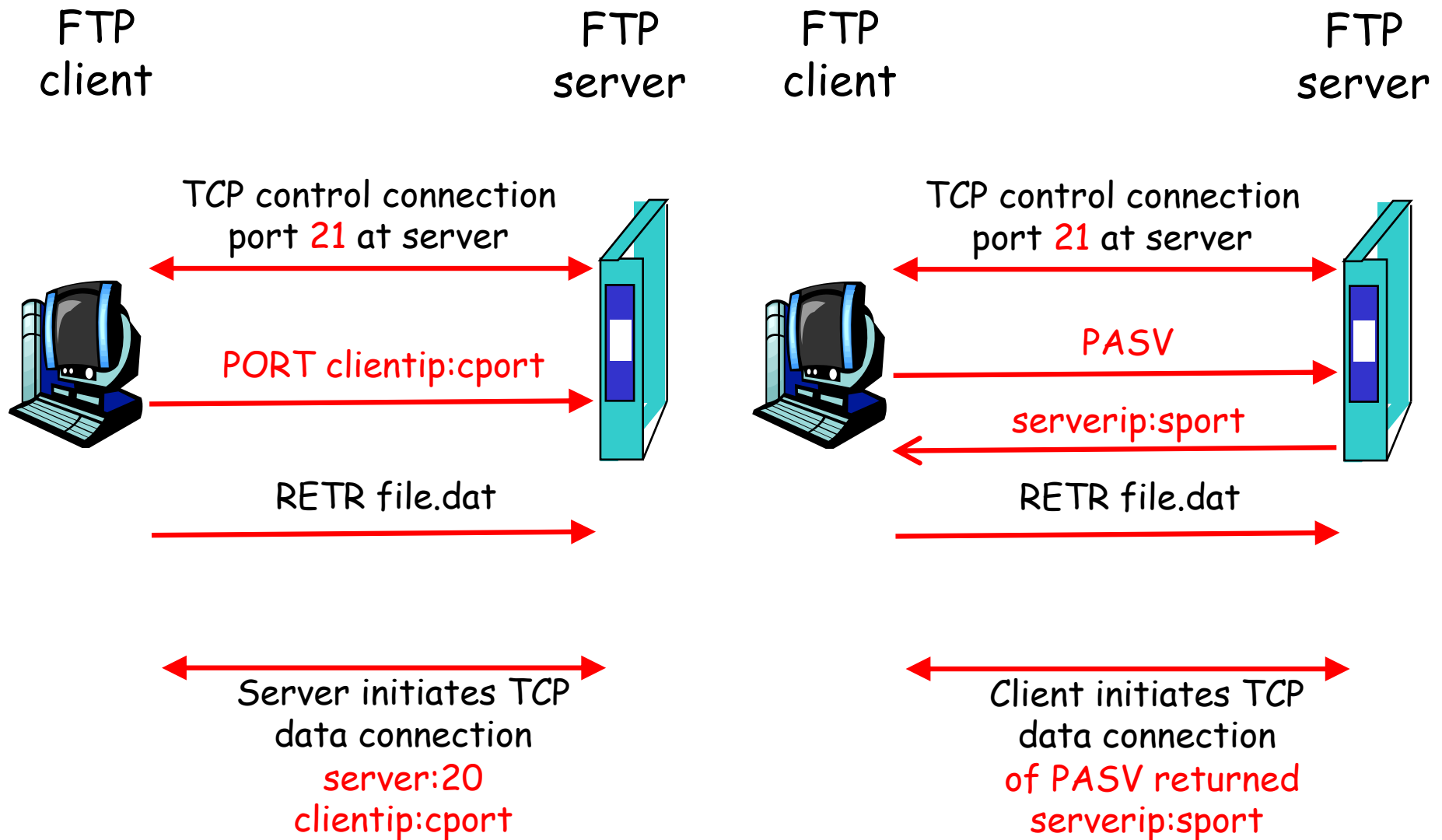
Example

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

FTP Extensions

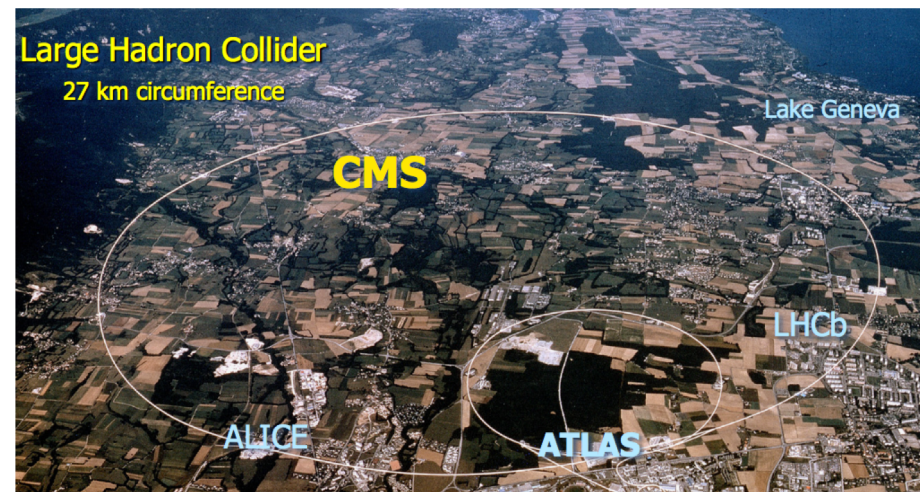- FTP with extensions are being used extensively in large data set transfers (e.g., LHC)

Large Hadron Collider
27 km circumference

Lake Geneva

CMS

LHCb

ALICE

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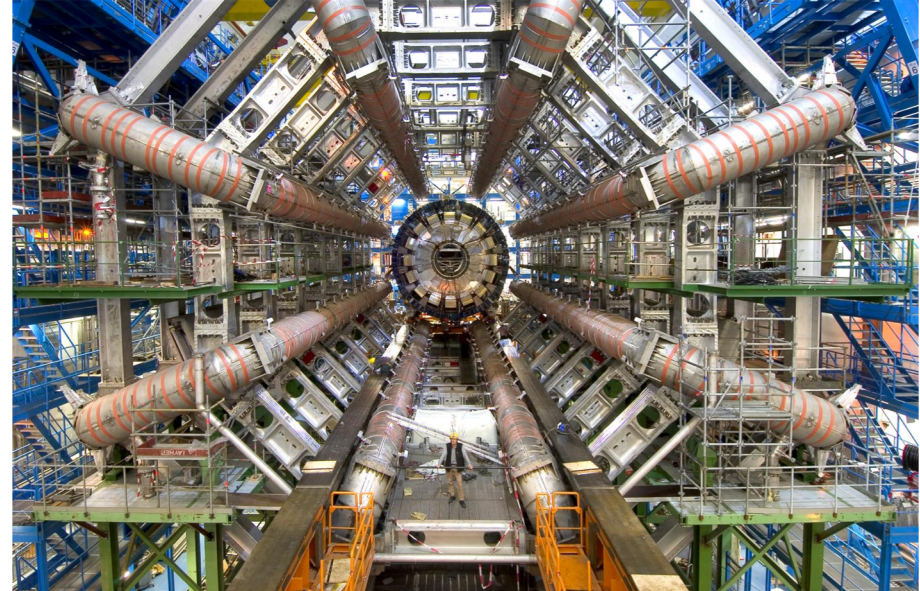
Data Transfer Structure
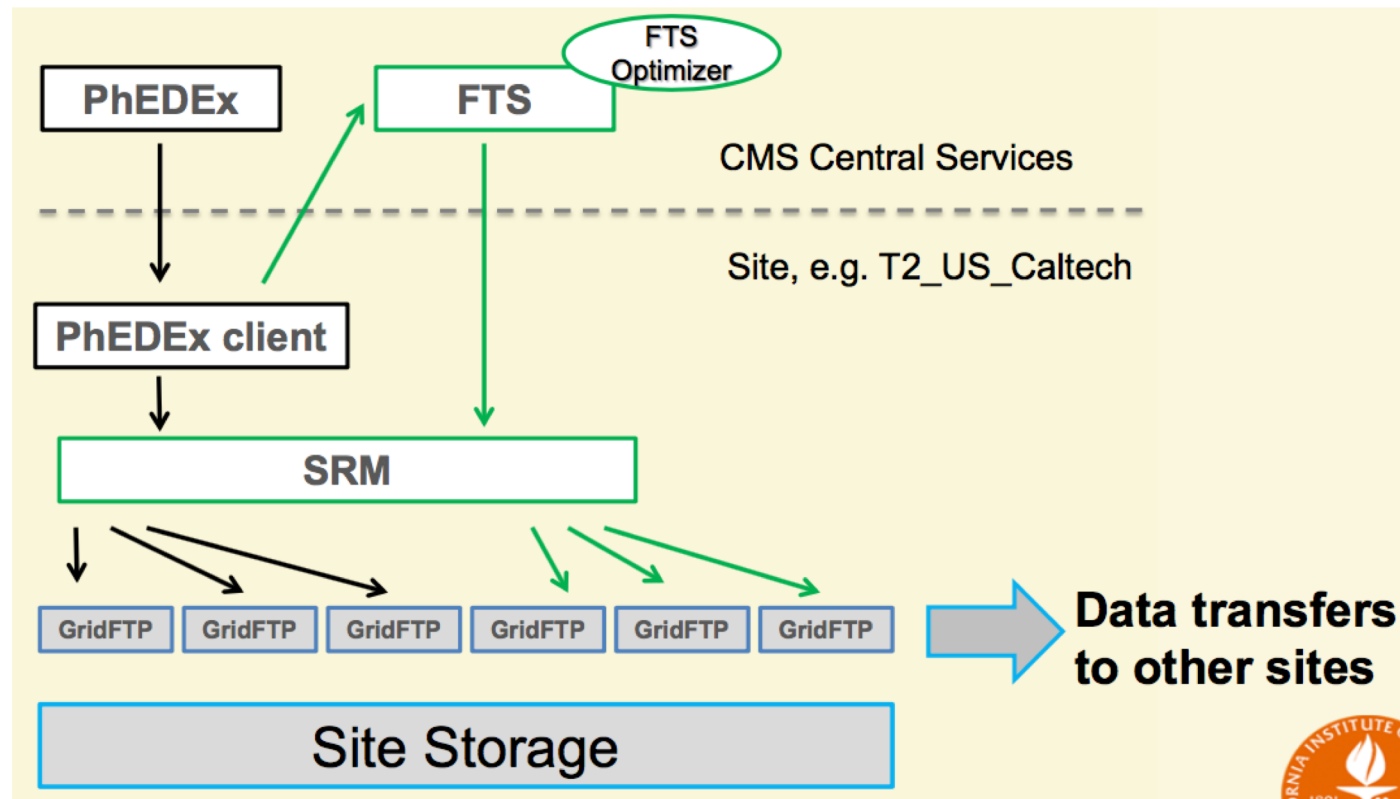
- ❑ See GridFTP to FTP extensions
  - o https://www.ogf.org/documents/GFD.20.pd
- ❑ Goal of GridFTP: allow parallel, high-throughput data transfer
  - o Discussion: What features do we need to add to FTP to allow parallel transfers?

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?
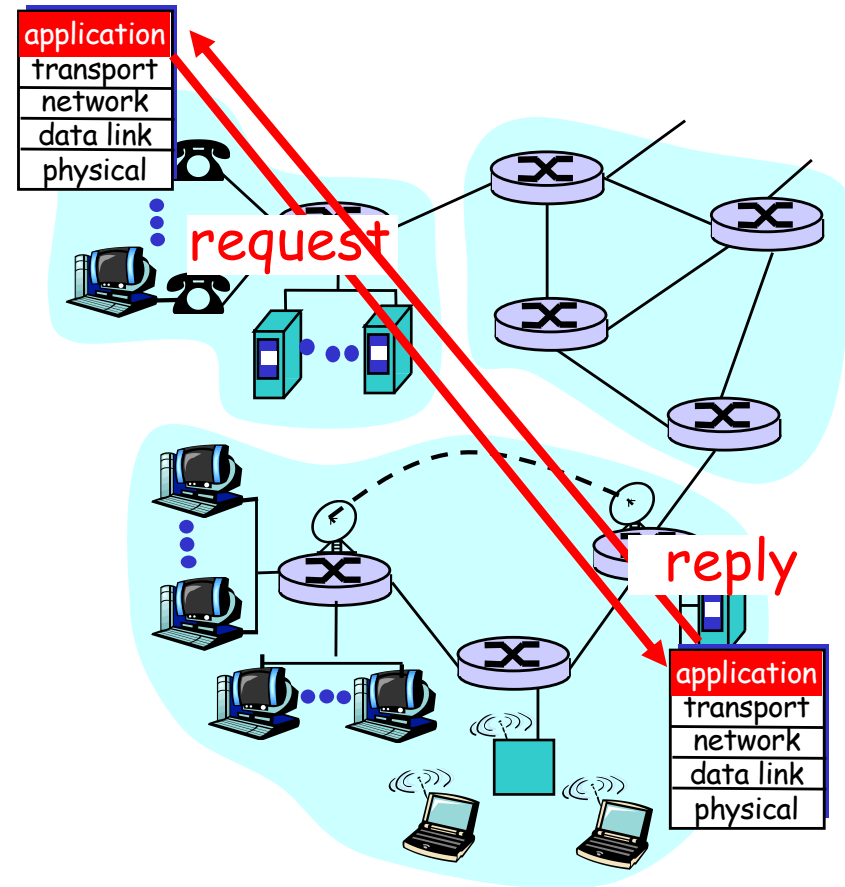
application
transport
network
data link
physical

request

reply

application
transport
network
data link
physical

Outline

❑ Admin. and recap

❑ Network application programming

    o UDP sockets

    o TCP sockets

❑ Network applications (continue)

    ➢ File transfer (FTP) and extension

    ➢ *HTTP*

From Opaque Files to Web Pages

❑ Web page:
- o authored in HTML
- o 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:
- o base HTML page, and
- o several referenced objects
  - o E.g., image

User agent: Explorer

http request

http response

http request

http response

Server running Apache Web server

User agent: Firefox
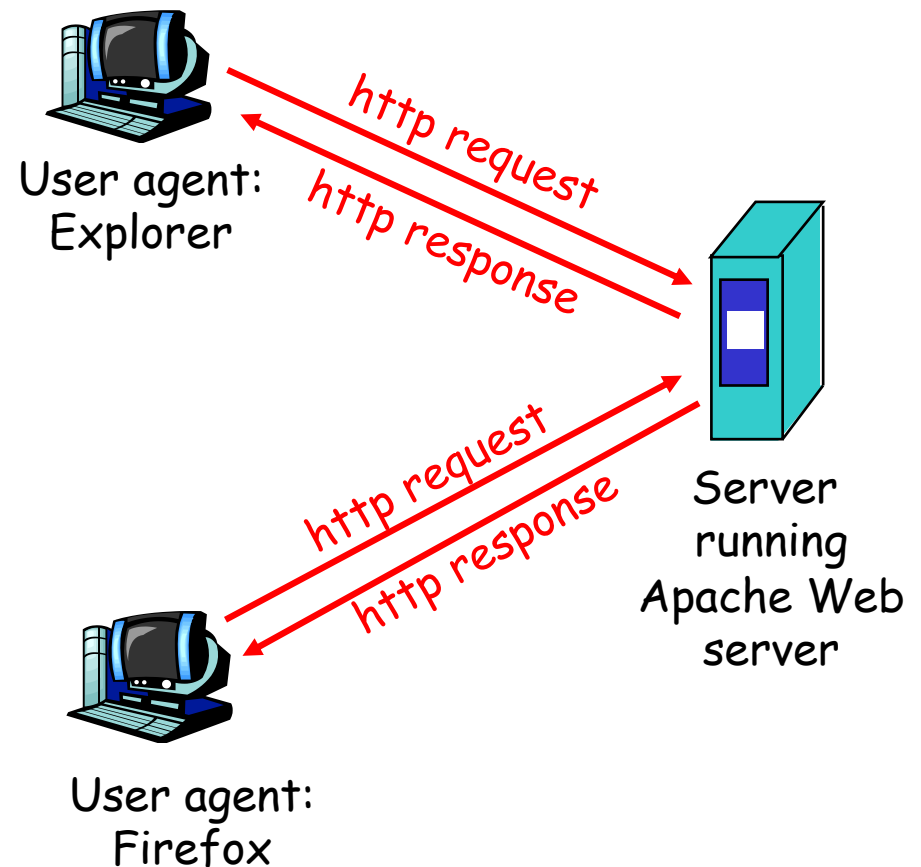
The Web pages are requested through HTTP: hypertext transfer protocol

HTTP is Still Evolving

HTTP/0.9   HTTP/1.0 RFC 1945   HTTP/1.1 RFC 2068   HTTP/2 RFC 7540

1991   1996   1999   2015
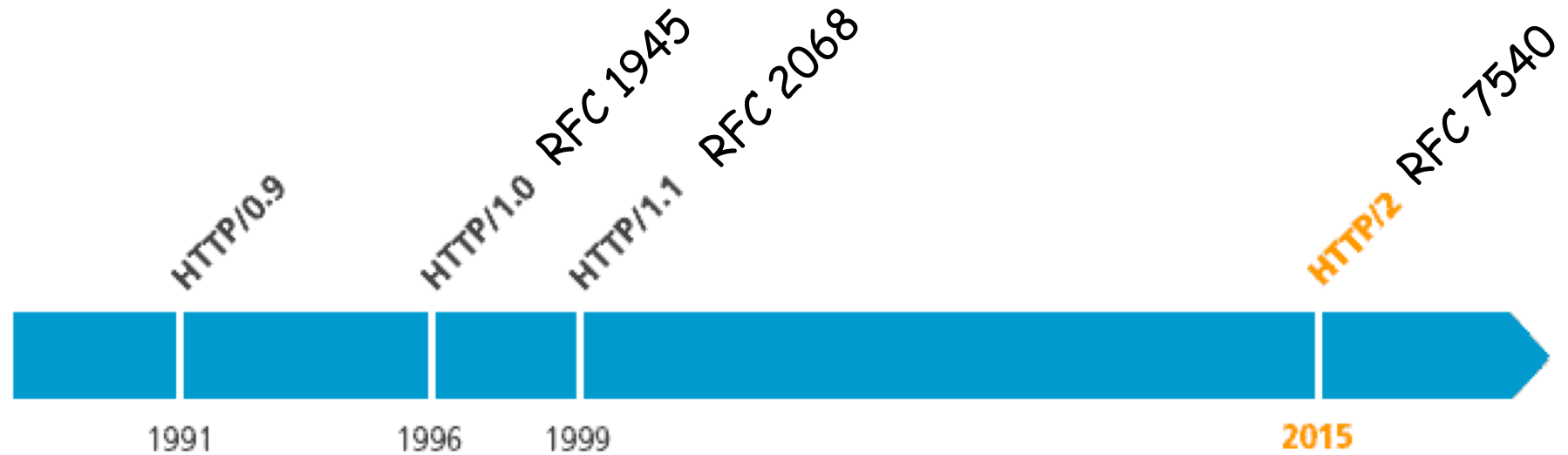
HTTP 1.0 Message Flow

❑ 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)
   o repeat above for each image

21

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

22

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?

FTP client — TCP control connection port 21 at server → FTP server

PORT clientip:cport

RETR index.html

Server initiates TCP data connection
server:20
clientip:cport

FTP client — TCP control connection port 21 at server → FTP server

PASV

serverip:sport

RETR index.html

Client initiates TCP data connection
of PASV returned
serverip:sport
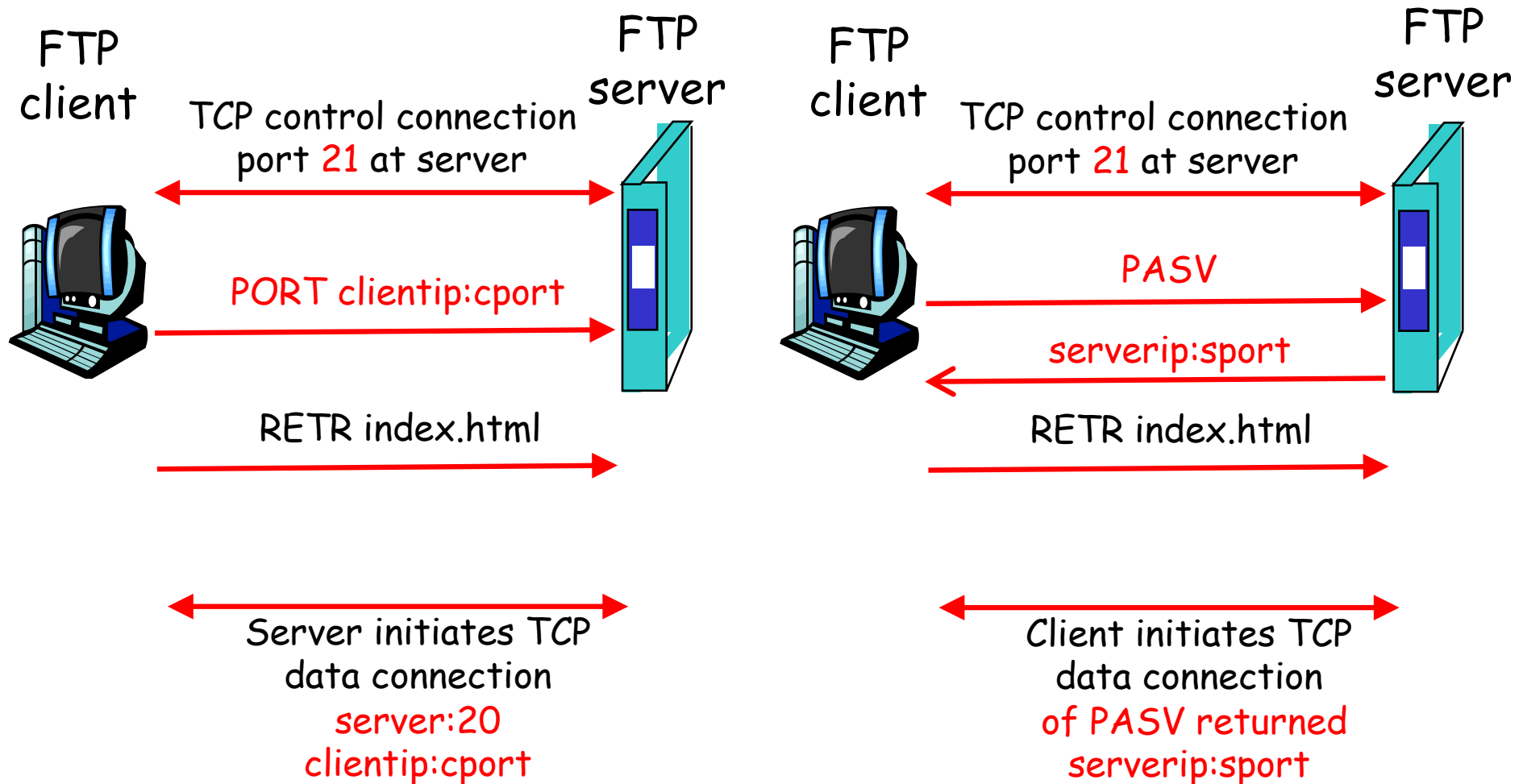
HTTP1.0 Message Flow

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

FTP client   FTP server   HTTP client   HTTP server

USER xxx

PASS xxx

CWD home

PORT clientip:cport

RETR index.html

Server initiates TCP data connection
server:20
clientip:cport

GET /home/index.html
USER: xxx
PASS: xxx

Server sends file on same connection
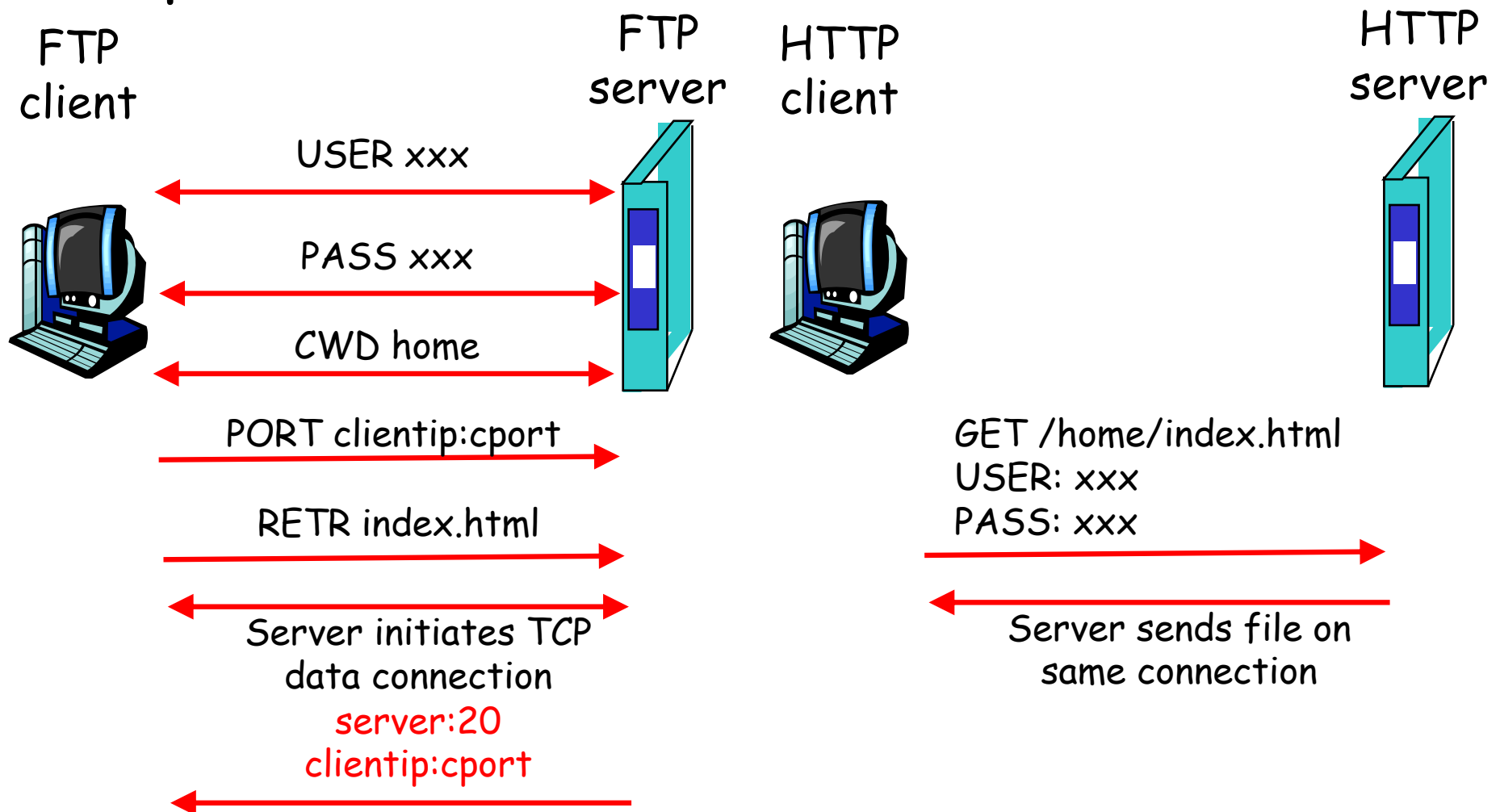
Trying out HTTP (client side) for yourself

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

Virtual host multiplexing

request line
(GET, POST,
HEAD, PUT,
DELETE,
TRACE … commands)

```
GET /somedir/page.html HTTP/1.0
Host: www.somechool.edu
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: en
```

header
lines

Connection management

Carriage return,
line feed
indicates end
of message

(extra carriage return, line feed)

Content negotiation

28

HTTP Response Message

status line
(protocol
status code
status phrase)

```
HTTP/1.0 200 OK
Date: Wed, 23 Jan 2008 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
```

header
lines

```
data data data data data ...
```

data, e.g.,
requested
html file

HTTP Response Status Codes

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

200 OK

 o request succeeded, requested object later in this message

301 Moved Permanently

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

400 Bad Request

 o request message not understood by server

404 Not Found

 o 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.someschool.edu
```

Basic HTTP Server Workflow

Create
ServerSocket(6789)

connSocket = accept()

read request from
connSocket

Map URL to file

Read from file/
write to connSocket

close connSocket

128.36.232.5
128.36.230.2

TCP socket space

state: listening
address: {*.**6789**, *.*}
completed connection queue:
sendbuf:
recvbuf:

state: established
address: {128.36.232.5:**6789**, 198.69.10.10.**1500**}
sendbuf:
recvbuf:

state: listening
address: {*.**25**, *.*}
completed connection queue:
sendbuf:
recvbuf:

Example Code

❑ See BasicWebServer.java

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

  what difference in behavior?

Static -> Dynamic Content

```
┌─────────────────────────┐
│         Create          │
│   ServerSocket(6789)     │
└─────────────────────────┘
           │
           ▼
┌─────────────────────────┐
│  connSocket = accept()   │
└─────────────────────────┘
           │
           ▼
┌─────────────────────────┐
│   read request from      │
│       connSocket         │
└─────────────────────────┘
           │
           ▼
┌─────────────────────────┐
│     Map URL to file      │
└─────────────────────────┘
           │
           ▼
┌─────────────────────────┐
│     Read from file/      │
│   write to connSocket    │
└─────────────────────────┘
           │
           ▼
┌─────────────────────────┐
│     close connSocket     │
└─────────────────────────┘
```
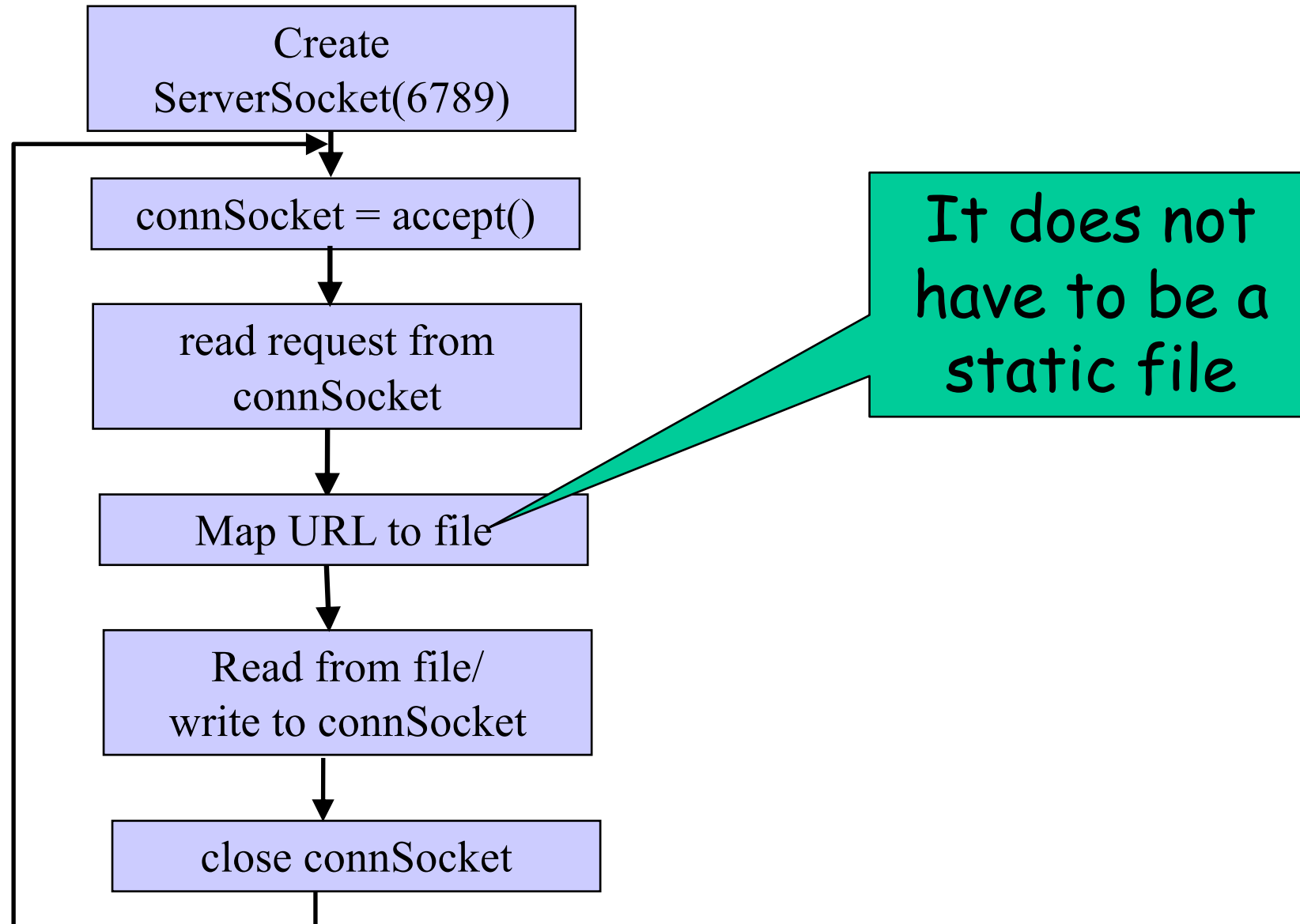
It does not have to be a static file

Outline

❑ Admin and recap

❑ HTTP/1.0

    o Basic design

    ➢ *Dynamic content*

Dynamic Content Pages

❑ **There are multiple approaches to make dynamic web pages:**

    o **Embed code into pages (server side include)**

        • http server includes an interpreter for the type of pages

    o **Invoke external programs (http server is agnostic to the external program execution)**

        • E.g., Common Gateway Interface (CGI)

```
http://www.cs.yale.edu/index.shtml
http://www.cs.yale.edu/cgi-bin/ureserve.pl
http://www.google.com/search?q=Yale&sourceid=chrome
```

Example SSI

❑ See programming/examples-java-socket/BasicWebServer/ssi/index.shtml, header.shtml, …