DATA COMMUNICATOIN NETWORKING

Instructor: Ouldooz Baghban Karimi

Course Book: Computer Networking, A Top-Down Approach
By: Kurose, Ross

Course Overview

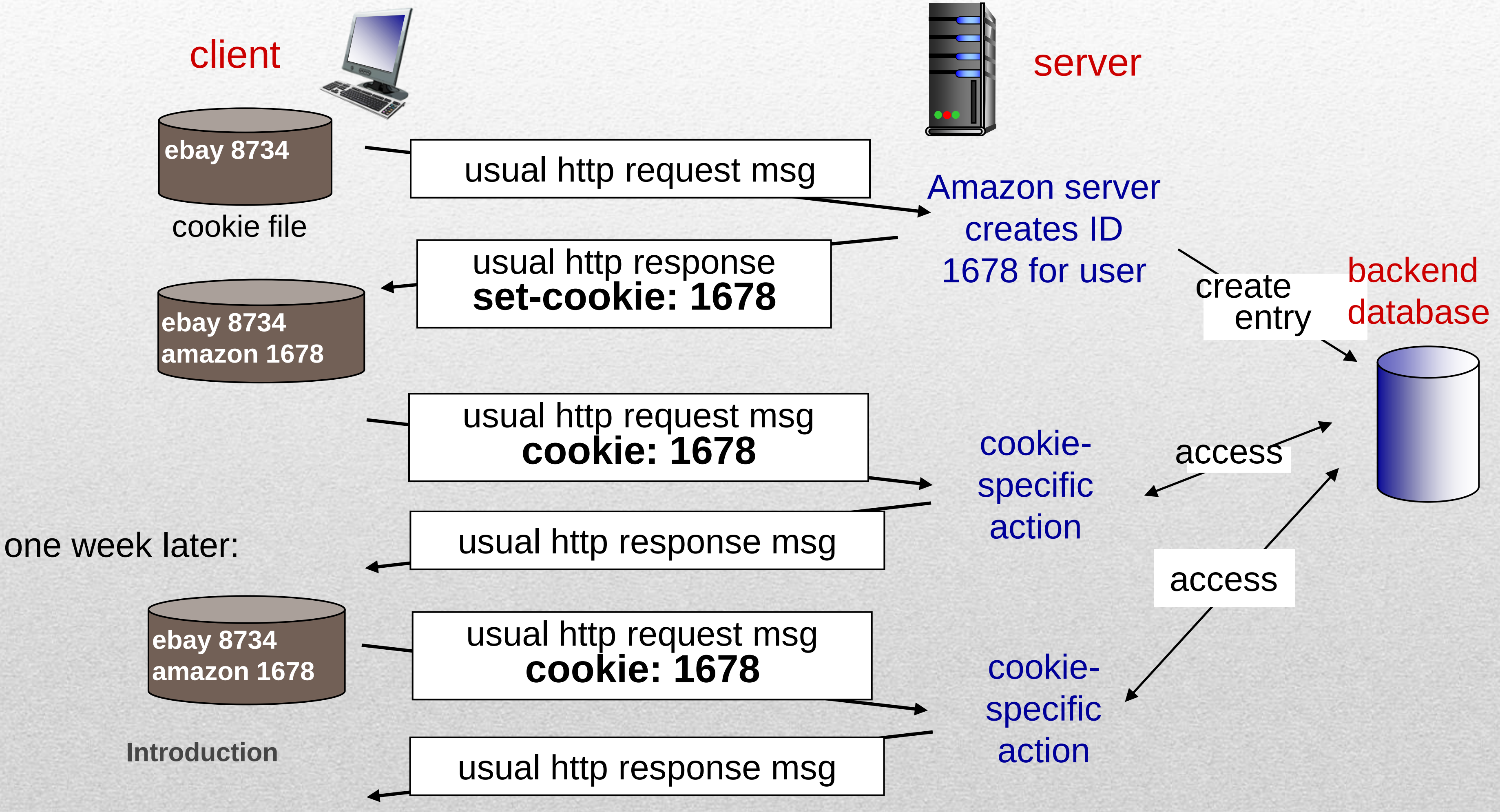
- **Basics of Computer Networks**
  - Internet & Protocol Stack
  - Application Layer
  - Transport Layer
  - Network Layer
  - Data Link Layer

- **Advanced Topics**
  - Case Studies of Computer Networks
  - Internet Applications
  - Network Management
  - Network Security

User-Server State: Cookies

- **Four Components**
  - Cookie header line of HTTP response
  - Cookie header line in next HTTP *request* message
  - Back-end database at Web site
  - Cookie file kept on user's host, managed by user's browser

Cookies


client
server
ebay 8734
cookie file
usual http request msg
Amazon server creates ID 1678 for user
usual http response
set-cookie: 1678
create entry
backend database
ebay 8734
amazon 1678
usual http request msg
cookie: 1678
cookie-specific action
access
usual http response msg
one week later:
access
ebay 8734
amazon 1678
usual http request msg
cookie: 1678
cookie-specific action
usual http response msg

Cookies Usage

- **Usage**
  - Authorization
  - Shopping Carts
  - Recommendations
  - User session state (Web email)

cookies and privacy

cookies permit sites to learn a lot about you

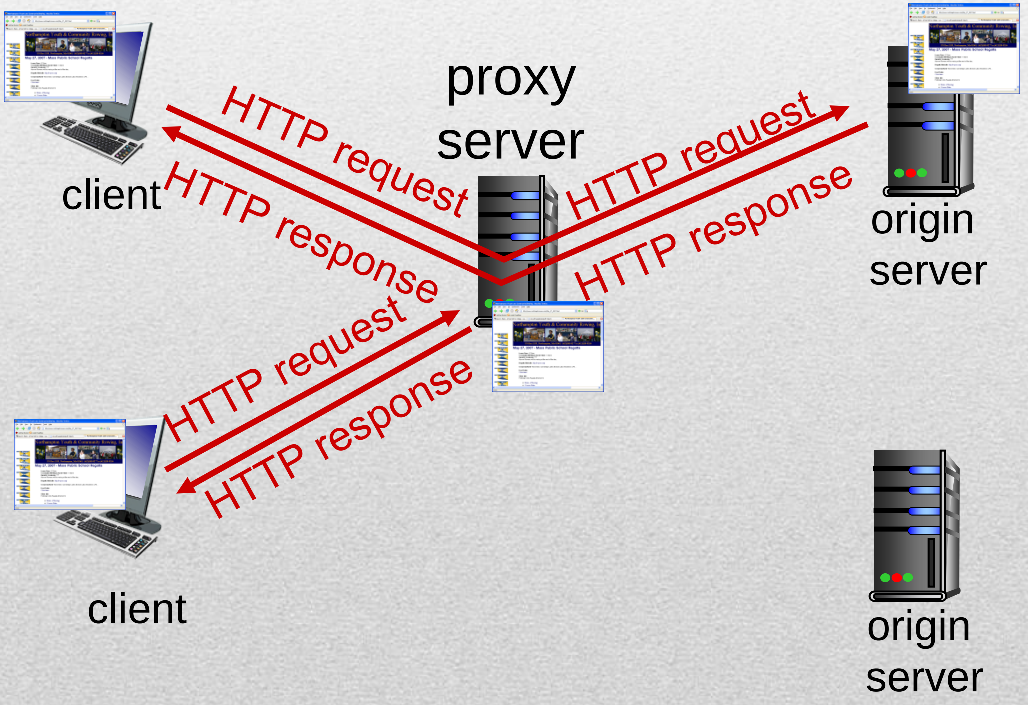
you may supply name and e-mail to sites

- **Keeping State**
  - protocol endpoints: maintain state at sender/receiver over multiple transactions
  - cookies: http messages carry state

Web Caches (Proxy Server)

Goal: satisfy client request without involving origin server

- **User sets browser**
  - Web accesses via cache
- **Browser sends all HTTP requests to cache**
  - Object in cache
    - cache returns object
  - Else cache requests object from origin server, then returns object to client

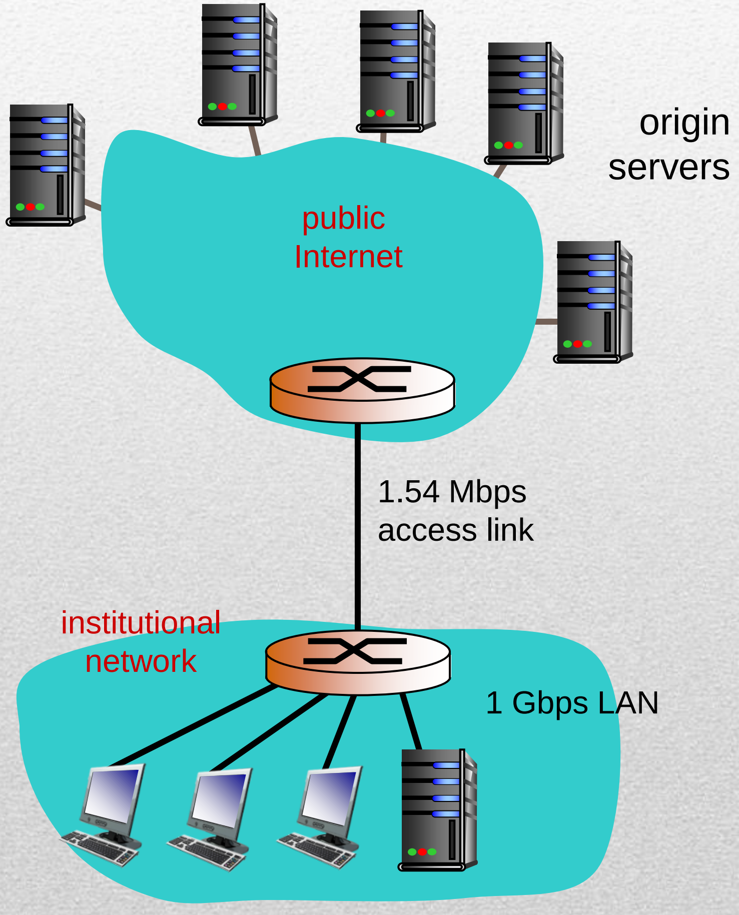


Web Catching

- **Cache acts as both client and server**
  - server for original requesting client
  - client to origin server
- **Typically cache is installed by ISP**
  - university, company, residential ISP
- **Why Web Catching?**
  - Reduce response time for client request
  - Reduce traffic on an institution’s access link
  - Internet dense with caches: enables “poor” content providers to effectively deliver content (so too does P2P file sharing)

Web Catching Example

- **Example**
  - Average object size: 100kbits
  - Average request rate from browser to origin servers: 15/sec
  - Average Data rate to Browsers: 1.50Mbps
  - RTT from institutional router to any high origin server: 2sec
  - Access link rate: 1.54Mbps



Web Catching Example

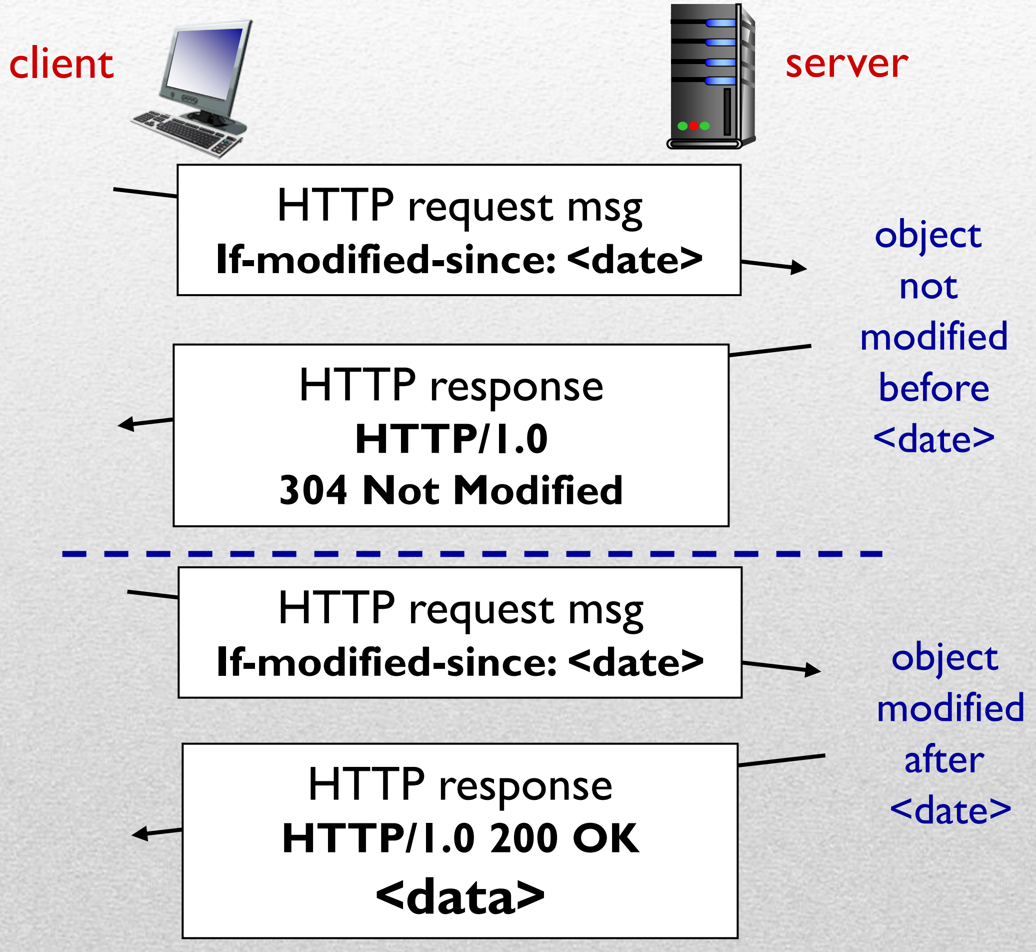
- **Example**
  - LAN utilization: 15%
  - Access link utilization = 99%
  - Total delay = Internet delay + access delay + LAN delay
    = 2 sec + minutes + usecs

- **Increase access link speed to 154Mbps**
  - LAN utilization: 15%
  - Access link utilization = 9.9%
  - Total delay = Internet delay + access delay + LAN delay
    = 2 sec + msecs + usecs

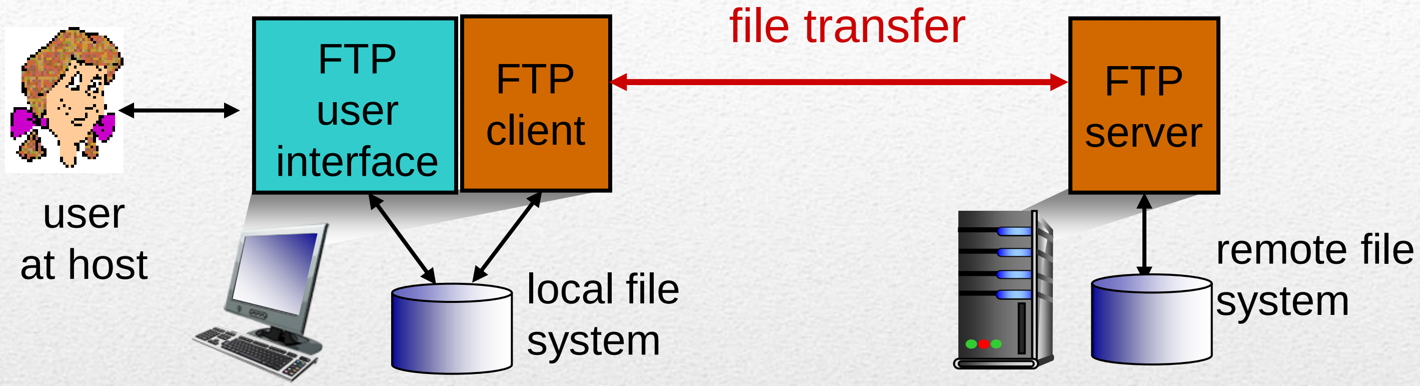
- **Cache with hit rate 0.4**
  - Access link utilization:
    - 60% of requests use access link
      - Data rate to browsers over access link = 0.6*1.50 Mbps = .9 Mbps
      - Utilization = 0.9/1.54 = 0.58
  - Total delay
    - = 0.6 * (delay from origin servers) +0.4 * (delay when satisfied at cache) = 0.6 (~2.01) + 0.4 (~msecs) = ~ 1.2 secs
    - less than with 154 Mbps link (and cheaper too!)

Conditional GET

- ***Goal:*** don't send object if cache has up-to-date cached version
  - no object transmission delay
  - lower link utilization
- *Cache:* specify date of cached copy in HTTP request
  - **If-modified-since: <date>**
- *Server:* response contains no object if cached copy is up-to-date:
  - **HTTP/1.0 304 Not Modified**



FTP: File Transfer Protocol

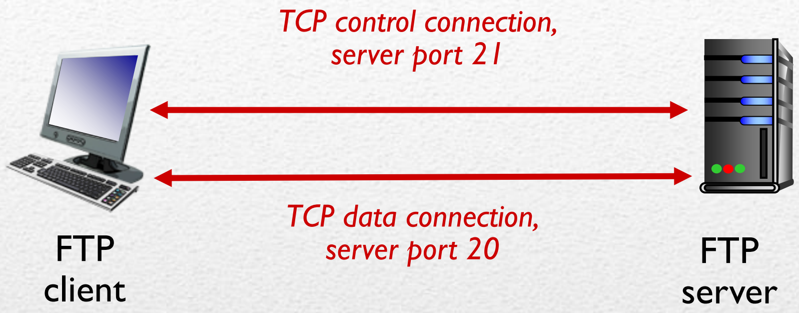


- **Transfer to/from remote host**
- **Client/server model**
  - *Client:* side that initiates transfer (either to/from remote)
  - *Server:* remote host

- ftp: RFC 959
- ftp server: port 21

FTP: Separate Control & Data

- FTP client contacts FTP server at port 21, using TCP
- Client authorized over control connection
- Client browses remote directory, sends commands over control connection
- When server receives file transfer command, server opens 2nd TCP data connection (for file) *to* client
- After transferring one file, server closes data connection



- Server opens another TCP data connection to transfer another file
- control connection: *"out of band"*
- FTP server maintains "state": current directory, earlier authentication

FTP: Commands & Responses

sample commands:

- sent as ASCII text over control channel

- `USER` ***`username`***
- `PASS` ***`password`***

- **`LIST`** return list of file in current directory

- **`RETR filename`** retrieves (gets) file

- **`STOR filename`** stores (puts) file onto remote host

sample return codes:

- status code and phrase (as in HTTP)

- **`331 Username OK, password required`**

- **`125 data connection already open; transfer starting`**

- **`425 Can't open data connection`**

- **`452 Error writing file`**