15-441: Computer Networking

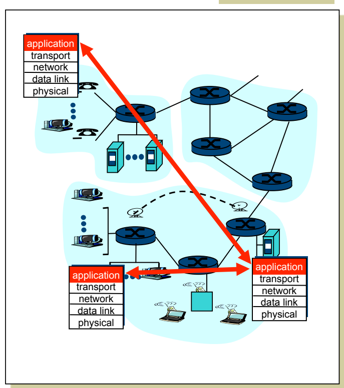
Lecture 3: Design Philosophy & Applications

Lecture Overview

- Last time:
  - Protocol stacks and layering
  - OSI and TCP/IP models
- Application requirements
- Application examples
  - ftp
  - http
- Internet Architecture & Performance intro

Applications and Application-Layer Protocols

- Application: communicating, distributed processes
  - Running in network hosts in "user space"
  - Exchange messages to implement app
  - e.g., email, file transfer, the Web
- Application-layer protocols
  - One "piece" of an app
  - Define messages exchanged by apps and actions taken
  - User services provided by lower layer protocols



Client-Server Paradigm

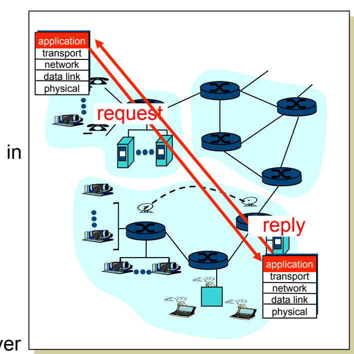
Typical network app has two pieces: *client* and *server*

Client:

- Initiates contact with server ("speaks first")
- Typically requests service from server,
- For Web, client is implemented in browser; for e-mail, in mail reader

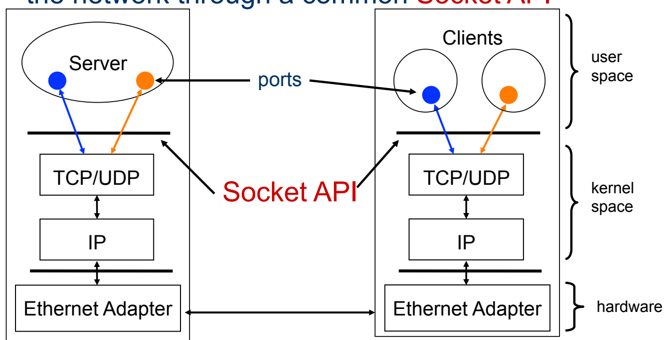
Server:

- Provides requested service to client
- e.g., Web server sends requested Web page, mail server delivers e-mail

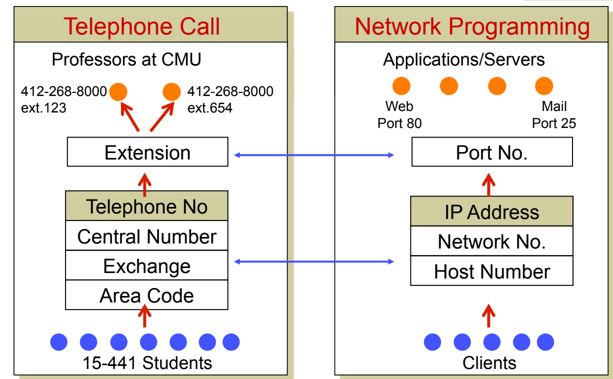


Server and Client

Server and Client exchange messages over the network through a common Socket API



Network Addressing Analogy



What Service Does an Application Need?

Data loss

Timing

Bandwidth

Transport Service Requirements of Common Apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
web documents	no loss	elastic	no
real-time audio/ video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps	yes, 100's msec
financial apps	no loss	elastic	yes and no

Other Requirements

- Network reliability
  - Network service must always be available
- Security: privacy, denial of service, authentication, …
- Scalability.
  - Scale to large numbers of users, traffic flows, …
- Manageability: monitoring, control, …

User Datagram Protocol(UDP): An Analogy

UDP	Postal Mail
• Single socket to receive messages • No guarantee of delivery • Not necessarily in-order delivery • Datagram – independent packets • Must address each packet	• Single mailbox to receive letters • Unreliable ☺ • Not necessarily in-order delivery • Letters sent independently • Must address each reply

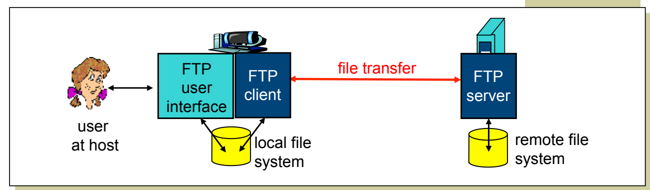
Example UDP applications
Multimedia, voice over IP

Transmission Control Protocol (TCP): An Analogy

TCP	Telephone Call
• Reliable – guarantee delivery • Byte stream – in-order delivery • Connection-oriented – single socket per connection • Setup connection followed by data transfer	• Guaranteed delivery • In-order delivery • Connection-oriented • Setup connection followed by conversation

Example TCP applications
Web, Email, Telnet

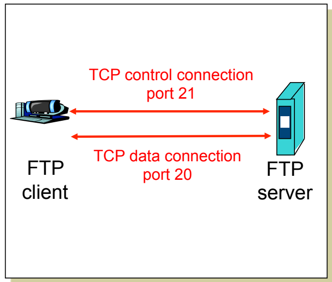
FTP: The File Transfer Protocol



- Transfer file 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 Connections

- Ftp client contacts ftp server at port 21, specifying TCP as transport protocol
- Two parallel TCP connections opened:
  - Control: exchange commands, responses between client, server.
    "out of band control"
  - Data: file data to/from server
- 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 files in current directory
- `RETR filename` retrieves (gets) file
- `STOR filename` stores (puts) file onto remote host

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`

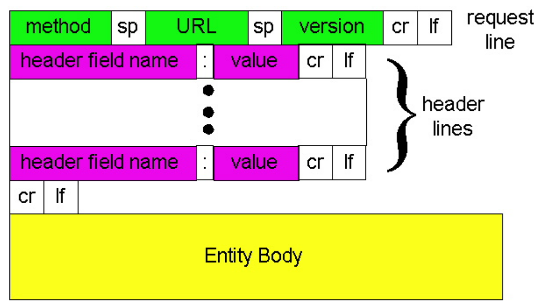
HTTP Basics

- HTTP layered over bidirectional byte stream
  - Almost always TCP
- Interaction
  - Client sends request to server, followed by response from server to client
  - Requests/responses are encoded in text
- Stateless
  - Server maintains no information about past client requests

How to Mark End of Message?

- Size of message → Content-Length
  - Must know size of transfer in advance
- Delimiter → MIME style Content-Type
  - Server must "escape" delimiter in content
- Close connection
  - Only server can do this

HTTP Request



HTTP Request

- Request line
  - Method
    - GET – return URI
    - HEAD – return headers only of GET response
    - POST – send data to the server (forms, etc.)
  - URI
    - E.g. http://www.intel-iris.net/index.html with a proxy
    - E.g. /index.html if no proxy
  - HTTP version

HTTP Request

- Request headers
  - Authorization – authentication info
  - Acceptable document types/encodings
  - From – user email
  - If-Modified-Since
  - Referrer – what caused this page to be requested
  - User-Agent – client software
- Blank-line
- Body

HTTP Request Example

GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.intel-iris.net
Connection: Keep-Alive

HTTP Response

- Status-line
  - HTTP version
  - 3 digit response code
    - 1XX – informational
    - 2XX – success
      - 200 OK
    - 3XX – redirection
      - 301 Moved Permanently
      - 303 Moved Temporarily
      - 304 Not Modified
    - 4XX – client error
      - 404 Not Found
    - 5XX – server error
      - 505 HTTP Version Not Supported
  - Reason phrase

HTTP Response

- Headers
  - Location – for redirection
  - Server – server software
  - WWW-Authenticate – request for authentication
  - Allow – list of methods supported (get, head, etc)
  - Content-Encoding – E.g x-gzip
  - Content-Length
  - Content-Type
  - Expires
  - Last-Modified
- Blank-line
- Body

HTTP Response Example

HTTP/1.1 200 OK
Date: Tue, 27 Mar 2001 03:49:38 GMT
Server: Apache/1.3.14 (Unix) (Red-Hat/Linux) mod_ssl/2.7.1 OpenSSL/0.9.5a DAV/1.0.2 PHP/4.0.1pl2 mod_perl/1.24
Last-Modified: Mon, 29 Jan 2001 17:54:18 GMT
ETag: "7a11f-10ed-3a75ae4a"
Accept-Ranges: bytes
Content-Length: 4333
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
…..

Cookies: Keeping "state"

Many major Web sites use cookies

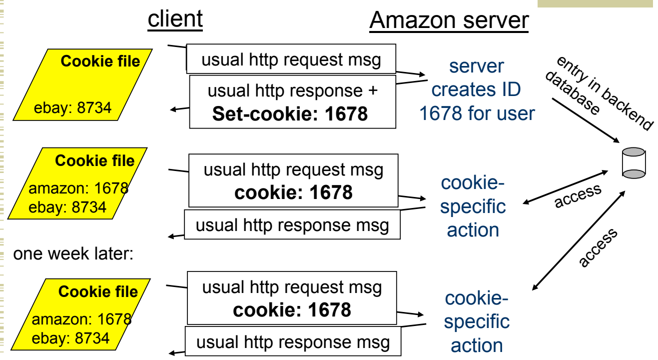
Four components:

1) Cookie header line in the HTTP response message
2) Cookie header line in HTTP request message
3) Cookie file kept on user's host and managed by user's browser
4) Back-end database at Web site

Example:

- Susan accesses Internet always from same PC
- She visits a specific e-commerce site for the first time
- When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

Cookies: Keeping “State”



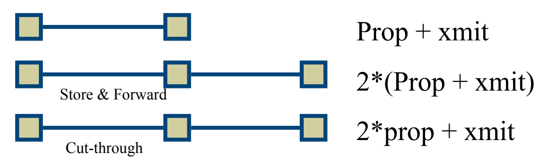
Typical Workload (Web Pages)

- Multiple (typically small) objects per page
- File sizes
  - Why different than request sizes?
  - Also heavy-tailed
    - Pareto distribution for tail
    - Lognormal for body of distribution
- Embedded references
  - Number of embedded objects = pareto – $p(x) = ak^a x^{-(a+1)}$

HTTP 1.1 - new features

- Newer versions of HTTP add several new features (persistent connections, pipelined transfers) to speed things up.
- Let’s detour into some performance evaluation and then look at those features

Packet Delay



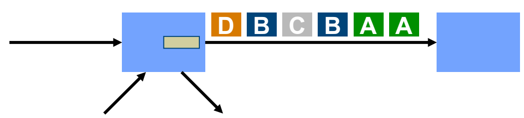
When does cut-through matter?

Next: Routers have finite speed (processing delay)

Routers may buffer packets (queueing delay)

Packet Delay

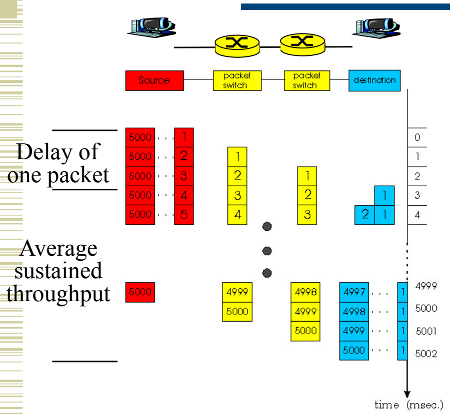
- Sum of a number of different delay components.
- Propagation delay on each link.
  - Proportional to the length of the link
- Transmission delay on each link.
  - Proportional to the packet size and 1/link speed
- Processing delay on each router.
  - Depends on the speed of the router
- Queuing delay on each router.
  - Depends on the traffic load and queue size



A Word about Units

- What do “Kilo” and “Mega” mean?
  - Depends on context
- Storage works in powers of two.
  - 1 Byte = 8 bits
  - 1 KByte = 1024 Bytes
  - 1 MByte = 1024 Kbytes
- Networks work in decimal units.
  - Network hardware sends bits, not Bytes
  - 1 Kbps = 1000 bits per second
  - To avoid confusion, use 1 Kbit/second

Application-level Delay



$$\text{Delay}^* + \frac{\text{Size}}{\text{Throughput}}$$

Units: seconds + bits/(bits/seconds)

* For minimum sized packet

Some Examples

- How long does it take to send a 100 Kbit file?
  - Assume a perfect world
  - And a 10 Kbit file

Throughput / Latency	100 Kbit/s	1 Mbit/s	100 Mbit/s
500 μsec			
10 msec			
100 msec			

Some Examples

- How long does it take to send a 100 Kbit file?
  - Assume a perfect world
  - And a 10 Kbit file

Throughput / Latency	100 Kbit/s	1 Mbit/s	100 Mbit/s
500 μsec	1.0005	0.1005	0.0015
10 msec	1.01	0.11	0.011
100 msec	1.1	0.2	0.101

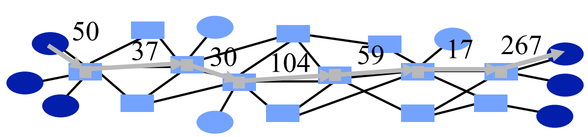
Some Examples

- How long does it take to send a 100 Kbit file?
  - Assume a perfect world
  - And a 10 Kbit file

Throughput / Latency	100 Kbit/s	1 Mbit/s	100 Mbit/s
500 μsec	0.1005	0.0105	0.0006
10 msec	0.11	0.02	0.0101
100 msec	0.2	0.11	0.1001

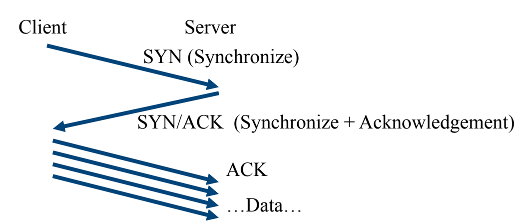
Sustained Throughput

- When streaming packets, the network works like a pipeline.
  - All links forward different packets in parallel
- Throughput is determined by the slowest stage.
  - Called the bottleneck link
- Does not really matter why the link is slow.
  - Low link bandwidth
  - Many users sharing the link bandwidth



One more detail: TCP

- TCP connections need to be set up
  - "Three Way Handshake":

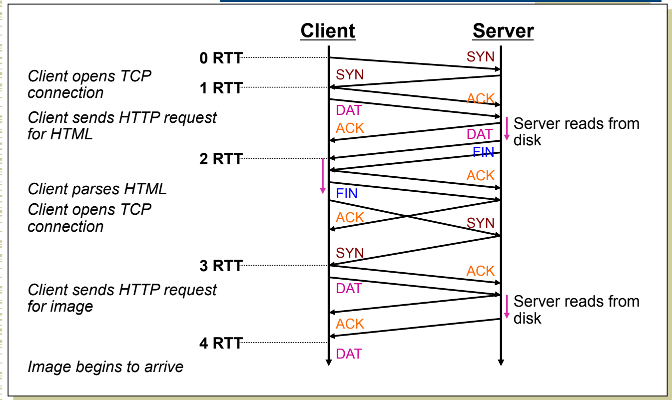


2: TCP transfers start slowly and then ramp up the bandwidth used (so they don't use too much)

HTTP 0.9/1.0

- One request/response per TCP connection
  - Simple to implement
- Disadvantages
  - Multiple connection setups → three-way handshake each time
    - Several extra round trips added to transfer
  - Multiple slow starts

Single Transfer Example



Performance Issues

- Short transfers are hard on TCP
  - Stuck in slow start
  - Loss recovery is poor when windows are small
- Lots of extra connections
  - Increases server state/processing
- Servers also hang on to connection state after the connection is closed
  - Why must server keep these?
  - Tends to be an order of magnitude greater than # of active connections, why?

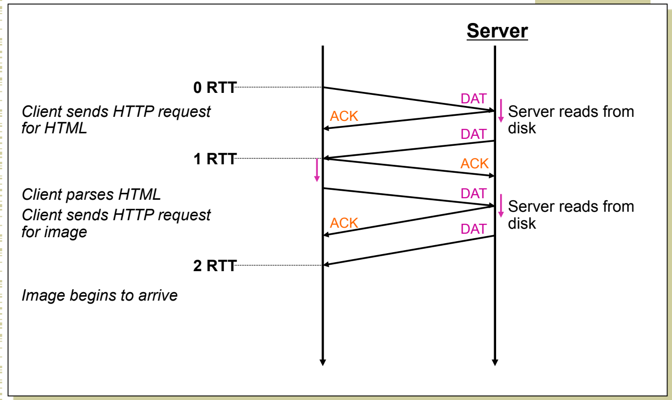
Netscape Solution

- Mosaic (original popular Web browser) fetched one object at a time!
- Netscape uses multiple concurrent connections to improve response time
  - Different parts of Web page arrive independently
  - Can grab more of the network bandwidth than other users
- Doesn't necessarily improve response time
  - TCP loss recovery ends up being timeout dominated because windows are small

Persistent Connection Solution

- Multiplex multiple transfers onto one TCP connection
- How to identify requests/responses
  - Delimiter → Server must examine response for delimiter string
  - Content-length and delimiter → Must know size of transfer in advance
  - Block-based transmission → send in multiple length delimited blocks
  - Store-and-forward → wait for entire response and then use content-length
  - Solution → use existing methods and close connection otherwise

Persistent Connection Solution



Remaining Problems

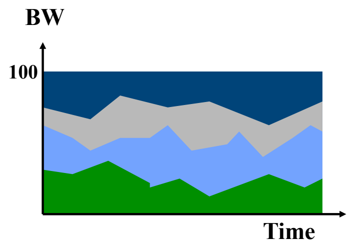
- Serialized transmission
  - Much of the useful information in first few bytes
    - May be better to get the 1st 1/4 of all images than one complete image (e.g., progressive JPEG)
  - Can "packetize" transfer over TCP
    - Could use range requests
- Application specific solution to transport protocol problems. :(
  - Solve the problem at the transport layer
  - Could fix TCP so it works well with multiple simultaneous connections
    - More difficult to deploy

Back to performance

- We examined delay,
- But what about throughput?
- Important factors:
  - Link capacity
  - *Other traffic*

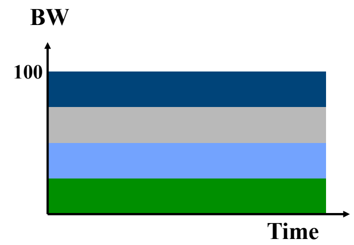
Bandwidth Sharing

- Bandwidth received on the bottleneck link determines end-to-end throughput.
- Router before the bottleneck link decides how much bandwidth each user gets.
  - Users that try to send at a higher rate will see packet loss
- User bandwidth can fluctuate quickly as flows are added or end, or as flows change their transmit rate.

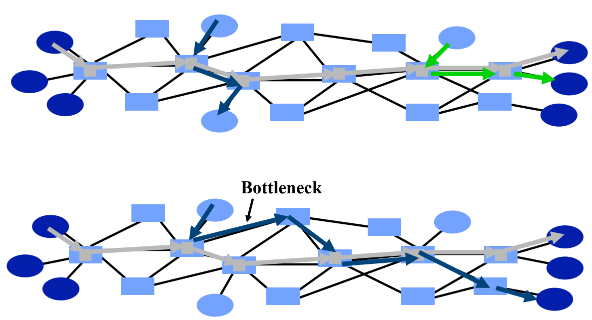


Fair Sharing of Bandwidth

- All else being equal, fair means that users get equal treatment.
  - Sounds fair
- When things are not equal, we need a policy that determines who gets how much bandwidth.
  - Users who pay more get more bandwidth
  - Users with a higher "rank" get more bandwidth
  - Certain classes of applications get priority



But It is Not that Simple



Today

- Application layer
- Each application needs different services e.g., data loss? Elastic? Timing?
- FTP
- HTTP
  - Interaction with TCP: Persistant? Pipelining?
- Delay/Throughput

Goals [Clark88]

0 **Connect existing networks**

initially ARPANET and ARPA packet radio network

1. Survivability

   ensure communication service even in the presence of network and router failures

2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

Other IP Design Weaknesses

- Weak administration and management tools
- Incremental deployment difficult at times
  - Result of no centralized control
  - No more "flag" days

Changes Over Time → New Principles?

- Developed in simpler times
  - Common goals, consistent vision
- With success came multiple goals – examples:
  - ISPs must talk to provide connectivity but are fierce competitors
  - Privacy of users vs. government's need to monitor
  - User's desire to exchange files vs. copyright owners
- Must deal with the tussle between concerns in design
- Provide choice → allow all parties to make choices on interactions
  - Creates competition
  - Fear between providers helps shape the tussle

New Principles?

- Design for variation in outcome
  - Allow design to be flexible to different uses/results
- Isolate tussles
  - QoS designs uses separate ToS bits instead of overloading other parts of packet like port number
  - Separate QoS decisions from application/protocol design
- Provide choice → allow all parties to make choices on interactions
  - Creates competition
  - Fear between providers helps shape the tussle

Summary: Internet Architecture

- Packet-switched datagram network
- IP is the "compatibility layer"
  - Hourglass architecture
  - All hosts and routers run IP
- Stateless architecture
  - no per flow state inside network