15-441 15-641 : Computer Networking

Lecture 3: Design Philosophy & Applications

Peter Steenkiste

Fall 2010

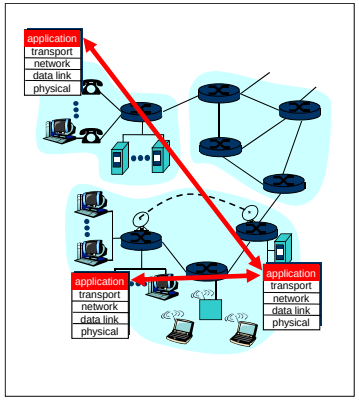
www.cs.cmu.edu/~prs/15-441-F10

Lecture Overview

- Application requirements
- Application examples
  - ftp
  - http
  - Today's applications
- Network performance

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

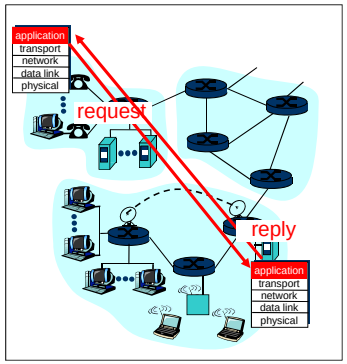
Typical network app has two pieces: *client* and *server*

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

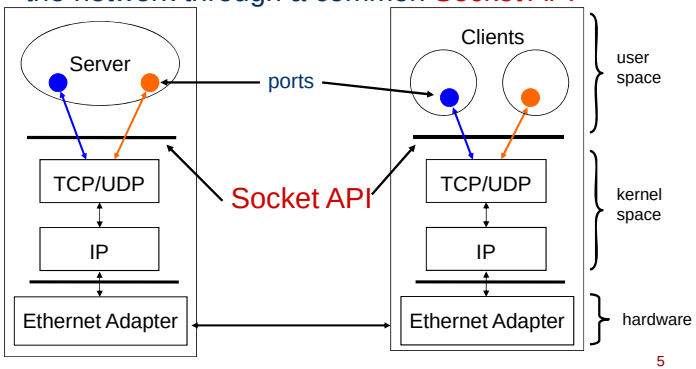
Server:

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

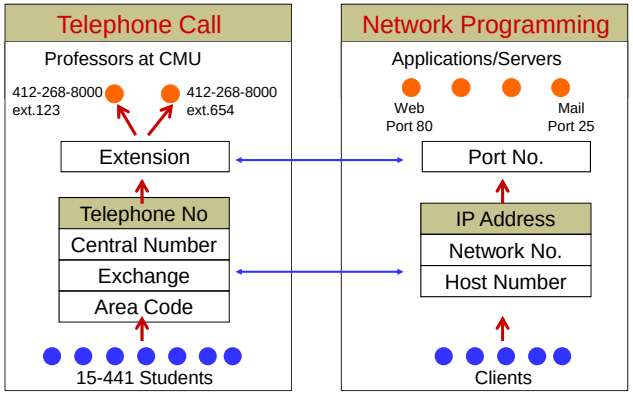


Server and Client

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, billing, …

9

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

10

Transmission Control Protocol (TCP): An Analogy

TCP	Telephone Call
• Reliable – guaranteed 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

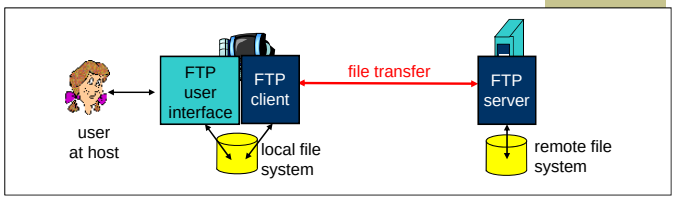
11

Some Example Applications

- A basic early application – ftp
- The first popular app – http
- Today's apps – lots …

12

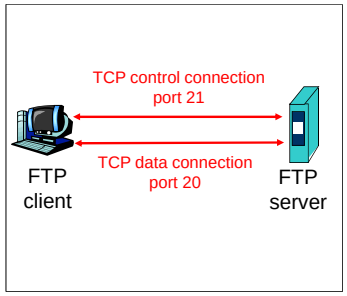
FTP: The File Transfer Protocol



- 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`

Performance Considerations?

- Time for request to be completed
- Typically dominated by the file transfer time
  - Is likely to be determined by the bandwidth
  - But latency could dominate – later today

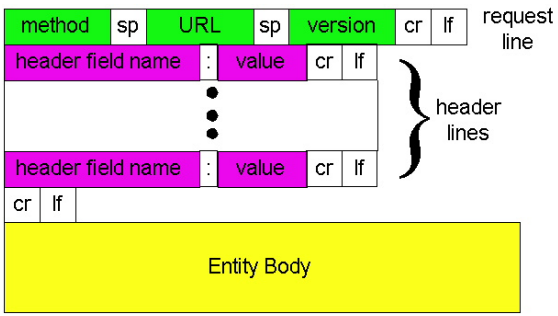
HTTP Basics

- 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
  - HTTP protocol is stateless – application logic is likely to want to know about you

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

21

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

22

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

23

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

24

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
.....
```

25

Performance Considerations?

- Time for web page to be shown on the display
  - Or at least the part the user cares about
- Similar to ftp, except …
  - Web pages typically have many components, represented as files
  - Many components are generated dynamically
  - Components can come from many different content providers and servers

26

Today's Applications

- Amazon, Facebook, etc.
  - What matters most?
  - 2009 quote: "*Amazon* found every 100ms of *latency* cost them 1% in sales"
- Video streaming
  - Accounts for very high percentage of bandwidth
  - Interactive versus broadcast versus playback
  - What matters most?
- Skype audio and video conferencing
  - Traditional telephone app
  - What matters most?

31

Some More Specialized Apps

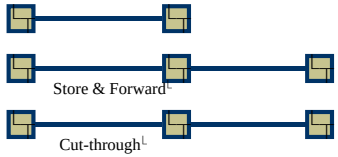
- Data center processing
  - Map-reduce, 3-tier business apps, load balancing (VM migration), …
  - What matters most?
- IP TV
  - Traditional cable app – mostly video!
  - What matters most?
- Wireless and mobile apps
  - For some users, primary way of accessing Internet
  - What matters most?

32

Lecture Overview

- Application requirements
- Application examples
  - ftp
  - http
- Internet Architecture & Performance intro

Packet Delay



Prop + xmit

2*(Prop + xmit)

2*prop + xmit

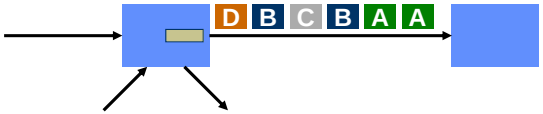
When does cut-through matter?

Next: Routers have finite speed (processing delay)

Routers may buffer packets (queueing delay)

Packet Delay Components

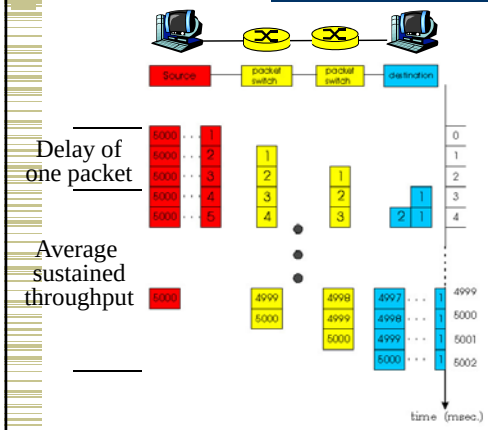
- Sum of a number of different delay components.
- Propagation delay on each link.
  - Proportional to the length of the link
- Transmission delay on each link.
  - Proportional to the packet size and 1/link speed
- Processing delay on each router.
  - Depends on the speed of the router
- Queuing delay 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
- Why? Historical: CS versus ECE.

Application-level Delay



Delay of one packet

Average sustained throughput

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

Units: seconds + bits/(bits/seconds)

* For minimum sized packet

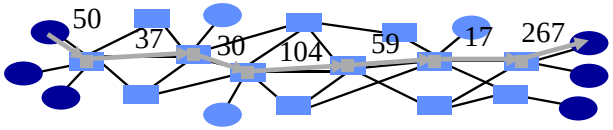
Some Examples

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

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

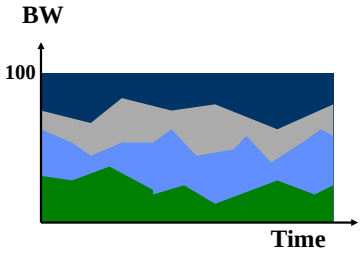
Sustained Throughput

- 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 matter why the link is slow!
  - Low link bandwidth
  - Many users sharing the link bandwidth



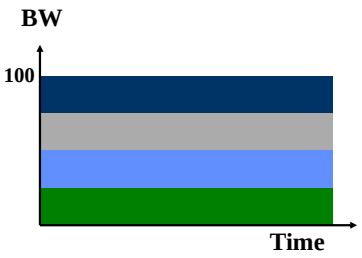
Bandwidth Sharing

- 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.

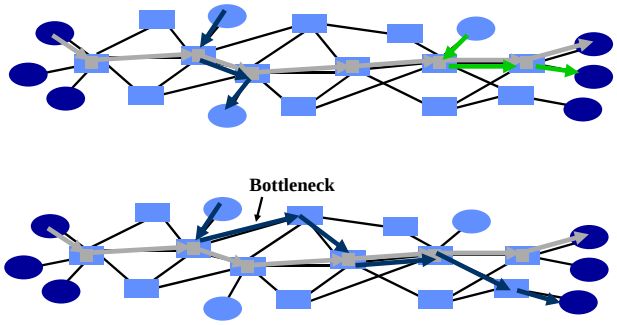


Fair Sharing of Bandwidth

- 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

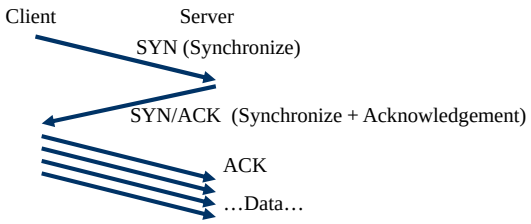


But It is Not that Simple



Must Consider Network Layer - Example: TCP

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



2: TCP transfers start slowly and then ramp up the bandwidth used – bandwidth is variable

Today

- Application layer
- Each application needs different services e.g., data loss? Elastic? Timing?
- FTP
- HTTP
- Today's hot apps: Facebook, amazon, skype, video, …
- Delay/Throughput