CS456 Notes

Module 1

Hosts: end systems

- Running *network apps*

Communication Links:

- Ex: fiber, copper, radio, satellite

Transmission rate: *bandwidth*

Packet switches: forward packets (chunks of data)

- *Routers* and *switches*

Internet: *"Network of networks"*

Protocols: control sending, receiving of messages

- TCP, IP, HTTP, Skype, 802.11

Network Edge: hosts (clients/servers)

Access networks/Physical Media: wired and wireless communication links

Network core:

- network of networks
- interconnected routers

Digital Subscriber Line (DSL):

- use existing telephone line to a central office
- data (internet) and voice over DSL

Access net: Cable Network:

- neighborhood connection

Frequency division multiplexing: different channels transmitted in different frequency bands

HFC: Hybrid fiber coax: asymmetric

Home Network:

- single access point (router + modem)
- many devices

Enterprise Access Networks (Ethernet)

- many access pointers or switches
- still one access link to internet

Wireless Access Networks

- wireless LANs: WiFi
- wide-area wireless networks: HSPA, LTE

Physical Media:
Bit: propagates between transmitter/receiver pairs

Physical link: what lies between transmitter & receiver

Guided Media: signals propagate in solid media – copper, fiber, coax

Unguided Media: signals propagate freely – radio

Packets: chunks of data of length L

Transmission Rate: link bandwidth

Transmission delay = time needed to transmit L-bit packet into link
= L/R

where R (bits/sec)

Store and Forward: entire packet must arrive at router before it can be transmitted onto the next link

End-to-end delay: 2L/R (zero propagation delay)

Queuing and Loss: if arrival rate (in bits) exceeds transmission rate for a long period of time.

- Packets will fill a queue and wait
- Once queue is full – packets will get lost

Routing: determines source-destination route taken by packets

Forwarding: move packets from router's input to appropriate routers output

Circuit Switching: end-to-end resources allocated (on hold)

- Dedicated resources: no sharing – guaranteed performance
- Telephone networks

Packet vs circuit switching

- Packet allows more users on the network
  - Bursty
  - Excessive congestion – packet loss
- Stable connection on circuit but can be simulated by packet
  - Guarantee skype get priority

Assess ISP: internet service providers

- They must be interconnected

Global ISPs → Regional Network → Local Net

- Content providers also in this – want to be closer to consumer

Packet delay

$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

Queuing delay:

Where a is the avg arrival rate

La/R ~ 0: delay will be small
La/R → 1: delay large
La/R > 1: more "work" arriving than can be serviced: infinite

`Traceroute`: provides delay measurement from source router along end-to-end internet path towards destination

Throughput: rate (bits/time) at which bits transferred between sender/receiver

- Avg vs instantaneous

Bottleneck link: link on end-end path that constrains end-end throughput

Protocols: define the *format, order* of message sent and received and actions taken on those messages

Layering: explicit structure allows identification, relationship of complex system's pieces

Internet Protocol Stack:

Application: supporting networking applications

- FTP, SMTP, HTTP

Transport: process-process data transfer

- TCP, UDP

Network: routing of datagrams from source to destination

- IP, routing protocols

Link: data transfer between neighboring network elements

- Ethernet, 802.11

Physical: actual bits

Denial of Service: (DoS): oversaturate resources of target such that it is unavailable to legitimate traffic

Packet Sniffing: read packets that are in the air (wireless)

IP Spoofing: send packet with false source address

Module 2

Client-Server Architecture

Server:

- Always-on
- Permanent IP address
- Data centers for scaling

Clients:

- Communicate with server
- May be intermittently connected
- May have dynamic IP
- Do not communicate directly with each other

P2P Architecture

- No always-on server
- arbitrary end systems directly communicate
- peer requests fulfilled by peers

Process: program running within a host

- different process may manage exchanging different messages

Client process initiates communication and *server* process waits to be contacted.

Sockets: analogous to door

- messages come in and out the door

To receive messages, processes must have an identifier

- IP address is not enough
- We need port numbers

Ex: HTTP: port 80, SMTP: port 25

Data integrity: how reliable is the data

Throughput: some apps require a lot of bandwidth

Timing: some apps require low delay (skype)

TCP
Reliable transport
Flow control – sender won't overwhelm receiver
Congestion control – throttle sender when network overloaded

Does not provide: timing, minimum throughput, security
Connection-oriented: setup required

UDP
Unreliable data transfer

Does not provide:

- Reliability
- Flow control
- Congestion control
- Timing
- Throughput guarantee
- Security
- Connection setup

HTTP: Hypertext transfer protocol:
Client: requests web objects
Server: sends web objects

Uses TCP:

- Connection initiated at port 80
- Stateless – no past info

Non-persistent HTTP

- At most one object sent over TCP
- Multiple objects = multiple connections

1. Client init connection
2. Waiting server accepts connection and lets client know
3. Client sends request message
4. Server receives request and sends message back about HTML file
5. Client see that there are 10 other web objects (imgs) referenced
6. Steps 1-4 repeated for each remaining object

2RTT + file transmission time

Persistent HTTP

- Multiple objects can be sent over single connection

- Leaves connection open after sending response
- Subsequent messages sent over open connection
- As little as one RTT/object

Two types of HTTP messages:

- Request
- Response

POST and GET methods

HTTP/1.0:

- GET
- POST
- HEAD

HTTP/1.1:

- GET,POST,HEAD
- PUT
  - Uploads file
- DELETE

Web Caches

- Act as both server and client

Conditional GET

- Only get if new version or if old version expires

FTP:

- Connect Port 21 – control connection
- Data over port 20 – transfer connection
  - Only open for the duration of the transfer

Electronic Mail:

- User agents
- Mail servers
- Simple mail transfer protocol: SMTP

User Agent: Mail client – outlook, thunderbird
Mail Server: contains mailbox, message queue (out), use SMTP to send mail

POP3 and IMAP for receiving mail
POP3:

  - Download-and-keep copies on different clients
  - Stateless across sessions

IMAP:

- Keeps all messages on server
- Keeps state
- Allows folder management

SMTP: uses TCP port 25 to reliably transfer messages

Domain Name System (DNS):

- Distributed database – hierarchy of many name servers
- Application-layer protocol – hosts communicate to resolve name translates

DNS Services

- Translation
- Aliasing
- Mail server aliasing
- Load distribution

Root DNS Servers → com/org/edu DNS servers → yahoo/google/amazon.com
13 root servers in world

Top-level domain (TLD) servers:

- Responsible for all top-level domain names com/org/edu/etc…

Authoritative DNS servers:

- Organization's own DNS servers(s)

Local DNS name server

- Each ISP or Institution has one
- May not belong to normal hierarchy

Iterated Query:

- Connected server replies with name of server to connect
- "I don't know but try this number"

Recursive Query:

- burden on contacted server
- heavy load at upper levels of hierarchy

Once name server learns a mapping – cache it

DNS records

- type=A
  - name – hostname
  - values – IP address
- type=NS
  - name - domain

- value – hostname of authoritative name server for this domain

- type=CNAME
  - name – alias (www.ibm.com → servereast2.ibm.com)
  - value – canonical name
- type=MX
  - value – name of mailserver associated with name

Pure P2P architecture

- no always-on server
- peers intermittently connected
- peers connect directly and service each other

Client-Server Transmission: must upload N times
P2P Transmission: peers upload and download file – unchoke each other

Distributed Hash Table:

- (key, value) pairs

Module 3

Transport protocols run in end systems

- send messages in segments to network layer
- reassemble on app layer

network layer: between hosts
transport layer: between processes

Transmission Control Protocol TCP

- congestion control
- flow control
- connection setup

User Datagram Protocol UDP

- unordered, unreliable
- datagram contains IP and port of src and dest
- connectionless

RDT 1.0: reliable transfer over reliable channel

- assuming: no packet loss, no bit errors

RDT 2.0: with bit errors

- checksum to detect bit errors
- ACKs and NAKs
- Flaw: ACKs and NAKs could get corrupted
- Handling duplicates:

  - Sender adds a sequence number to each packet (receiver discards duplicates)

Stop and wait
- Sender sends then waits for response

RDT 2.1: handle garbled ACK/NAKs
- Sequence number [0,1]

RDT 2.2: NAK-free protocol
- Use ACKs only
- Instead of NAK – send last OK ACK: include seq#

RDT 3.0: handle bit errors and loss
- Sender waits for ACK
- Resend if no ACK received
- If duplicate sent seq# handles this

Works! But performance is really bad

Pipelined Protocols
- Pipelining: sender allows multiple “in-flight” yet-to-be-acknowledged packets
  - Seq# must be increases
  - Buffering at sender and/or receiver

Go-Back-N
- No buffer if out of order resend anyways

Selective repeat
- Needs buffer