Chapter 1 - Introduction

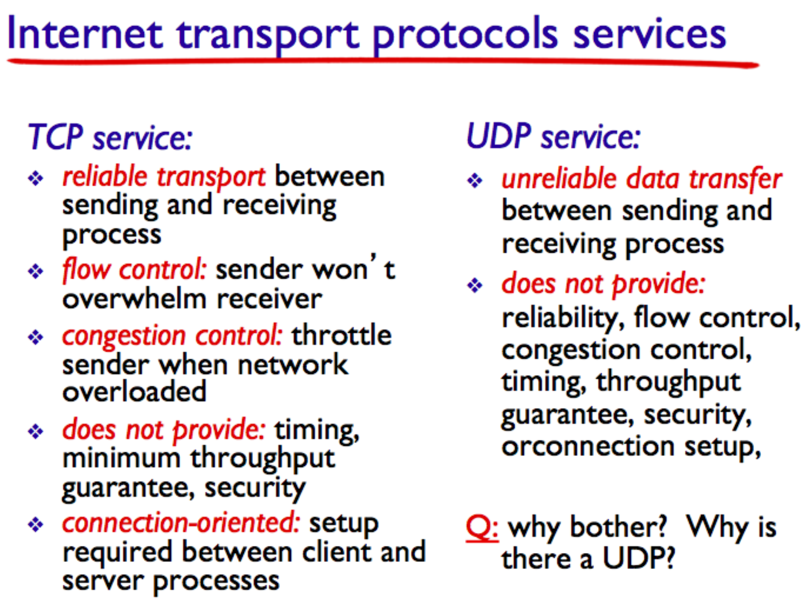
1. Internet: a "network of networks"; infrastructure that provides service to applications; provides programmable interfaces to applications.
2. Network edge: “hosts”; clients & servers; end-systems.
3. Network core: “interconnected routers”; routers; network of networks.
4. Protocol Stack: internet protocol stack (see layered model)
5. Packet switching: hosts break application-layer messages into packets. Forward packets from one router to another from source to destination. Takes L/R seconds to transmit L-bits packet into link at R bps.
6. Circuit switching: end-to-end resources allocated to, reserved for “call” between source and destination. “Reserved line/connection"
7. Internet Service Providers (ISPs): residential, commercial, and university. ISPs must be connected for communication. “Networks of networks"
8. Backbones: central conduit designed to transfer network traffic at high rates. Used to maximize the reliability and performance of large-scale networks.
9. Forms of:
    a. Delay:
        i. nodal processing: check bit errors, determine output link, normally < msec
        ii. queuing: time waiting at output link for transmission.
        iii. transmission: L, packet length (bits) & R, bandwidth (bps) = L/R
        iv. propagation: d, length of physical link & s, propagation speed = d/s
    b. Loss: queue preceding link in buffer has finite capacity, packets arriving with full queue will cause packet loss.
    c. Throughput: rate at which bits are transferred between sender and receiver.
10. Layered Model:
    a. Application: supporting network applications (FTP, SMTP, HTTP, etc.)

   b. Transport: process-process data transfer (TCP, UDP)
   c. Network: routing of datagrams from source to destination (IP, routing protocols)
   d. Link: data transfer between neighboring network elements (Ethernet, WiFi)
   e. Physical: bits “on the wire"

11. <u>Messages:</u> application layer
12. <u>Segments:</u> transport layer
13. <u>Datagrams:</u> network layer
14. <u>Frames:</u> link layer
15. <u>Network Security Issues:</u>
    a. malware: infects hosts.
    b. denial of service (DoS): make resources unavailable to legitimate traffic by overwhelming resource with bogus traffic.
    c. packet sniffing: promiscuous network reads/records all packets passing through network.
    d. IP spoofing: send packet with false source address.
16. <u>Internet History:</u> BLAH!

<u>Chapter 2 - Application Layer</u>

1. <u>Client-Server:</u>
   a. Client: communicates with server, dynamic or static IP, do not communicate directly with each other.
   b. Server: always-on, permanent IP, data centers for scaling.
2. <u>Peer to Peer:</u> no servers, end-systems communicate directly with each other, peers request services with other peers, self-scalability, peers are intermittently connected and change IP addresses.
3. <u>Services from Transport layer:</u>
   a. Data integrity: some apps require RDT
   b. Timing: some apps require low delay to be effective
   c. Throughput: some require minimum throughput to be effective, others make use of what they can get (elastic applications)
   d. Security: encryption, data integrity,etc.



Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

UDP service:

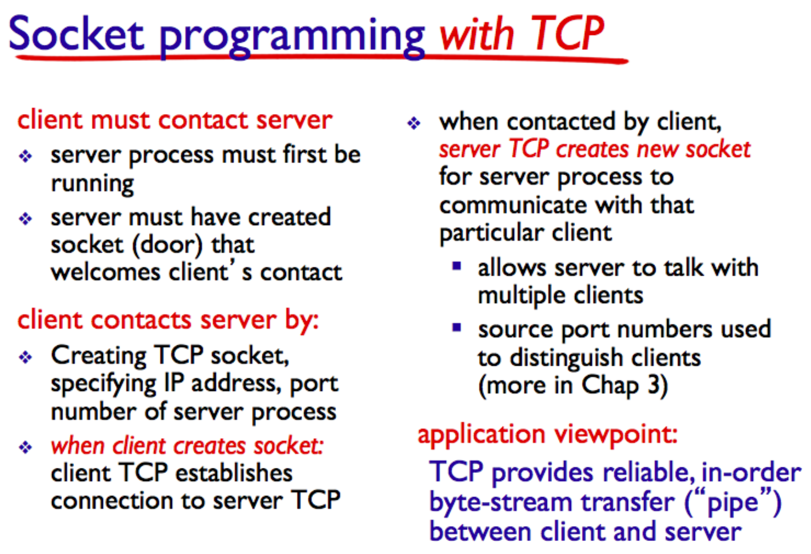
- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, orconnection setup,

Q: why bother? Why is there a UDP?

4.
5. HTTP Features:
    a. persistent: multiple objects sent over single TCP connection.
    b. non-persistent: at most 1 object sent over TCP connection then connection is closed.
    c. Two types of HTTP messages, *request & response*
    d. Cookies
    e. Proxy-server:
6. File Transfer Protocol (FTP): transfer file to/from remote host
    a. FTP client contacts FTP server at port 21 using TCP. When server receives file transfer command it opens another TCP data connection to client. After transferring one file server closes connection and reopens if it needs to send more files.
    b. FTP server maintains "state" which includes current directory and earlier authentication.
7. Simple Mail Transfer Protocol (SMTP): uses TCP
    a. User agents: "mail reader"
    b. Mail servers: mailbox, message queue, and SMTP protocol
    c. SMTP protocol

   d. Three phases of transfer
      i. handshaking
      ii. transfer of messages
      iii. closing
   e. Uses persistent connections

8. <u>Domain Naming Service (DNS):</u>
   a. Advantage of using UDP: connectionless therefore faster.
   b. Root name servers: contacted by local name servers who cannot find name.
      i. contacts authoritative name server if mapping not known, gets mapping, returns it.
   c. Top Level Domain servers: responsible for top level “country” domains
   d. Authoritative server: organizations own DNS server.
   e. Local server: called when host makes DNS query, also known as default name server.
   f. Iterated query: contacted server returns with name of server to contact next.
   g. Recursive query: contacted server contacts next level server until it finds name.
   h. Caching: once name server learns mapping it caches it. Can become out of date.
9. <u>Sockets:</u> door between application process and end-to-end-transport protocol.
   a. <u>UDP:</u> unreliable datagram, sender explicitly attaches IP destination address and port # to packet and receiver extracts address and port # from packet.
   b. <u>TCP:</u> reliable datagram.



Socket programming *with TCP*

client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client' s contact

client contacts server by:

- Creating TCP socket, specifying IP address, port number of server process
- *when client creates socket:* client TCP establishes connection to server TCP

- when contacted by client, *server TCP creates new socket* for server process to communicate with that particular client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients (more in Chap 3)

application viewpoint:

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server

Chapter 3 - Transport Layer

1. Logical Communication:
   a. transport protocols run in end-systems.
   b. send side: breaks up message(s) into segments then sends to network layer.
   c. rcv side: reassembles segments into messages then passes to application layer.
2. Relationship between Transport & Network layers:
   a. transport layer: logical communication between processes.
   b. network layer: logical communication between hosts (i.e. postal service)
3. Multiplexing: (sender) handles data from multiple sockets, add transport header used by demultiplexer.
   a. TCP: Works similar as demultiplexing. Uses port numbers as a solution for identifying services.
   b. UDP: Also uses port numbers to identify services.
4. Demultiplexing: (receiver) uses info from header to deliver received segments to correct socket.
   a. TCP: (4-tuple) requires source/destination IP address & port #.

Receiver uses all 4 values to direct segment to appropriate segment.

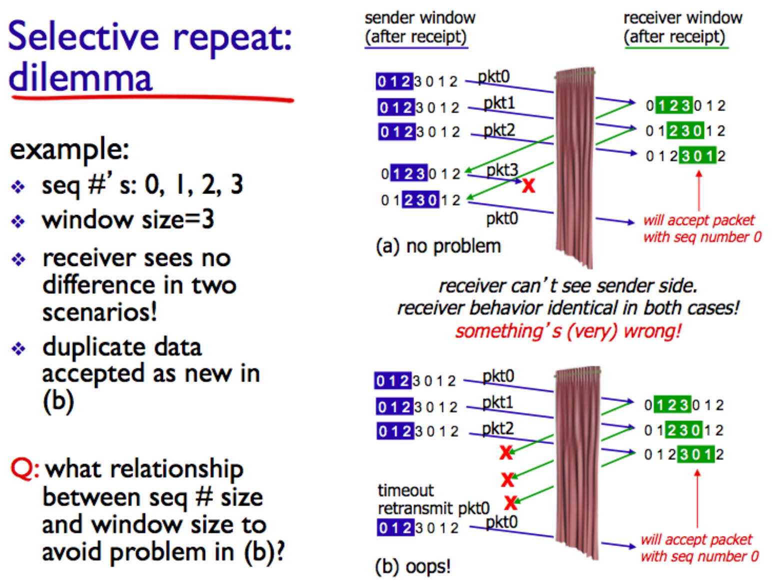
b. UDP: requires destination IP address & port # in datagram header.

5. <u>UDP services:</u>
    a. “best effort”
    b. segments may be lost or out of order.
    c. no handshaking and handled independently.
    d. UDP use:
        i. streaming multimedia apps
        ii. DNS
        iii. SNMP
    e. reliable transfer over UDP
        i. reliability added at application layer.
    f. <u>Checksum:</u> detect “errors” in transmitted data (i.e. flipped bits).
        i. Sender: addition of segment contents.
        ii. Receiver: check if computed checksum equals checksum field value.
6. <u>Reliable Data Transfer versions 1,2,3 (RDT):</u>
    a. important in Application, Transport, and Link layers.
    b. <u>RDT 1.0:</u> reliable data transfer over reliable channel.
        i. no bit errors
        ii. no loss of packets
        iii. separate FSMs for sender/receiver, only 1 for sender & 1 for receiver.
    c. <u>RDT 2.0:</u> channel with bit errors
        i. underlying channel may flip bits in packet
        ii. checksum used to detect bit errors.
        iii. How to resolve from bit errors?
        iv. ACK: sender sends ACK to receiver stating it received the packet w/o issues.
        v. NAK: sender explicitly tells receiver the packet had errors.
        vi. sender then retransmits packet after receiving NAK.
        vii. New mechanism in 2.0 (ERROR DETECTION, using ACK & NAK).

        viii. Fatal flaw: what if ACK/NAK are corrupt? - sender doesn't know what happened at receiver.
        ix. Solution: sender adds sequence # to each packet and receiver discards duplicate packets (duplicate sequence #'s).
        x. Leads to stop & wait protocol.
    d. RDT 2.1: Sender/receiver handle garbled ACK/NAK
    e. RDT 2.2: NAK-free protocols
    f. RDT 3.0: channels with errors & loss
7. Stop & Wait: send a packet and wait for an acknowledgement.
    a. Problem: very small bandwidth therefore very small throughput.
    b. Solution: Windows, on the fly packet transferring without acknowledgement.
8. Efficiency:

Commented [1]: Did not find anything on efficiency in powerpoint. Could be efficiency of RDT stop & wait or something but I'm not sure.

9. Pipelining: sender allows multiple, "in-flight," "yet-to-be-acknowledged" packets.
    a. Go-Back-N:
        i. sender can have up to N unacknowledged packets in the pipeline.
        ii. receiver only sends cumulative acknowledged packet
            1. doesn't acknowledge packet if there are any gaps in packet.
        iii. sender has a timer for oldest unacknowledged packet, the sends all unacknowledged packets once times out.
    b. Selective Repeat:
        i. sender can have up to N unacknowledged packets in the pipeline.
        ii. receiver sends acknowledgement after each packet is received.
        iii. sender maintains timer for each packet and only retransmits the packet that times out.
        iv. **Dilemma: duplicate data can be accepted as new input.



Selective repeat: dilemma

example:

- seq #'s: 0, 1, 2, 3
- window size=3
- receiver sees no difference in two scenarios!
- duplicate data accepted as new in (b)

Q: what relationship between seq # size and window size to avoid problem in (b)?

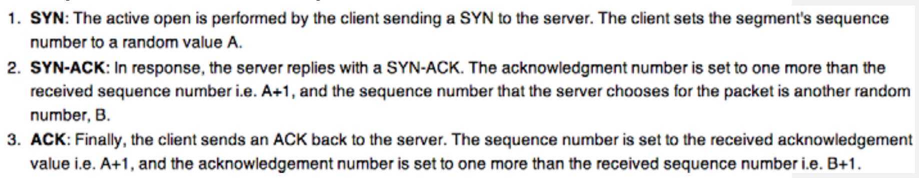
sender window (after receipt)
receiver window (after receipt)
pkt0
pkt1
pkt2
pkt3
pkt0
will accept packet with seq number 0
(a) no problem
receiver can't see sender side.
receiver behavior identical in both cases!
something's (very) wrong!
timeout retransmit pkt0
(b) oops!

10. TCP Reliable Data Transport (RDT): creates RDT service on top of IPs unreliable service via:
    a. Pipelining
    b. Cumulative acknowledgments
    c. Single retransmission timer
11. Round Trip Time (calculation):
    a. *EstimatedRTT = (1 - alpha) * EstimatedRTT + alpha * SampleRTT*
        i. Typically *alpha* = 0.125.
12. Timeout (calculation):
    a. *TimeoutInterval = EstimatedRTT + 4 * DevRTT*
    b. *DevRTT = (1 - beta)* DevRTT + beta * (SampleRTT - EstimatedRTT)*
        i. Typically *beta* = .25
13. Fast Retransmit on 3 repeat ACK: if sender receives 3 acknowledgements from same data then resend unacknowledged segment with smallest sequence #
14. Flow Control: receiver controls sender so sender won't overflow

receivers buffer by sending too much too fast.

   a. receiver advertises “free buffer space” using the *“rwnd”* value in TCP-header
   b. sender limits amount of unacknowledged “in-flight” segments to receivers *rwnd* value.

15. <u>TCP Connection Setup:</u> “handshake”
   a. 2-way handshake (issues): client requests connection & server accepts connection.
      i. What about timeouts? Established connection can get crossed, i.e. open server & closed client.
   b. 3-way handshake: used by TCP



1. **SYN**: The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.
2. **SYN-ACK**: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.
3. **ACK**: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1.

16.

17. <u>Congestion Control (TCP):</u> too many sources sending too much data too quickly for network to handle.
   a. sender increases transmission rate (window size), probing for usable bandwidth until loss occurs.
   b. additive increase: increase *cwnd* by 1 mss for every RTT until loss detected.
   c. multiplicative decrease: cut *cwnd* in half after loss.
   d. *rate = cwnd/RTT* bytes/second
   e. TCP slow start: starts of at 1mss rate but doubles until encountering a loss therefor speeds up exponentially fast.
   f. Loss indicated by timeout:
   g. *cwnd* cut to 1 mss, grows exponentially like “slow start” to threshold, then increments linearly.
   h. Loss indicated by three duplicate acknowledgements.
   i. *cwnd* cut in half then grows linearly.
   j. *Avg. TCP Throughput = 3/4*(W/RTT) bytes/second* where *W is the window size.*

Quiz 1

1. What is at the core of the internet?
    a. Routers
2. What is found at the edges of the internet?
    a. End-systems
3. Which parts of the protocol stack are found at the edges and core of the internet?
    a. Edge: Application, transport, and core layers (below).
    b. Core: Physical, link, and network layers.
    c. Session and presentation layers aren't really used in IP stack.
4. How does throughput relate to delay, loss, and bandwidth?
    a. Increase delay causes decrease in throughput.
    b. Increase loss causes decrease in throughput.
    c. Bandwidth increases and so does throughput.
5. Who started the internet (besides Al Gore)?
    a. ARPANET & NSFNET funded primarily by US government.
6. What is the major architectural difference between http and BitTorrent?
    a. HTTP is client/server.
    b. BitTorrent is peer to peer.
7. What is DNS and what protocol is it built over?
    a. Domain Name System used to translate hostnames to IP, mostly built over UDP.
8. What protocol is smtp built over?
    a. TCP
9. What is the major difference between a TCP socket and a UDP socket?
    a. TCP is connection oriented whereas UDP is connectionless (send and forget).
10. How many standard listening port numbers are assigned to FTP? What are they?
    a. Two, 20 & 21.