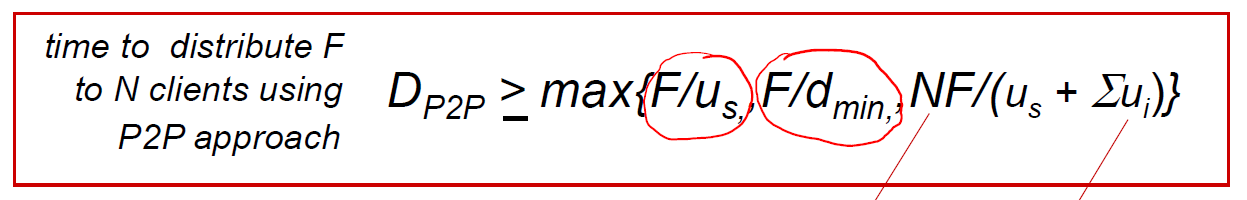
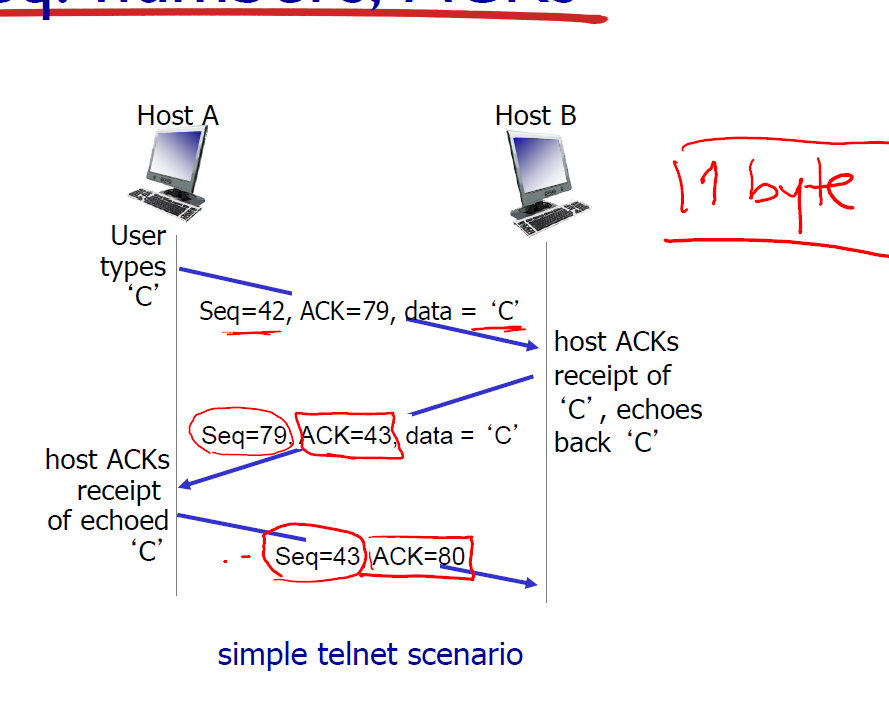
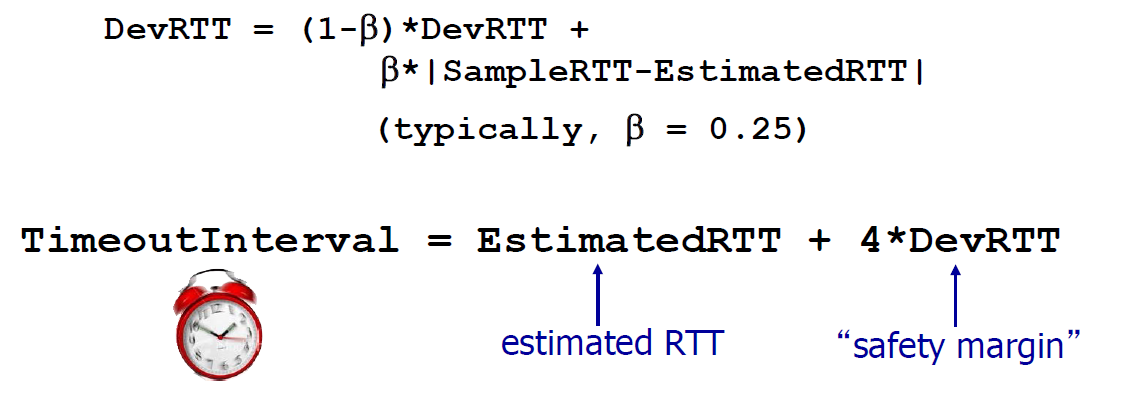
**Introduction:**

* What are computer networks:
  + Set of end systems that use protocols and internet standards to communicate with one another
* Distributed applications
  + Infrastructure that provides services to applications
  + Provides programming interface to apps
* Protocols
  + Schematics: set a standard so users can understand each other when communicating
  + Actions, format and order of messages sent and received
  + Actions taken on message transmission and receipt
* Types of communication networks
  + Circuit switching: alternative core
    - Telephone network
    - Serving resources over the network once connection established
    - Divide network: FDM, TDM, CDM\*
    - No sharing, guaranteed performance
    - Circuit segment idle if not used by call
  + Message switching:
    - Slower than packet switching
    - Store-and-forward: entire packet must arrive at router before it can be transmitted on next link: NL/R
  + Packet switching
    - Uniform traffic patterns
    - Resources requested on demand (statistical multiplexing, queueing)
    - Queueing: wait to be transmitted on link
    - Packets can be dropped (lost) if memory buffer fills up
  + Packet vs Circuit:
    - packet allows more users, but if too many, speeds can be low
    - packet is great for bursty data
    - packet may have excessive congestion: packet delay and loss
    - make circuit-like behavior: unsolved
* Internet architecture (edge and core)
  + Access networks and physical media (copper, fiber, coaxial, wireless, cables)
  + Tier-1: commercial ISPs (AT&T, NTT)
  + Content provider network bypass T1, regional ISPs (google, Microsoft)
  + POP: Point of Presence
  + Coaxial vs fiber: fiber has low error rate, immune to electromagnetic noise
  + Network of networks (interconnected ISPs, )
* Types of delay in packet switched networks (TCP/IP)
  + Transmission delay: size of packet/transmission rate (L/R)
  + Propagation delay: length of the link/propagation speed (M/P)
  + Queueing delay: packets arrive at buffer, time waiting to be transmitted, average queueing delay
  + Processing delay: time to process message/packet with overhead
* Protocol Layers:
  + TCP/IP: 5 layers
  + OSI reference model: 7-layer model, simplification of upper layers
    - Presentation after Application (encryption, compression, etc)
    - Session after Presentation (sync, checkpoints, recovery)
    - Internet is missing these separated layers (must be implemented)
  + Encapsulation: same layer protocol communication
  + Each layer implements a service
  + Application: supporting network applications
    - FTP, SMTP, HTTP, DNS
  + Transport: process-process data transfer
    - TCP (OC), UDP (NOC)
  + Network: routing of datagrams from source to destination
    - IP, routing protocols
  + Link: data transfer between neighboring network elements
    - Ethernet, Wifi, PPP
  + Physical: bits “on the wire”
  + Security:
    - DDos: attackers overwhelm system with bogus traffic
    - Packet sniffing: steals packets from source on its way to destination
    - IP spoofing: send packet with false source address

**Application Layer:**

* Application layer protocols: client-server, sending-receiving process
* Architecture:
  + Client/server:
    - Server: always on-host, permanent IP, data centers for scaling
    - Clients: communicate with server, dynamic IP, no direct client communication
  + Peer-to-peer: find other peers and exchange information, trial and error, switching from one neighbor to another by finding the best one
* Transport Protocol Services:
  + TCP: reliable transport, flow/congestion control, does not provide, connection-oriented
  + UDP: unreliable data transfer, does not provide, better for streaming multimedia
* Service available to an Internet application
  + Reliable, congestion-controlled data transfer (TCP) (more secure but slower)
  + Unreliable data transfer (UDP) more control at application layer (faster)
* Web and HTTP
  + Non-persistent: one session per object
  + Persistent: one session is open for all new objects
    - Get all webpages for the same time
* Caching
  + Save common info in a local storage, where I can access faster than going all the way to the internet and root servers, which is far
    - Reduce response time (increasing access link speed is expensive)
    - Reduce traffic on an institution’s access link
  + Conditional get: don’t send object if cache has up-to-date cached verion
    - No object transmission delay
    - Lower link utilization
  + DNS
  + Cookies usage: authorization, shopping carts, recommendations, user session state
* Electronic mail: user agents, mail servers, simple mail transfer protocol (SMTP)
  + User agents: mail reader (outlook, gmail)
  + Messages stored on server
  + SMTP uses TCP to reliably transfer email messages from client to server
    - Handshaking, transfer of messages, closure
    - ASCII text, response is status code and phrase
    - Persistent connections
    - HTTP push, SMTP pull
  + RFC 5322 has to-from-subject
  + SMTP has FROM, RCPT TO:
  + Main Access protocols (IMAP, POP3, HTTP)
    - POP3:
      * Authorization: user, pass, +OK, -ERR
      * Transaction: list, retr, dele, quit
    - IMAP : keeps all messages in one place (server)
      * Allows user to organize messages in folders
      * Keeps under state across sessions (name of folders, mapping, etc)
    - HTTP: user agent is web browser
      * Emails sent from receipt mail server to recipient’s browser
      * Emails message are sent from sender user agent to recipient’s mail server
      * Exchange between mail servers (still SMTP)
* DNS: domain name system
  + Types of records, hierarchy
  + Distributed database
  + Application-layer protocol
  + Runs on top of UDP
  + Host aliasing, mail server aliasing, hostname to IP address translation, load distribution
  + Centralize? Single point of failure, traffic volume, distance, maintenance, no scaling
  + Iterative: I don’t know this name, but ask this server
  + Recursive: puts burden on name resolution on contacted name server (heavy load up)
  + RR (Type A (hostname IP), Type NS (domain), Type CNAME (alias), Type MX (mailserver))
  + DDos not successful, redirect attacks (intercept) exploit DNs (spoofed IPs)
* P2P applications
  + No always-on server, arbitrary end systems, peers are connected
  + BitTorrent, KanKaM, Skype
  + Client-server vs p2p on small and large client pools
  + 
  + Tracker: tracks peers participating in torrent
  + Torrent: group of peers exchanging chunks of a file
  + Tit-for-tat: send chunks to peers with highest rate (optimistically unchoke)
* Video streaming and content distribution network
  + CBR: constant bit rate
  + VBR: variable bit rate
  + DASH (Dynamic Adaptive Streaming over HTTP)
    - Server: divide video file into chunks, chunks encoded at different rates, manifest
    - Client: periodically measure s-to-c bandwidth, max coding rate from bandwidth
    - Smart: when, what and where
  + CDN: Netflix, Youtube
    - Large megaserver? (no scale)
    - Store multiple copies geographically distributed?
    - OTT: adapt to congestion
    - Youtube: private CND, pull-caching, DNS redirect, manual resolution selection
* Socket Programming in TCP, UDP:
  + Client/server applications communicate using sockets
  + Socket : door between application process and E2E protocol
  + UDP: unreliable, no connection between C/S, data may be lost
  + TCP: reliable, handshaking

**Transport Layer:**

* Network layer: logical communication between hosts
* Transport layer: logical communication between processes
* TCP Reliable: congestion, flow control, connection setup
* UDP Unreliable: no-frills, delay guarantees bandwidth guarantees
* Multiplexing and demultiplexing
  + Connectionless and connection-oriented (socket programming)
  + Multiplexing: handle data from multiple sockets (add transport header)
  + Demultiplexing: use header info to deliver received segments to correct socket
    - TCP: Source-destination IPs and Ports (4), one datagram carries one segment
    - UDP: destination IP and port, same socket as destination
* UDP
* Reliable data transfer
  + Automated repeat request (ARQ) protocols (retransmitting packets)
  + Stop-and-wait
  + Pipelined RDT (GBN)
* TCP
  + Connection management (3-way handshake)
  + 
  + 
  + 
  + Retransmission: Lost ACK scenario, premature timeout, cumulative ACK
  + RDT (hybrid)
  + Flow control (other side reduces)
    - Receiver controls sender, so sender won’t overflow, too much too fast
  + Congestion control (sender reduces)
    - Too many sources sending too much data too fast for network to handle
    - Lost packets, long delays
    - Two senders, two receivers: large delays at arrival rate, near capacity
    - One router, finite buffers: retransmit data + original data
    - Four senders, multihop paths, timeout/retransmit:
      * Wasted upstream transmission when packet dropped
  + TCP Congestion control:
    - Sender increases transmission rate
      * Additive increase: increase cdnd by 1 MSS evert RTT until loss detected
      * Multiplicative decrease: cut cwnd in half after loss