CS3357a Notes

Protocols – Define format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

Network edge:

* Hosts: clients and servers
* Connect to end systems through
  + Residential access nets
  + Institutional access networks (school, company)
  + Mobile access networks

Access networks, physical media: wired, wireless communication links

Network core:

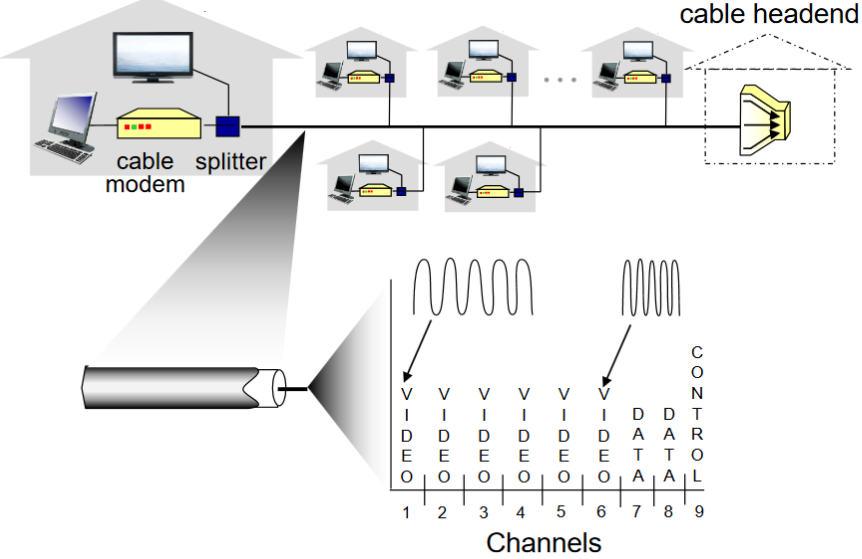
* Interconnected routers
* Network of networks

Digital subscriber line (DSL)

* Use existing telephone line to central office DSLAM (24/2.5 Mbps)
  + Data over DSL phone line goes to Internet
  + Voice over DSL phone line goes to telephone net

Cable network

* Frequency division multiplexing: Different channels transmitted in different frequency bands



* HFC: hybrid fiber coax
  + Asymmetric: up to 30/2 Mbps transmission rate
* Network of cable, fiber attaches homes to ISP router
  + Homes share access network to cable headend unlike DSL, which has dedicated access to central office

Enterprise access networks (Ethernet)

* Typically used in companies, universities, etc.
* 10 Mbps, 100 Mbps, 1 Gbps, 10 Gbps transmission rates
* Today, end systems typically connect into Ethernet switch

Wireless access networks

* Shared wireless access network connects end system to router via base station aka access point
  + Wireless LANs:
    - Within building (100ft)
    - 802.11b/g/n (WiFi): 11, 54, 450 Mbps transmission rate
  + Wire-area wireless access
    - Cellular, 10’s km
    - Between 1 and 10 Mbps
    - 3G, 4G:LTE

Host: sends packets of data

* Takes application message and breaks into smaller chunks, known as packets, of length L bits
* Transmits packet into access network at transmission rate R
  + Link transmission rate, aka link capacity, aka link bandwidth

Packet transmission delay = Time needed to transmit L-bit packet into link = L(bits) / R(bits/sec)

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, eg. Radio
* Twisted pair (TP): Two insulated copper wires (CAT5: 100 Mbps, 1 Gbps Ethernet, CAT6: 10 Gbps)
* Coaxial cable:
  + Two concentric copper conductors
  + Bidirectional
  + Broadband:
    - Multiple channels on cable
    - HFC
* Fiber optic cable
  + Glass fiber carrying light pulses, each pulse a bit
  + High-speed point-to-point transmission (eg. 10’s – 100’s Gbps transmission rate)
  + Low error rate:
    - Repeaters spaced far apart
    - Immune to electromagnetic noise
* Radio
  + Signal carried in electromagnetic spectrum
  + No physical wire
  + Bidirectional
  + Propagation environment effects:
    - Reflection
    - Obstruction by objects
    - Interference
  + Radio link types:
    - Terrestrial microwave
      * Up to 45 Mbps channels
    - LAN (eg WiFi)
      * 54 Mbps
    - Wide-area (eg cellular)
      * 4G cellular: ~10 Mbps
    - Satellite
      * Kbps to 45 Mbps channel (or multiple smaller channels)
      * 270 msec end-end delay
      * Geosynchronous versus low altitude

Network core

* Mesh of interconnected routers
* Packet-switching: hosts break application-layer messages into packets

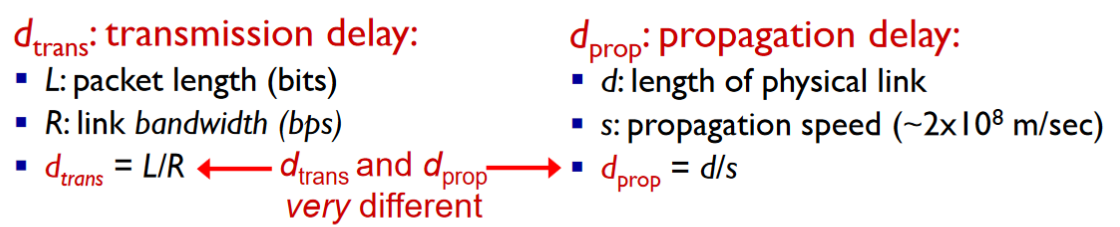
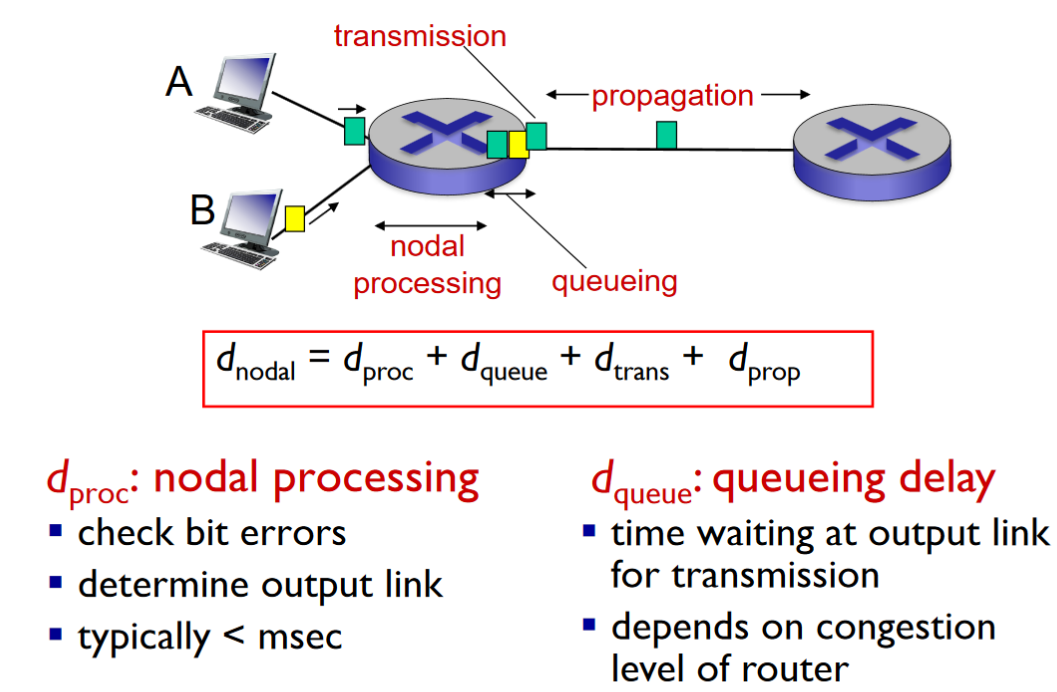
End-end delay = 2L/R (assuming zero propagation delay)

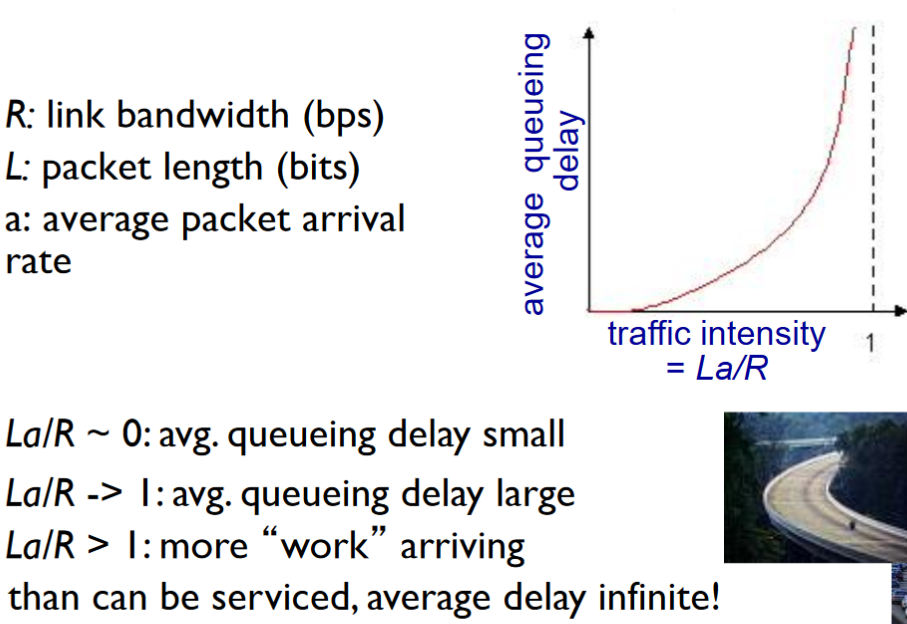
Routing: determines source-destination route taken by packets through algorithms

Forwarding: move packets from router’s input to appropriate router output

Packet switching allows more users to use network than circuit swiching

* Great for bursts of data, simpler
* Excessive congestion possible





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

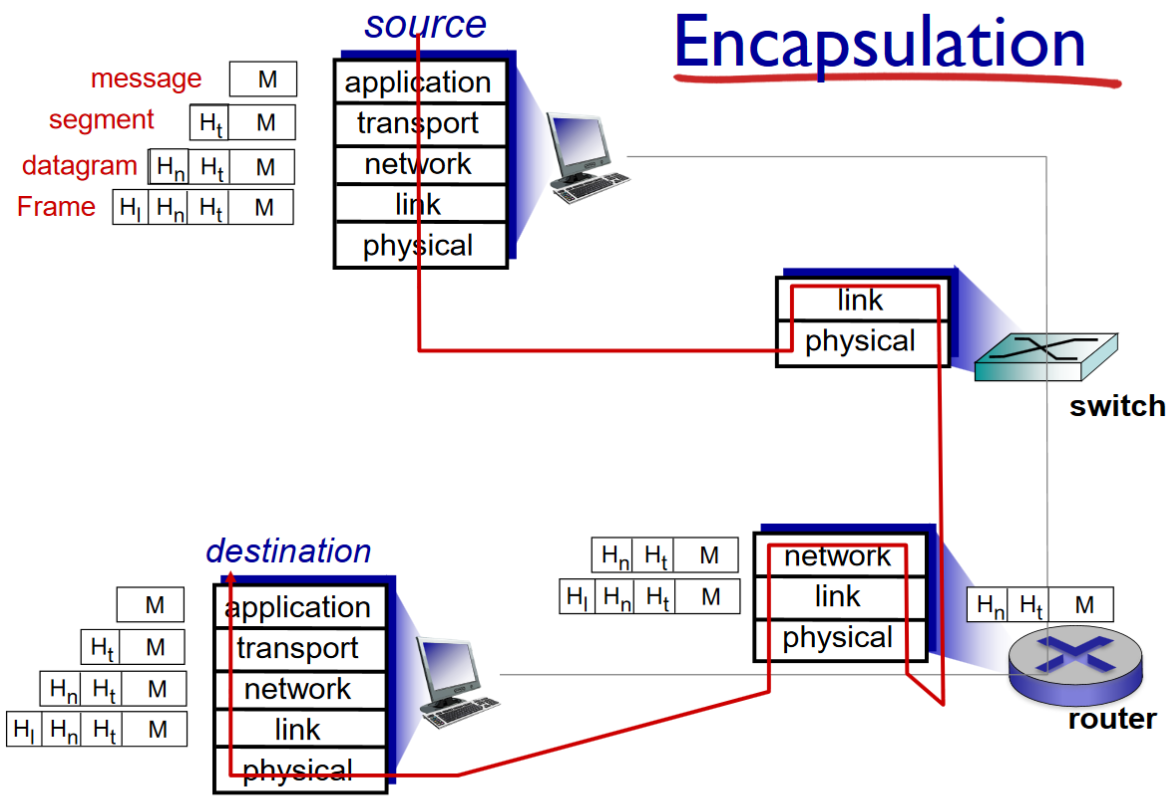
* Instantaneous: rate at given point in time
* Average: rate over longer period of time

Layers: Each layer implements a service via its own internal-layer actions relying on services provided by layer below

* Helps deal with complex systems
* Modularization eases maintenance, updating of system

Internet protocol stack

* Application: supporting network 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 (WiFi), PPP
* Physical: bits “on the wire”
* Presentation: allow applications to interpret meaning of data (eg encryption, compression, machine-specific conventions)
* Session: synchronization, checkpointing, recovery of data exchange



Cerf and Kahn’s internetworking principles:

* Minimalism, autonomy – no internal changes required to interconnect networks
* Best effort service model
* Stateless routers
* Decentralized control

Define today’s Internet architecture

Chapter 2

Client-server architecture:

Server:

* always-on host
* permanent IP/DNS address
* data centers for scaling

Clients:

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

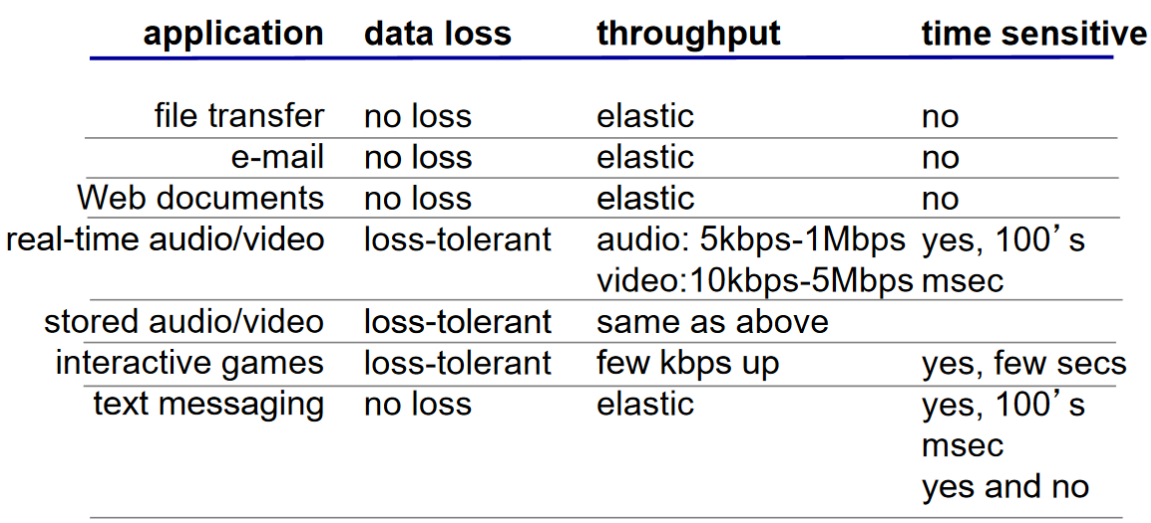
P2P architecture

* No always-on server
* Arbitrary end systems directly communicate
* Peers request service from other peers, provide service in return to other peers
  + Self scalability – new peers bring service capacity, as well as new service demands
* Peers are intermittently connected and change IP addresses
  + Complex management

Process: program running within a host

Sockets: analogous to door; where processes send/receive messages

To receive messages, process must have identifier that includes both IP address and port numbers associated with process on host.



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, or connection setup

Securing TCP

* TCP & UDP
  + No encryption
  + Cleartext passwords sent into socket traverse Internet in cleartext
* SSL
  + Provides encrypted TCP connection
  + Data integrity
  + End-point authentication
  + At the app layer: use SSL libraries that “talk” to TCP
  + SSL socket API: cleartext passwords sent into socket traverse Internet encrypted

Non-persistent HTTP: At most one object sent over TCP connection, downloading multiple objects requires multiple connections

Persistent HTTP: multiple objects can be sent over single TCP connection between client and server

RTT: time for a small packet to travel from client to server and back

HTTP response time:

* One RTT to initiate TCP connection
* One RTT for HTTP request and first few bytes of HTTP response to return
* File transmission time
* Non-persistent HTTP response time = 2RTT + file transmission time

Non-persistent HTTP issues:

* Requires 2 RTTs per object
* OS overhead for each TCP connection
* Browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:

* Server leaves connection open after sending response
* Subsequent HTTP messages between same client/server sent over open connection
* Client sends requests as soon as it encounters a referenced object
* As little as one RTT for all the referenced objects

Cookies: can be used for

* Authorization
* Shopping carts
* Recommendations
* User session state (Web email)

Electronic mail components:

* User agents
  + “mail reader” that composes, edits, and reads mail messages
  + Outgoing, incoming messages stored on server
* Mail servers
  + Mailbox contains incoming messages for user
  + Message queue of outgoing mail messages
  + SMTP protocol between mail servers to send email messages
    - Client: sending mail server
    - “server”: receiving mail server
* Simple mail transfer protocol: SMTP
  + Uses TCP to reliably transfer email message from client to server, port 25
  + Direct transfer: sending server to receiving server
  + Three phases of transfer
    - Handshaking (greeting)
    - Transfer of messages
    - Closure
  + Command / response interaction
    - Commands: ASCII text
    - Response: status code and phrase
    - Messages must be in 7-bit ASCII

Mail access protocol: retrieval from server

* POP: Post Office Protocol [RFC 1939]: authorization, download
* IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server
* HTTP: gmail, Hotmail, Yahoo! Mail, etc.

DNS: hostname to IP address translation

Root name servers: contacted by local name servers that cannot resolve name

Top-level domain (TLD) servers:

* Responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains (eg uk, fr, ca, jp)
* Network Solutions maintains servers for .com TLD
* Educause for .edu TLD

Authoritative DNS servers:

* Organization’s own DNS server(s), providing authoritative hostname to IP mappings for organization’s named hosts
* Can be maintained by organization or service provider

Local DNS name server

* Does not strictly belong to hierarchy
* Each ISP (residential ISP, company, university) has one
* When host makes DNS query, query is sent to its local DNS server
  + Has local cache of recent name-to-address translation pairs (but may be out of date)
  + Acts as proxy, forwards query into hierarchy

DNS records: Distributed database storing resource records (RR)

* RR format: (name, value, type, ttl)
* Type = A
  + Name is hostname
  + Value is IP address
* Type = NS
  + Name is domain (eg foo.com)
  + Value is hostname of authoritative name server for this domain
* Type = CNAME
  + Name is alias name for some “canonical” (the real) name
  + [www.ibm.com](http://www.ibm.com) is really servereast.backup2.ibm.com
  + Value is canonical name
* Type = MX
  + Value is name of mailserver associated with name

DASH: Dynamic, Adaptive Streaming over HTTP

* Server:
  + Divides video file into multiple chunks
  + Each chunk stored, encoded at different rates
  + Mainifest file: provides URLs for different chunks
* Client:
  + Periodically measures server-to-client bandwidth
  + Consulting manifest, requests one chunk at a time
    - Chooses maximum coding rate sustainable given current bandwidth
    - Can choose different coding rates at different points in time(depending on available bandwidth at the time)
  + Intelligence at client: decides
    - When to request chunk (so that buffer starvation, or overflow does not occur)
    - What encoding rate to request (higher quality when more bandwidth available)
    - Where to request chunk (can request from URL server that is “close” to client or has high available bandwidth)