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:

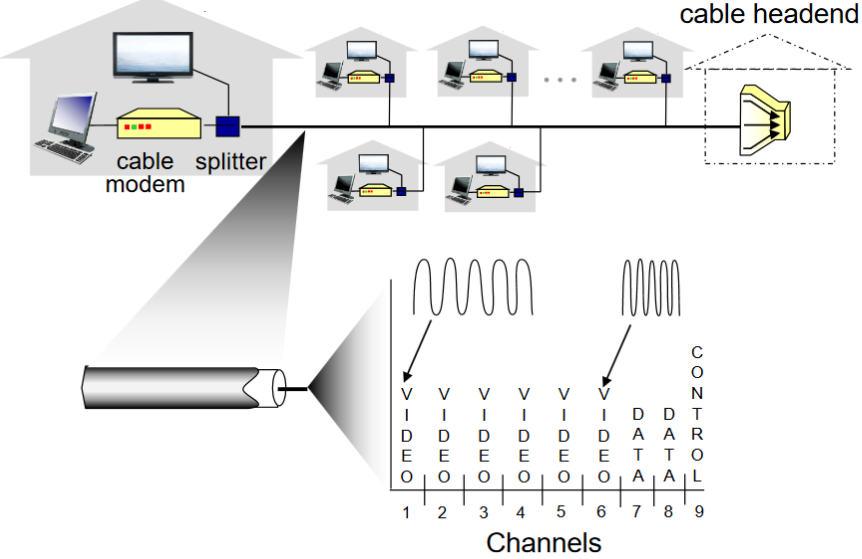
* Interconnected routers
* Network of networks

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

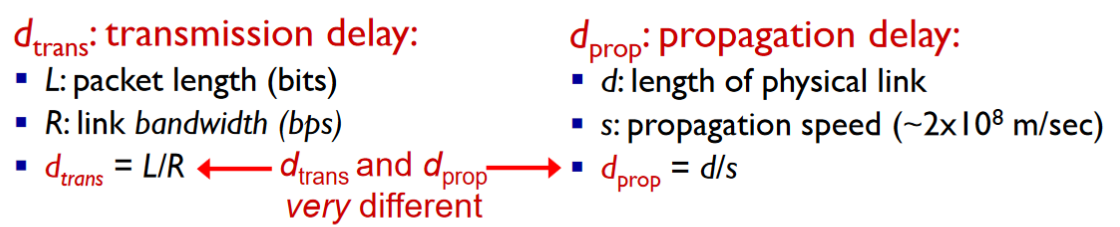
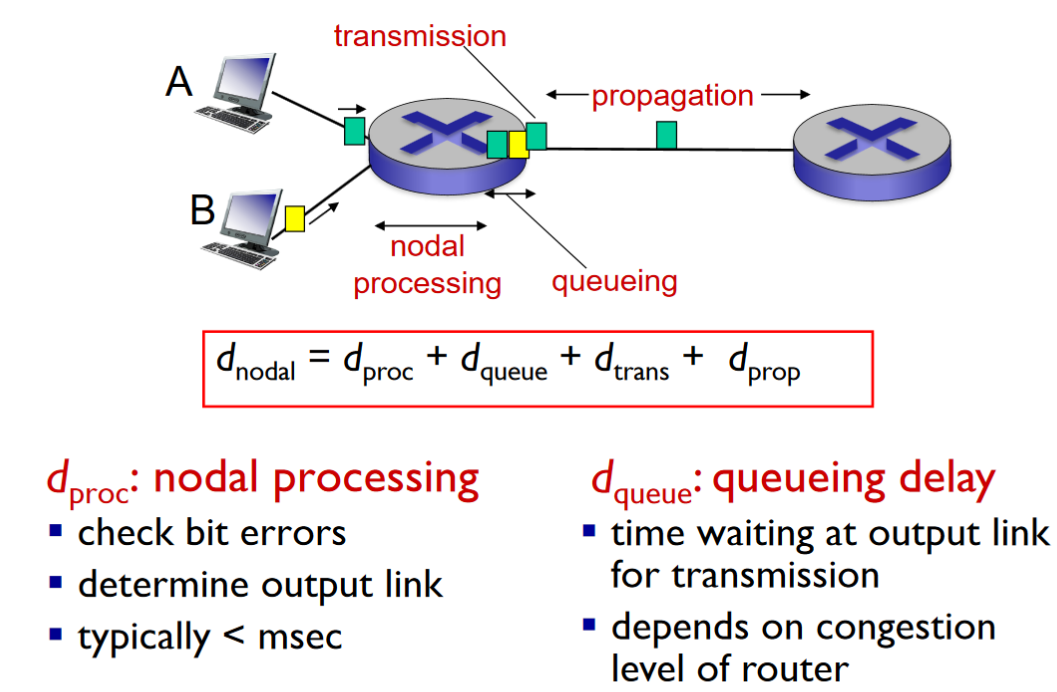
End-end delay = 2L/R (assuming zero propagation delay)

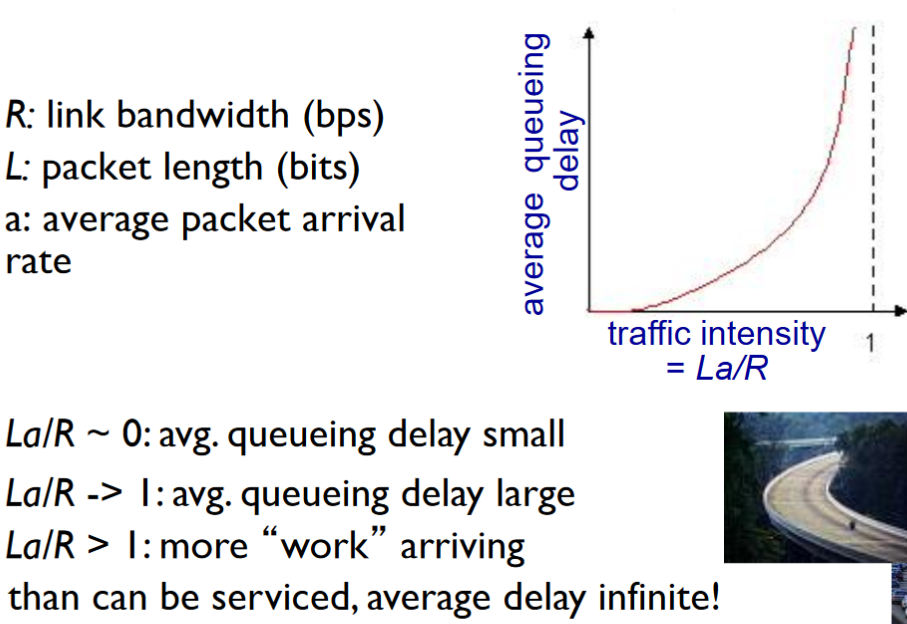
Routing: determines source-destination route taken by packets through algorithms

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

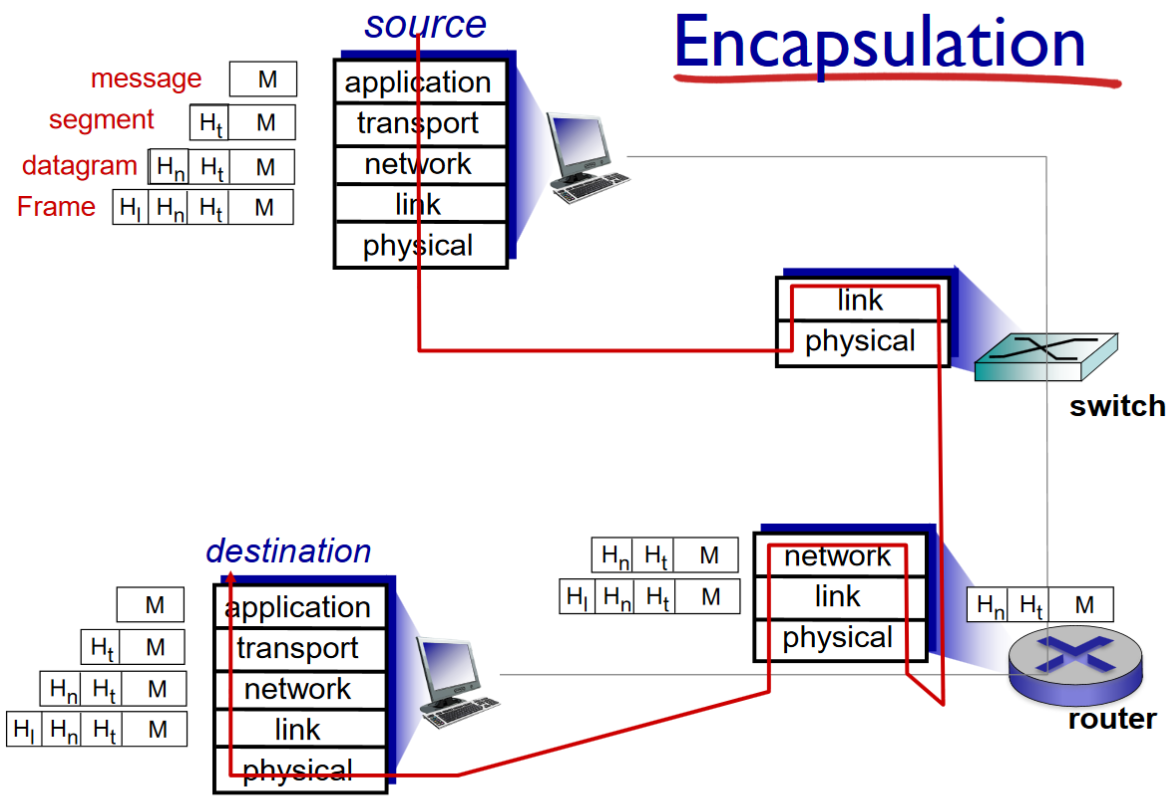
* Instantaneous: rate at given point in time
* Average: rate over longer period of time

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

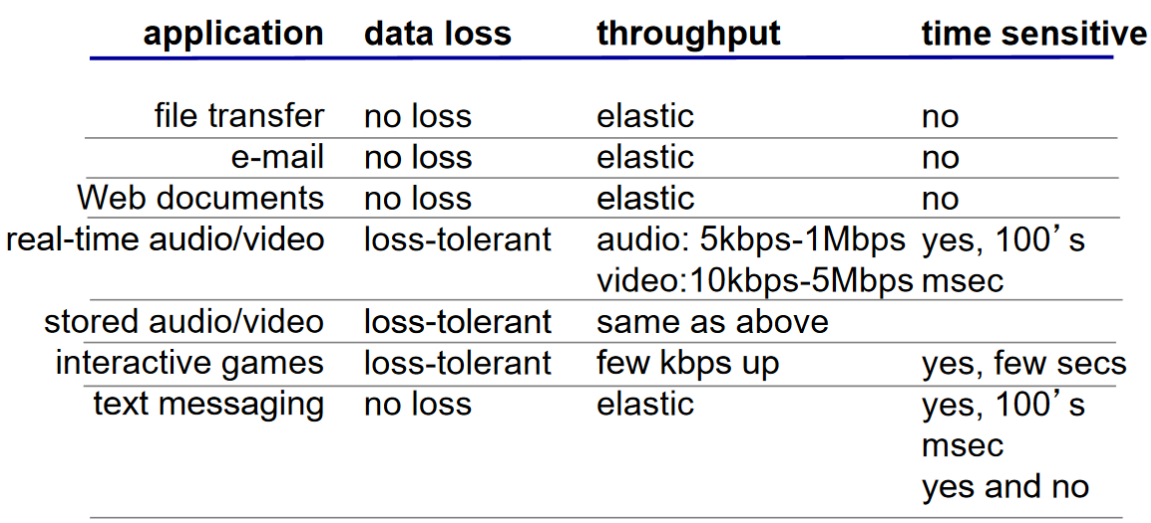
P2P architecture

* 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)

Chapter 3

Transport services and protocols: provide logical communication between app processes running on different hosts.

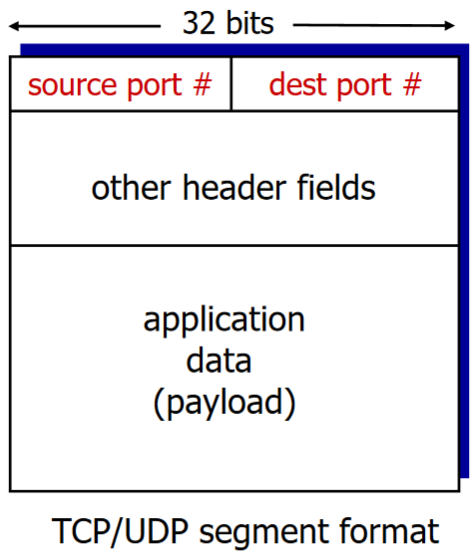
Network layer: logical communication between hosts

Transport layer: logical communication between processes

Multiplexing: handle data from multiple sockets, add transport header

Demultiplexing: Use header info to deliver received segments to correct socket

* Host receives IP datagrams
* Each datagram has source IP address, destination IP address
* Each datagram carries one transport-layer segment
* Each segment has source, destination port number
* Host uses IP addresses & port numbers to direct segment to appropriate socket



Connectionless demux: must specify destination IP address and port #

Connection-oriented demux: 4-tuple – Source IP address, port #, dest IP, port #. Each socket identified by its own 4-tuple (web server)

UDP: User Datagram Protocol

* Segments can be lost and delivered out of order
* Connectionless: no handshaking, each UDP segment handled independently of others
* Add reliability at application layer

Why is UDP used:

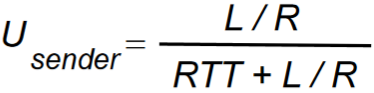
* No connection establishment
* Simple: no connection state at sender, receiver
* Small header size
* No congestion control: can blast away as fast as desired

How to recover from errors:

* Acknowledgements (ACKs): receiver explicitly tells sender that pkt received OK
* Negative acknowledgements (NAKs): receiver explicitly tells sender that pkt had errors
* Sender retransmits pkt on receipt of NAK

Handling duplicates:

* Retransmits current pkt if ACK/NAK corrupted
* Adds sequence number to each pkt
* Stop and wait: sender sends one packet, then waits for receiver response



Pipelining: Sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts

* Range of sequence numbers must be increased
* Buffering at sender and/or receiver
* Two generic forms of pipelined protocols: go-Back-N, selective repeat

Go-back-N:

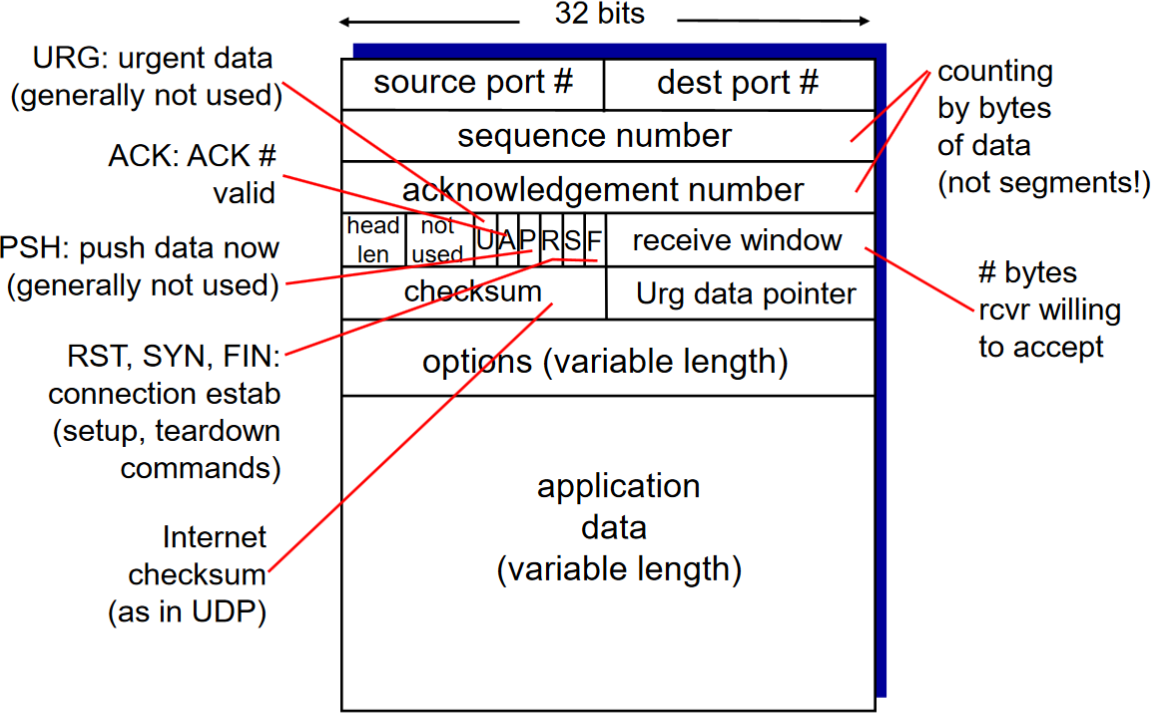
* Sender can have up to N unacked packets in pipeline
* Receiver only sends cumulative ack (doesn’t ack packet if there’s a gap)
* Sender has timer for oldest unacked packet (when time expires, retransmit all unacked packets)

Selective Repeat:

* Sender can have up to N unacked packets in pipeline
* Rcvr sends individual ack for each packet
* Sender maintains timer for each unacked packet (when timer expires, retransmit only that unacked packet)

TCP: Overview

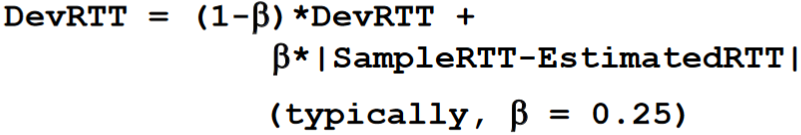
* Point-to-point: one sender, one receiver
* Reliable, in-order byte stream: no “message boundaries”
* Pipelined: TCP congestion and flow control set window size
* Full duplex data: bi-directional data flow in same connection
* MSS: maximum segment size
* Connection-oriented: handshaking inits sender, receiver state before data exchange
* Flow controlled: sender will not overwhelm receiver





Alpha typically = 0.125

Timeout interval: EstimatedRTT plus “safety margin” (4 \* DevRTT)



TCP creates rdt service on top of IP’s unreliable service

* Pipelined segments
* Cumulative acks
* Single retransmission timer

Congestion: too many sources sending too much data too fast for the network to handle

* Lost packets (buffer overflow at routers)
* Long delays (queuing in router buffers)

TCP congestion control:

* Approach: sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
* Additive increase: increase cwnd by 1 MSS every RTT until loss detected
* Multiplicative decrease: cut cwnd in half after loss

Network-assisted congestion control:

* Two bits in IP header (ToS field) marked by network router to indicate congestion
* Congestion indication carried to receiving host
* Receiver (seeing congestion indication in IP datagram) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion

Chapter 4

Network-layer functions:

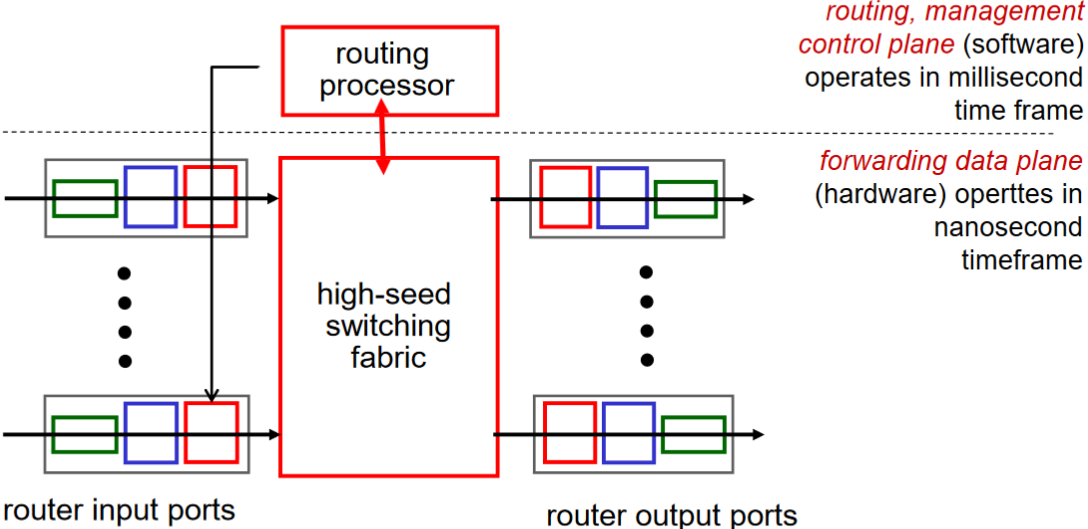
* Forwarding: move packets from router’s input to appropriate router output
* Routing: determine route taken by packets from source to destination

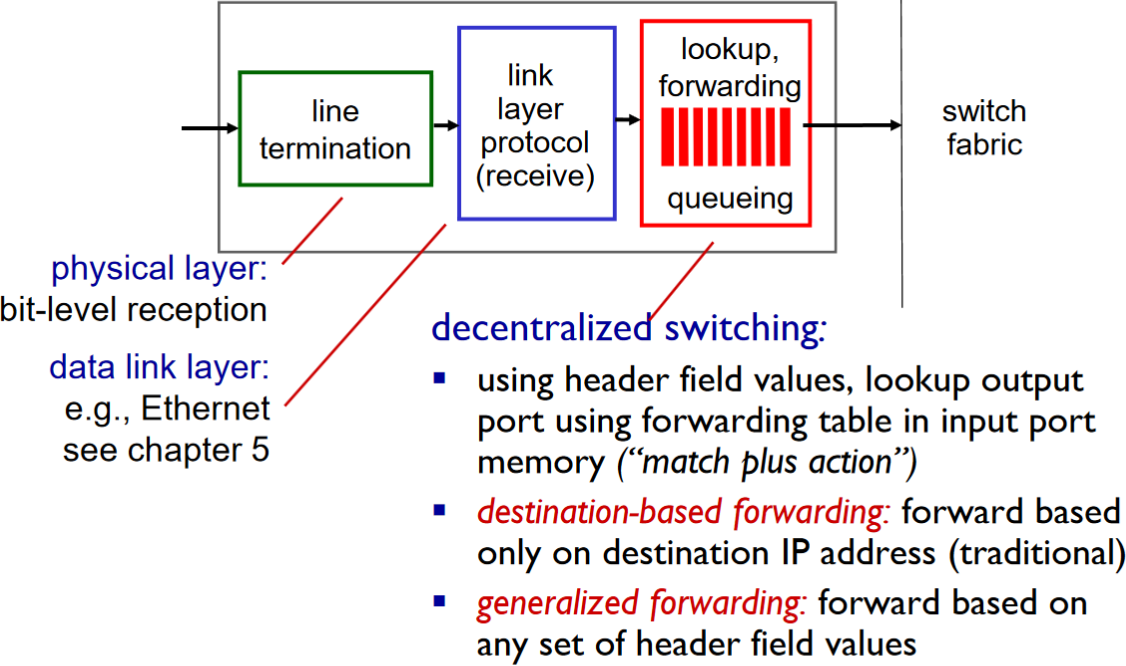
Data plane:

* local, per-router function
* Determines how datagram arriving on router input port is forwarded to router output port
* Forwarding function

Control plane:

* Network-wide logic
* Determines how datagram is routed among routers along end-end path from source host to destination host
* Two control-plane approaches:
* Traditional routing algorithms: implemented in routers
* Software-defined networking (SDN): implemented in (remote) servers





Switching rate: rate at which packets can be transferred from inputs to outputs (memory, bus, crossbar)

First generation routers:

* Traditional computers with switching under direct control of CPU
* Packet copied to system’s memory
* Speed limited by memory bandwidth (2 bus crossings per datagram)

Switching via a bus

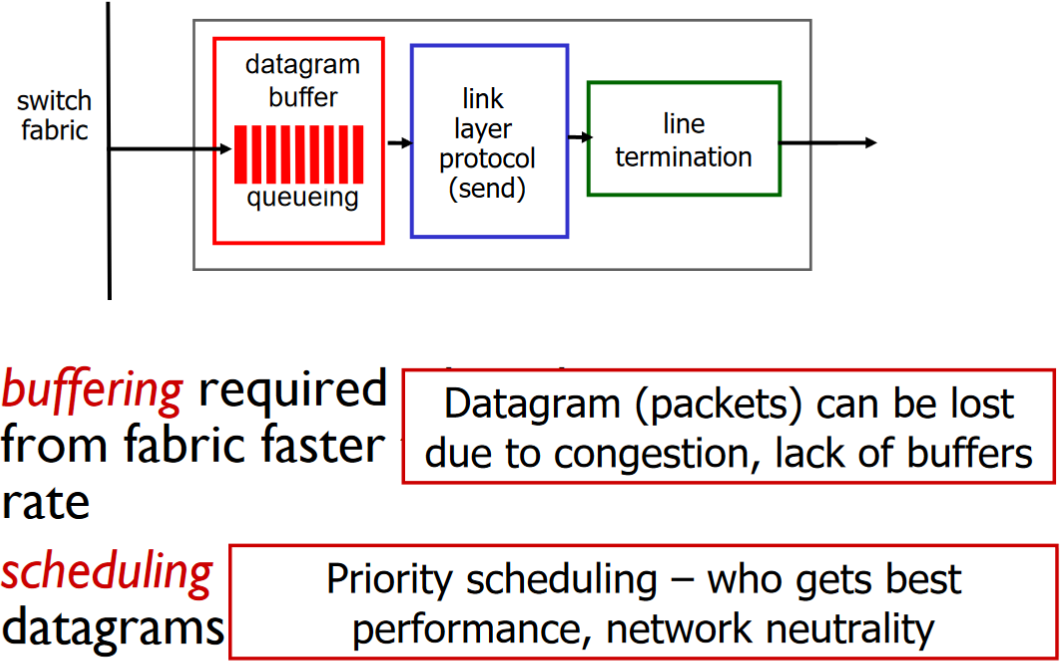
* Datagram from input port memory to output port memory via a shared bus
* Bus contention: switching speed limited by bus bandwidth

Switching via interconnection network

* Overcome bus bandwidth limitations
* Banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
* Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric

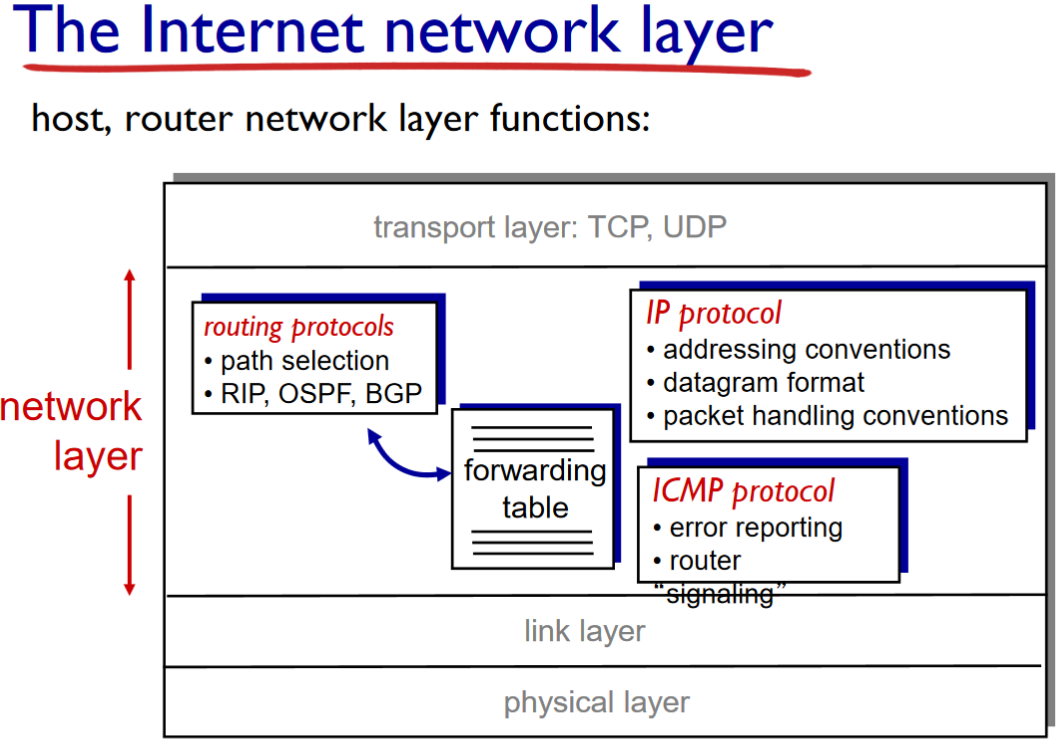
Input port queuing

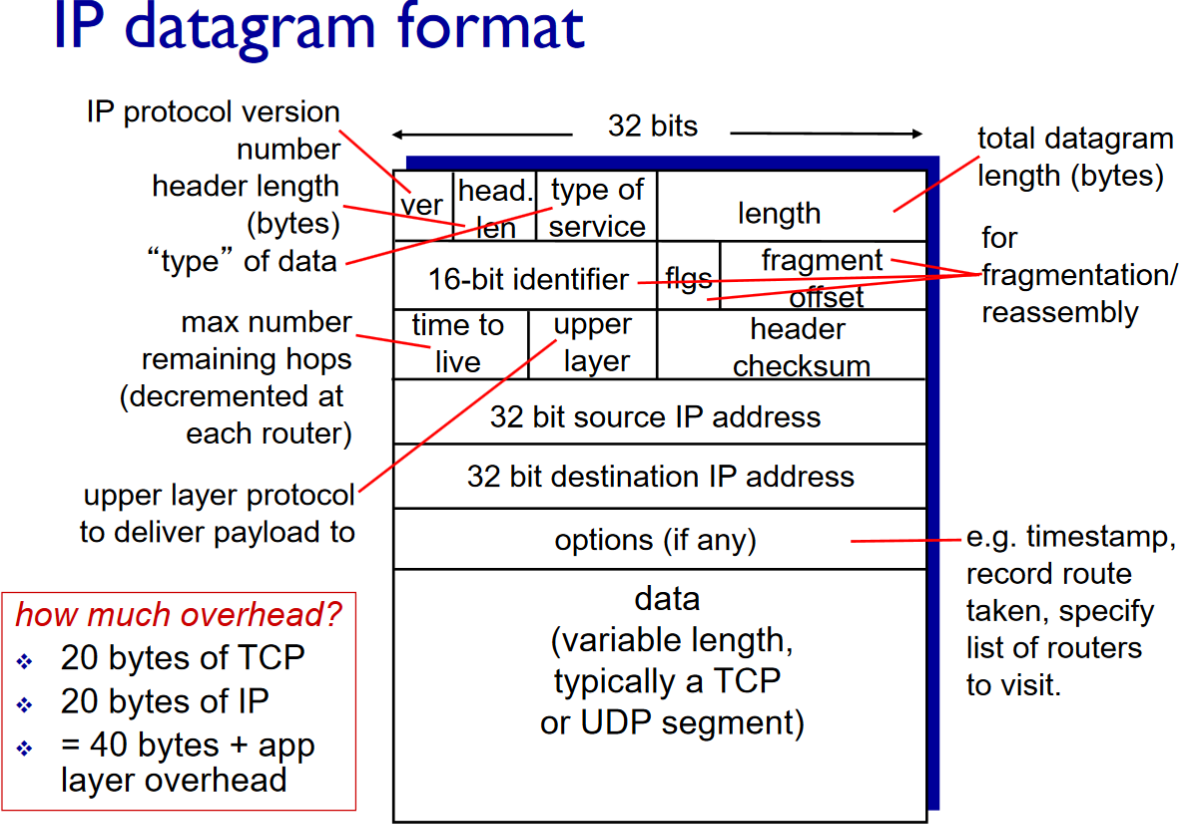
* Fabric slower than input ports combined -> queuing may occur at input queues (buffer overflow)
* Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



Scheduling mechanisms

* FIFO scheduling
* Priority scheduling:
* Multiple classes, with different priorities (marking, header info (source/dest, port,etc))
* Round Robin (RR)
* Multiple classes
* Cyclically scan class queues, sending one complete packet from each class (if available)
* Weighted Fair Queuing (WFQ)
* Generalized Round Robin
* Each class gets weighted amount of service in each cycle





Interface: Connection between host/router and physical link

Subnet: device interfaces with the same subnet part of IP address (reach without router)

CIDR: Classless InterDomain Rounting

* Subnet portion of address of arbitrary length
* Address format: a.b.c.d/x, where x is # bits in subnet portion of address

DHCP: Dynamic Host Configuration Protocol: dynamically get address from a server (“plug-and-play”)

Hierarchical addressing: allows efficient advertisement of routing information

ICANN: Internet Corporation for Assigned Names and Numbers

* Allocates addresses
* Manages DNS
* Assigns domain names, resolves disputes

NAT: network address translation: Local network uses just one IP address as far as outside world is concerned:

* Range of addresses not needed from ISP: just one IP address for all devices
* Can change addresses of devices in local network without notifying outside world
* Can change ISP without changing addresses of devices in local network
* Devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT router must:

* Outgoing datagrams: replace (source IP address, port#) of every outgoing datagram to (NAT IP address, new port #), remote clients/servers will respond using (NAT IP address, new port #) as destination addr
* Remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
* Incoming datagrams: replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT:

* 16-bit port-number field:
* 60 000 simultaneous connections with a single LAN-side address
* Controversial:
* Routers should only process up to layer 3
* Address shortage should be solved by IPv6
* Violates end-to-end argument
* NAT possibility must be considered by app designers, eg P2P applications
* NAT traversal: what if client wants to connect to server behind NAT?

IPv6 motivation:

* 32-bit address space soon to be completely allocated
* Header format helps speed processing/forwarding
* Header changes to facilitate QoS

IPv6 datagram format:

* Fixed-length 40 byte header
* No fragmentation allowed

