Computer Communication Networks

Final Review

ICEN/ICSI 416 – Fall 2016

Prof. Dola Saha

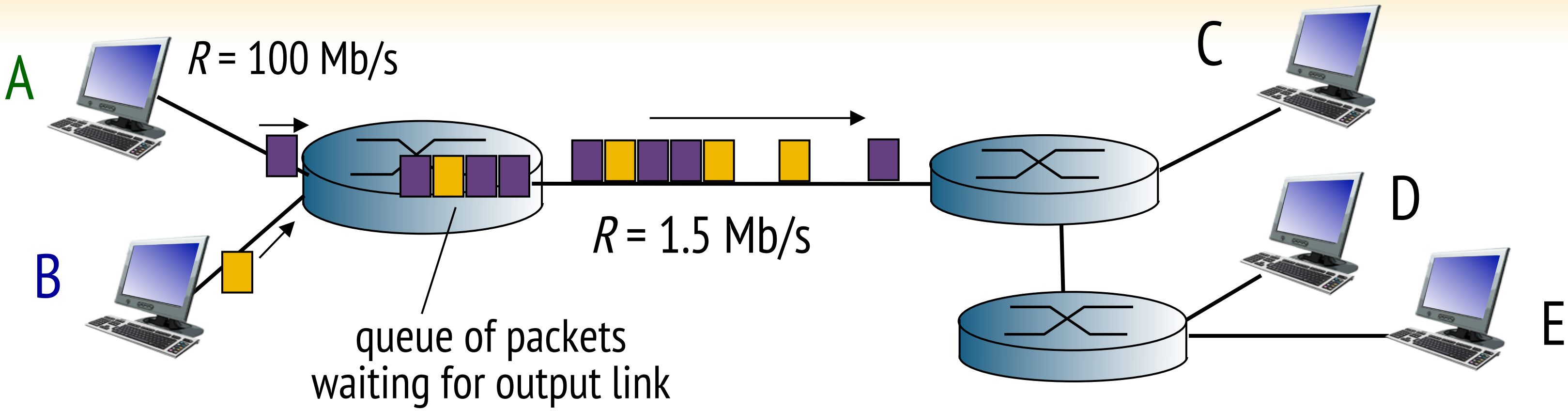
UNIVERSITY AT ALBANY State University of New York

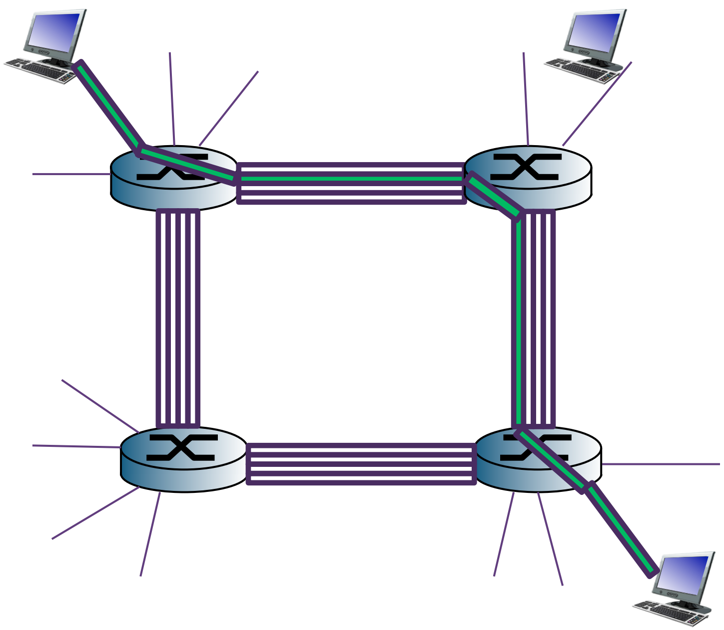
What is included?

- Foundation
- Application Layer
- Transport Layer
- Network Layer
- Link Layer
- Physical Layer
- Network Security

- The material covered by Prof. Hany Elgala will **NOT** be included in the midterm.

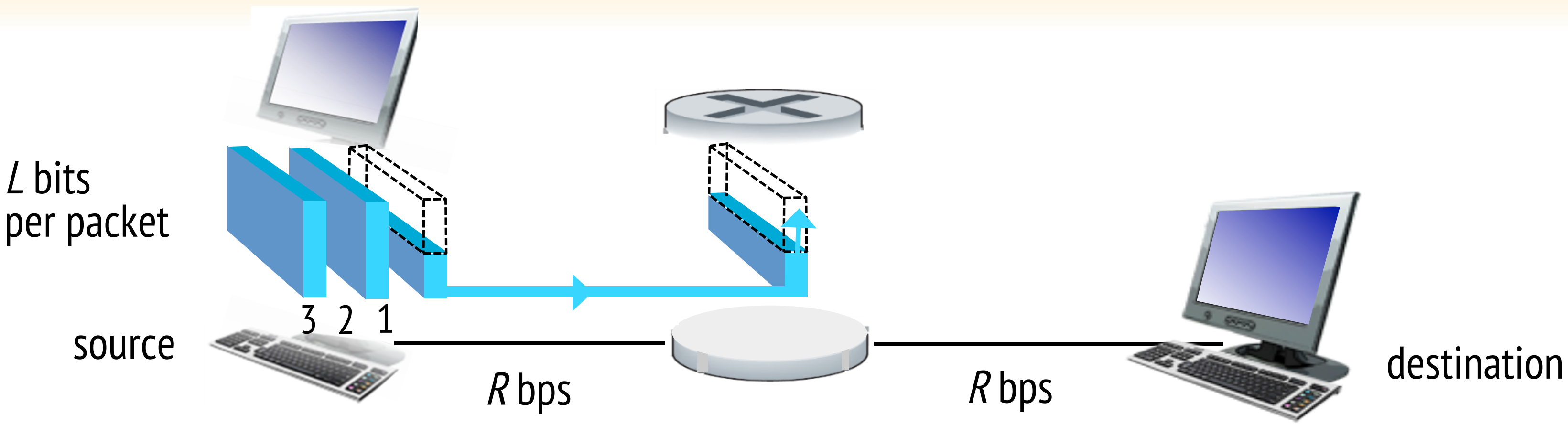
Packet Switching vs Circuit Switching





- Advantages
- Disadvantages

Packet-switching: store-and-forward



takes L/R seconds to transmit (push out) L-bit packet into link at R bps

store and forward: entire packet must arrive at router before it can be transmitted on next link

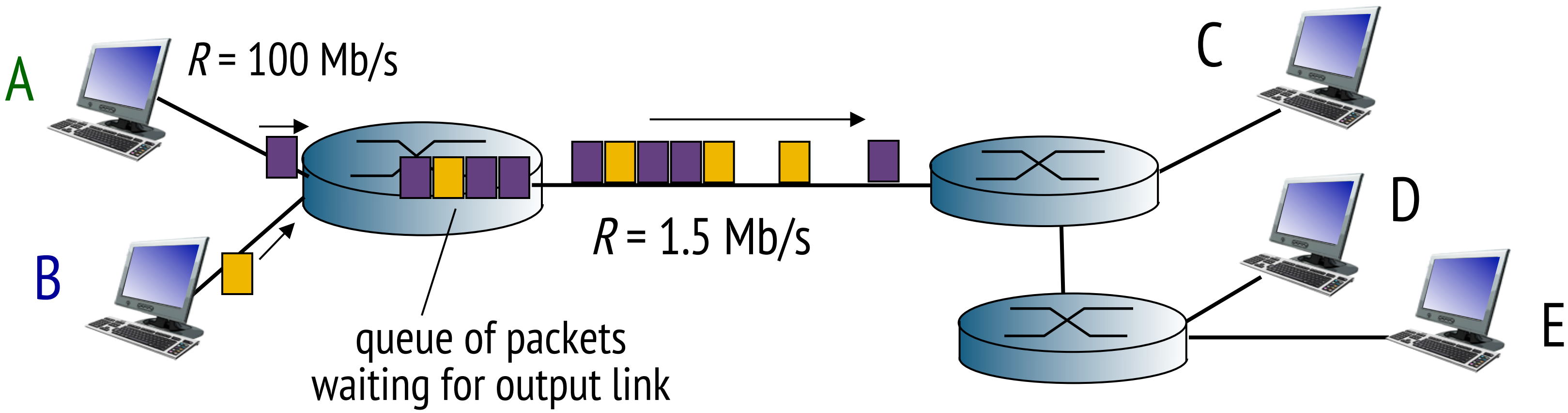
- end-end delay = $2L/R$ (assuming zero propagation delay)

more on delay shortly …

one-hop numerical example:

- L = 7.5 Mbits
- R = 1.5 Mbps
- one-hop transmission delay = 5 sec

Packet Switching: queueing delay, loss

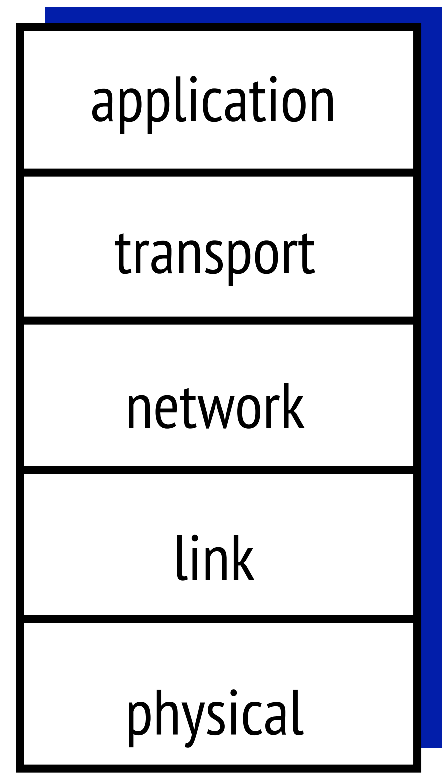


queuing and loss:

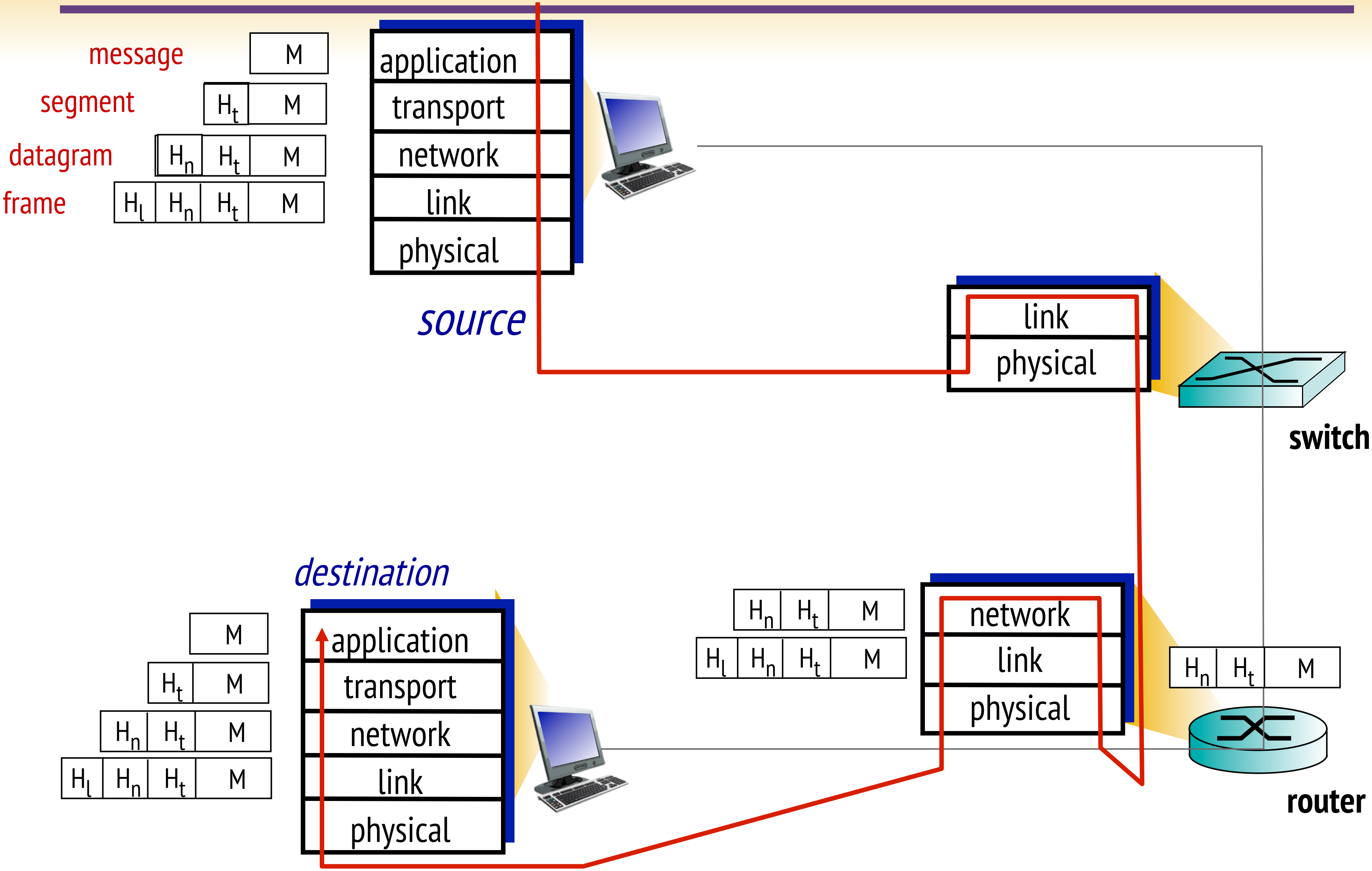
- If arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
  - packets will queue, wait to be transmitted on link
  - packets can be dropped (lost) if memory (buffer) fills up

Internet Protocol Stack

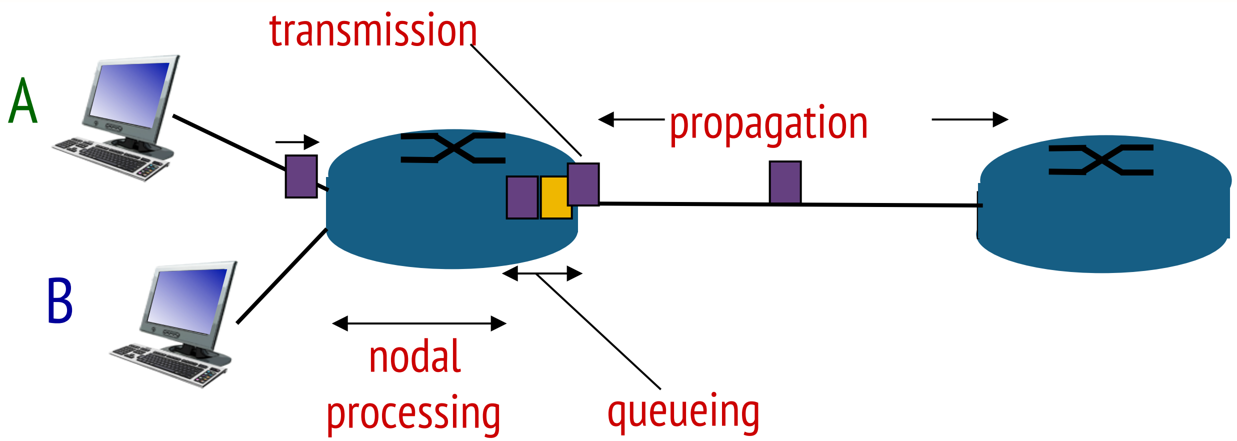
- application: supporting network applications
  - FTP, SMTP, HTTP
- transport: process-process data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi)
- physical: bits “on the wire” / “over the air”



Encapsulation



Four Sources of Packet Delay



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

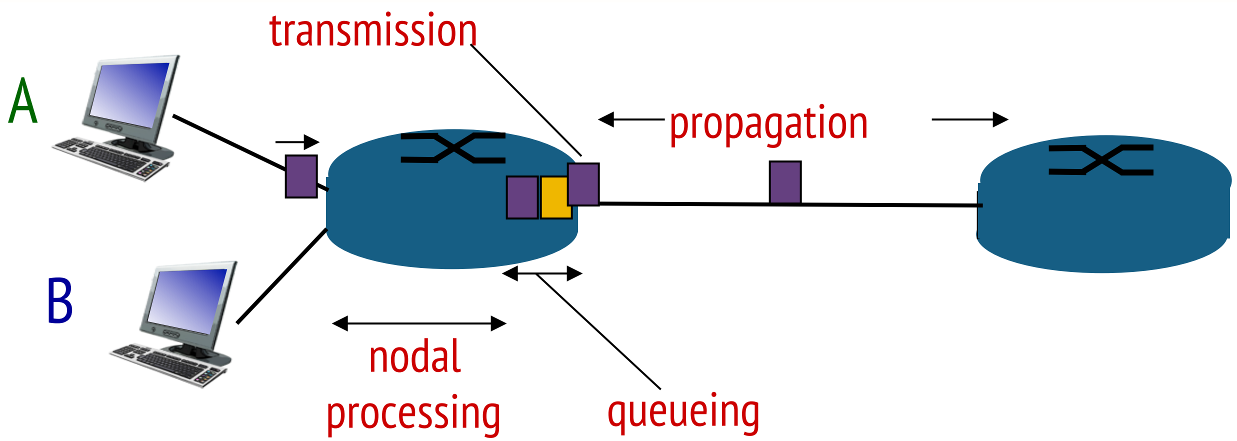
d_{proc}: nodal processing

- check bit errors
- determine output link
- typically < msec

d_{queue}: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

Four Sources of Packet Delay



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

d_{trans}: transmission delay:

- L: packet length (bits)
- R: link *bandwidth (bps)*
- $d_{trans} = L/R$

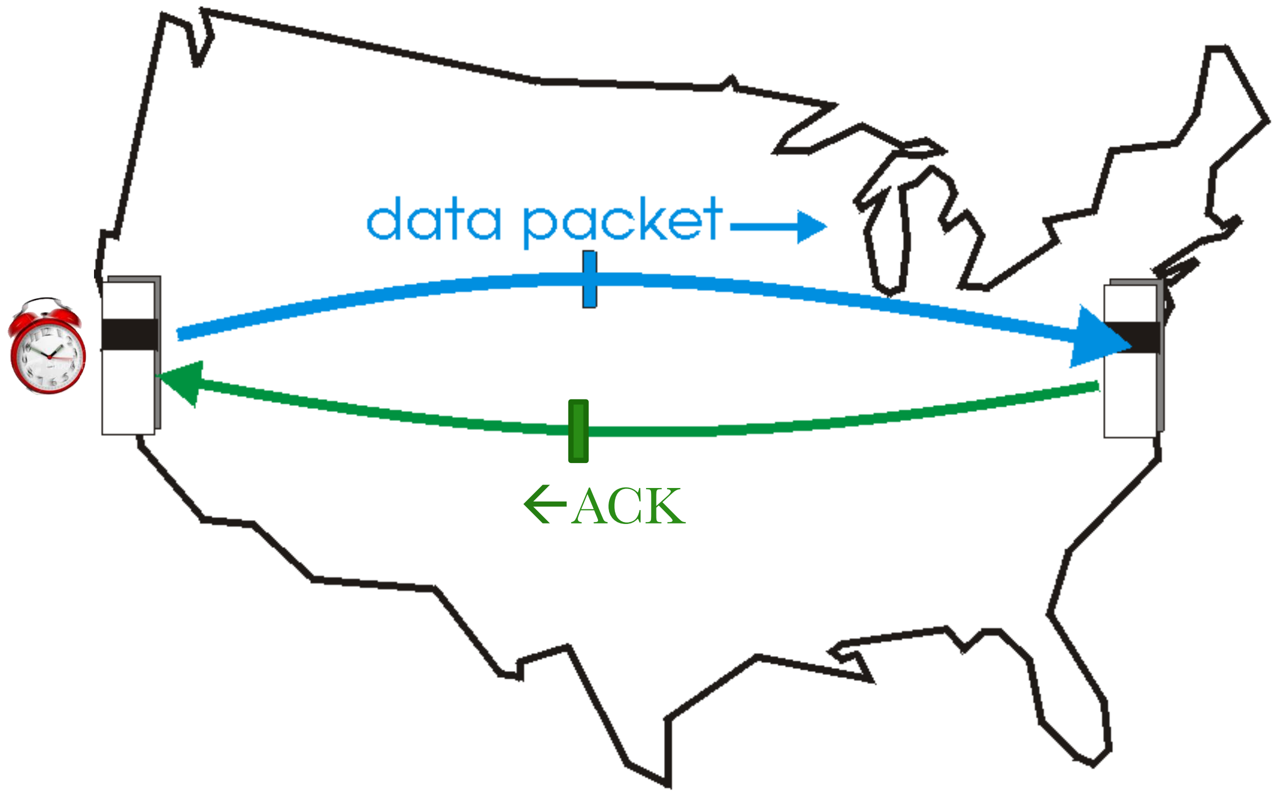
d_{trans} and d_{prop} *very* different

d_{prop}: propagation delay:

- d: length of physical link
- s: propagation speed in medium (~2×10^8 m/sec)
- $d_{\text{prop}} = d/s$

Round Trip Time (RTT)

- Time:
  - From packet starting to leave a node
  - To response came back to the same node



Persistent and non-persistent HTTP

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:

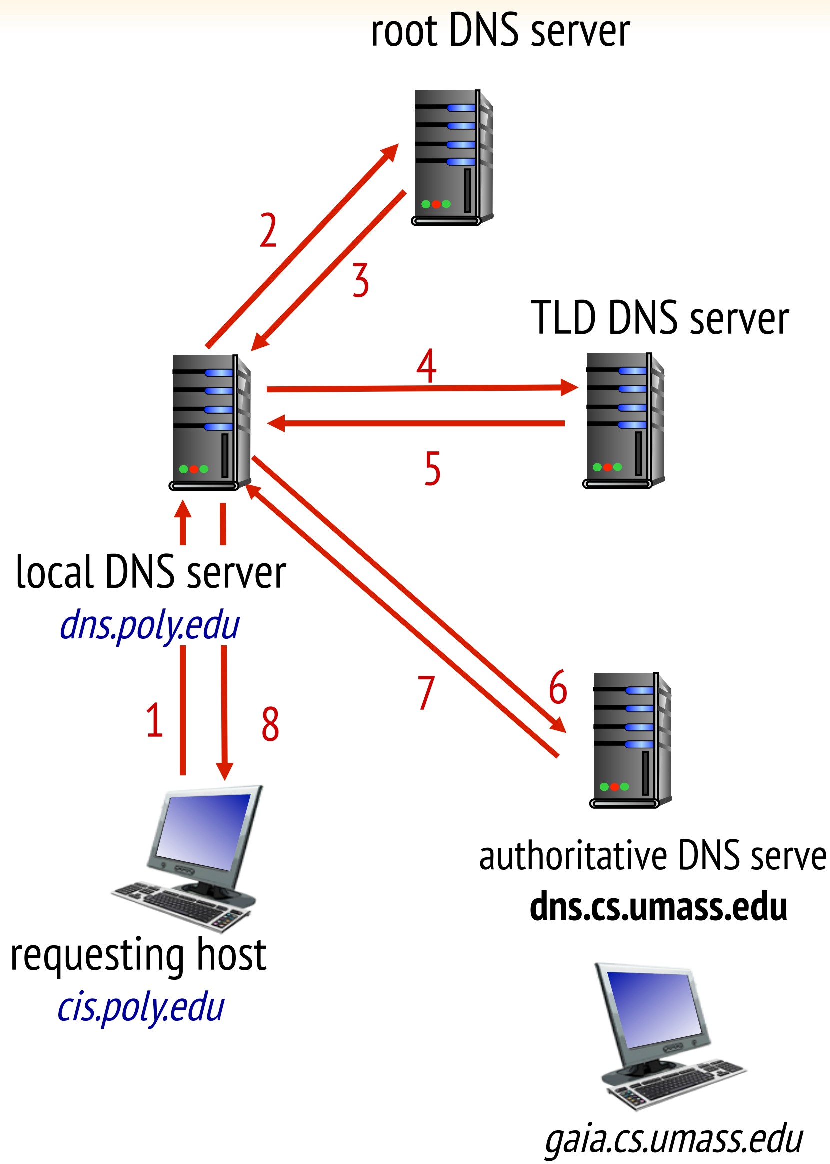
- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

DNS name resolution example

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



BitTorrent: requesting, sending file chunks

requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

sending chunks: tit-for-tat

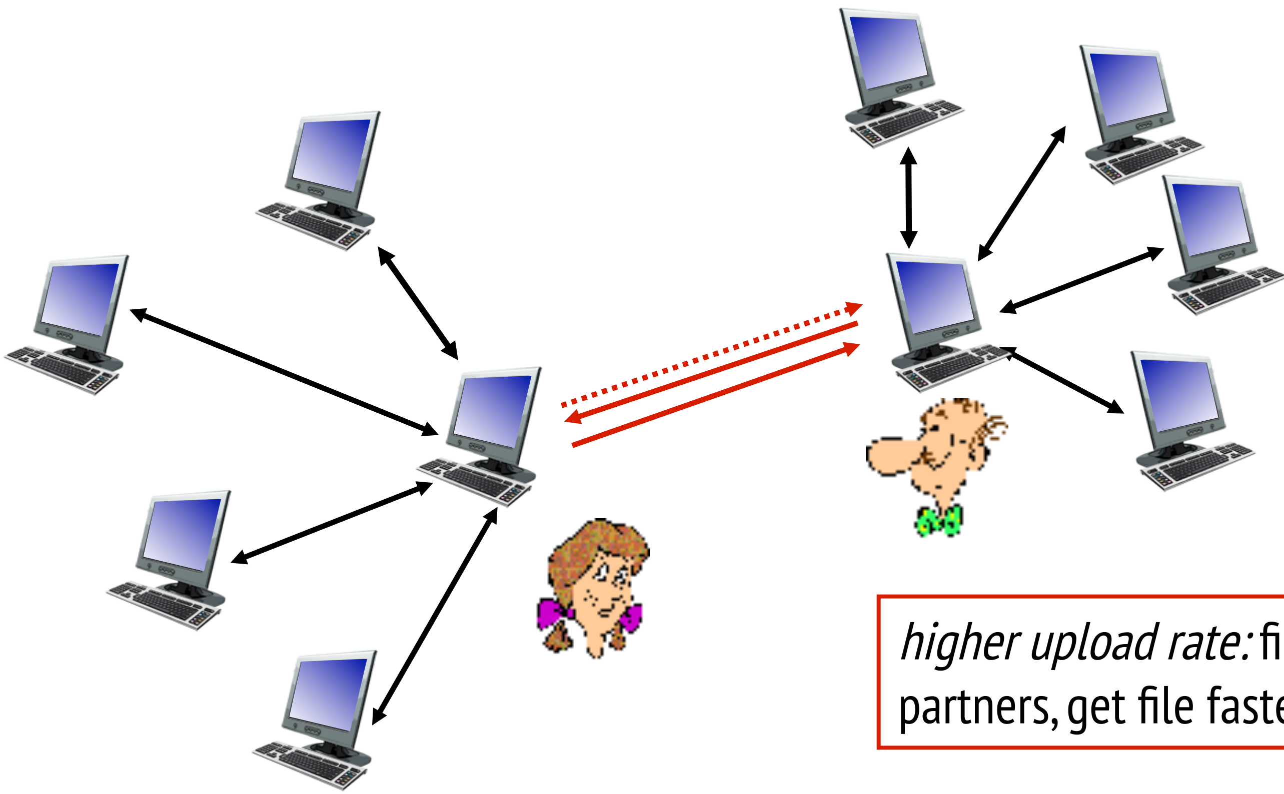
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster !

UDP: User Datagram Protocol [RFC 768]

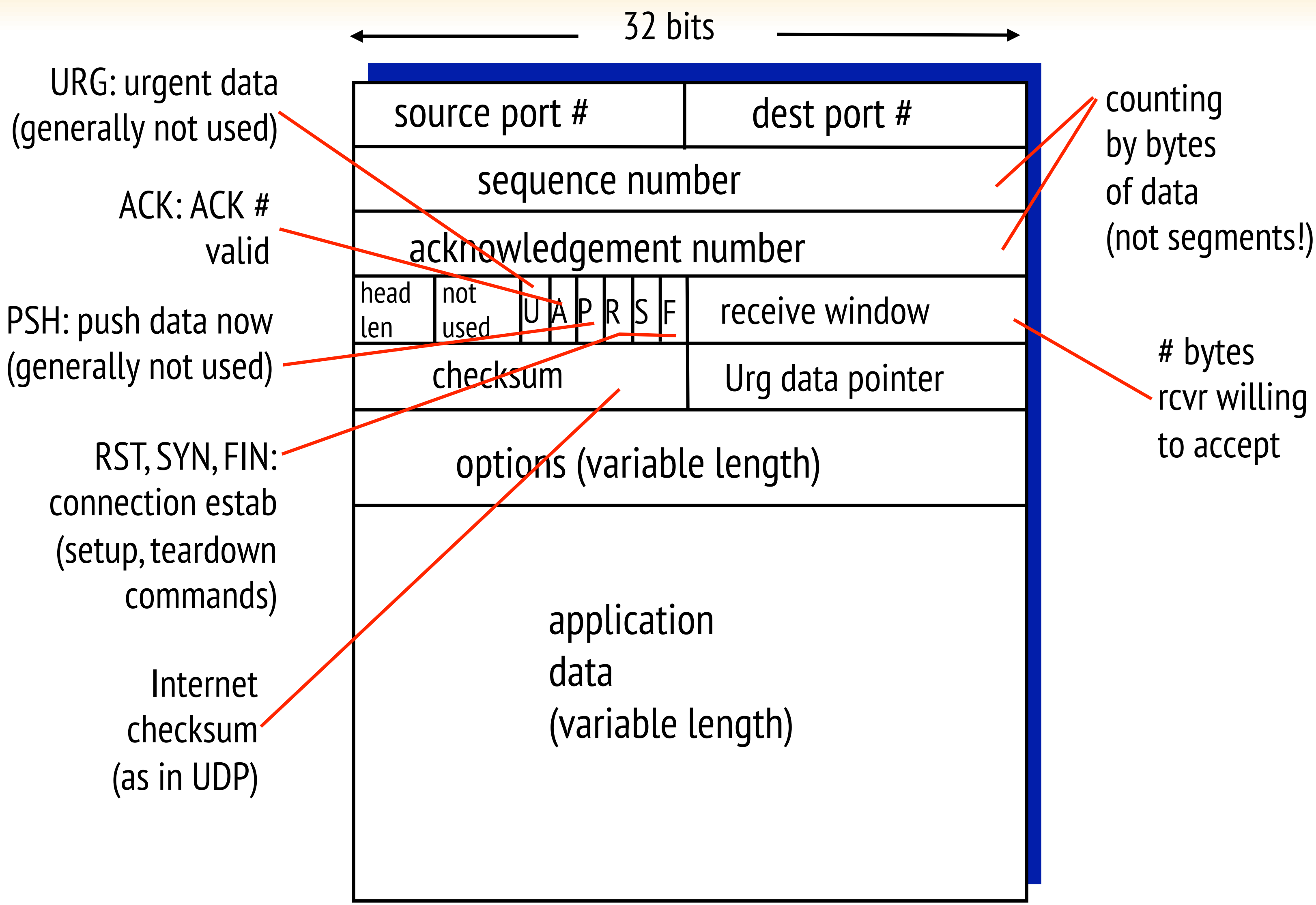
- “no frills,” “bare bones” Internet transport protocol
- “best effort” service, UDP segments may be:
  - lost
  - delivered out-of-order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

- UDP use:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - DNS
  - SNMP
- reliable transfer over UDP:
  - add reliability at application layer
  - application-specific error recovery!

TCP: Overview RFCs: 793,1122,1323, 2018, 2581

- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) inits sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver
- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size

TCP segment structure



TCP seq. numbers, ACKs

sequence numbers:

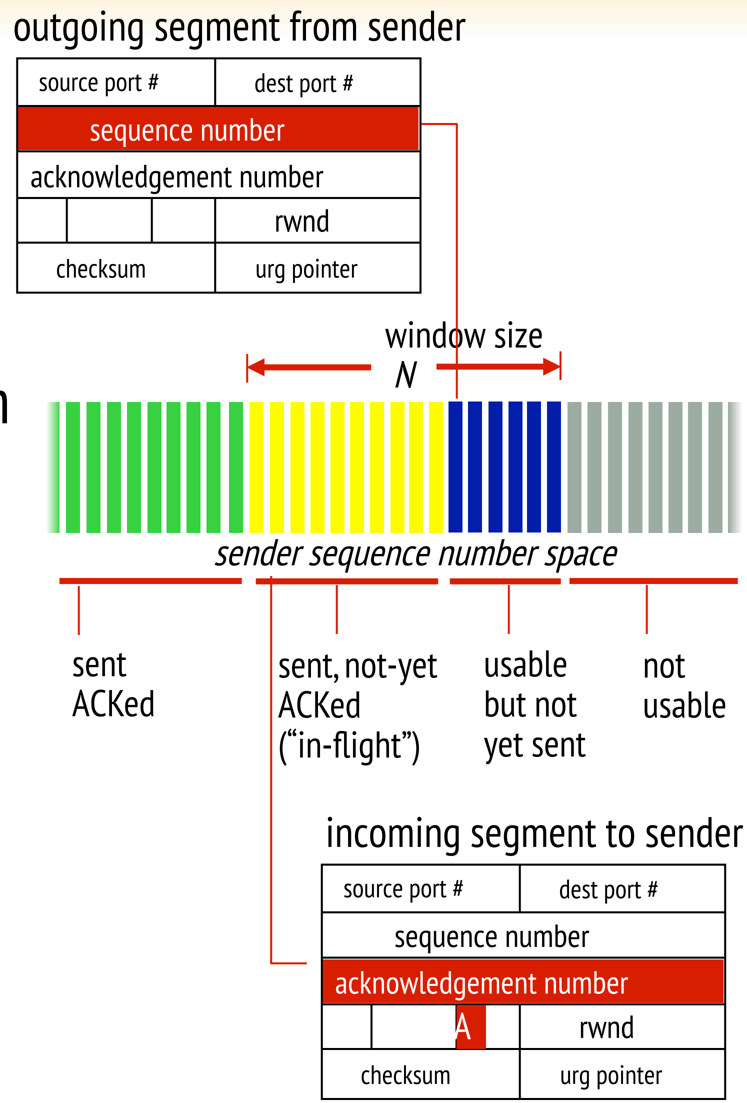
- byte stream "number" of first byte in segment's data

acknowledgements:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementor



TCP round trip time, timeout

Q: how to set TCP timeout value?

- longer than RTT
  - but RTT varies
- *too short:* premature timeout, unnecessary retransmissions
- *too long:* slow reaction to segment loss

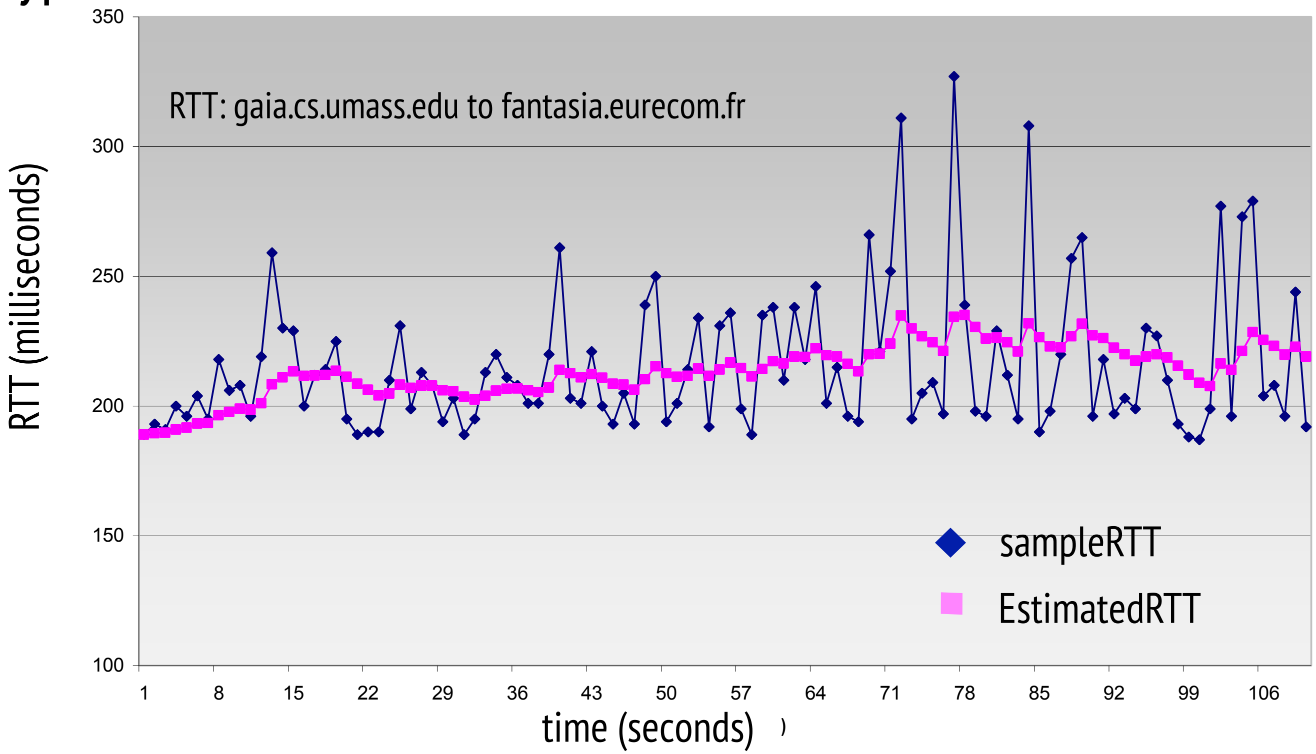
Q: how to estimate RTT?

- **SampleRTT**: measured time from segment transmission until ACK receipt
  - ignore retransmissions
- **SampleRTT** will vary, want estimated RTT "smoother"
  - average several *recent* measurements, not just current **SampleRTT**

TCP round trip time, timeout

EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT

- exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: α = 0.125



Timeout = 2*EstimatedRTT

Jacobson/Karels Algorithm

- timeout interval: **EstimatedRTT** plus “safety margin”
  - large variation in **EstimatedRTT ->** larger safety margin
- estimate SampleRTT deviation from EstimatedRTT:
- RFC 6298

DevRTT = (1-β)*DevRTT +
β*(|SampleRTT-EstimatedRTT|)

(typically, β = 0.25)

Measure of variability

TimeoutInterval = EstimatedRTT + 4*DevRTT

estimated RTT

“safety margin”

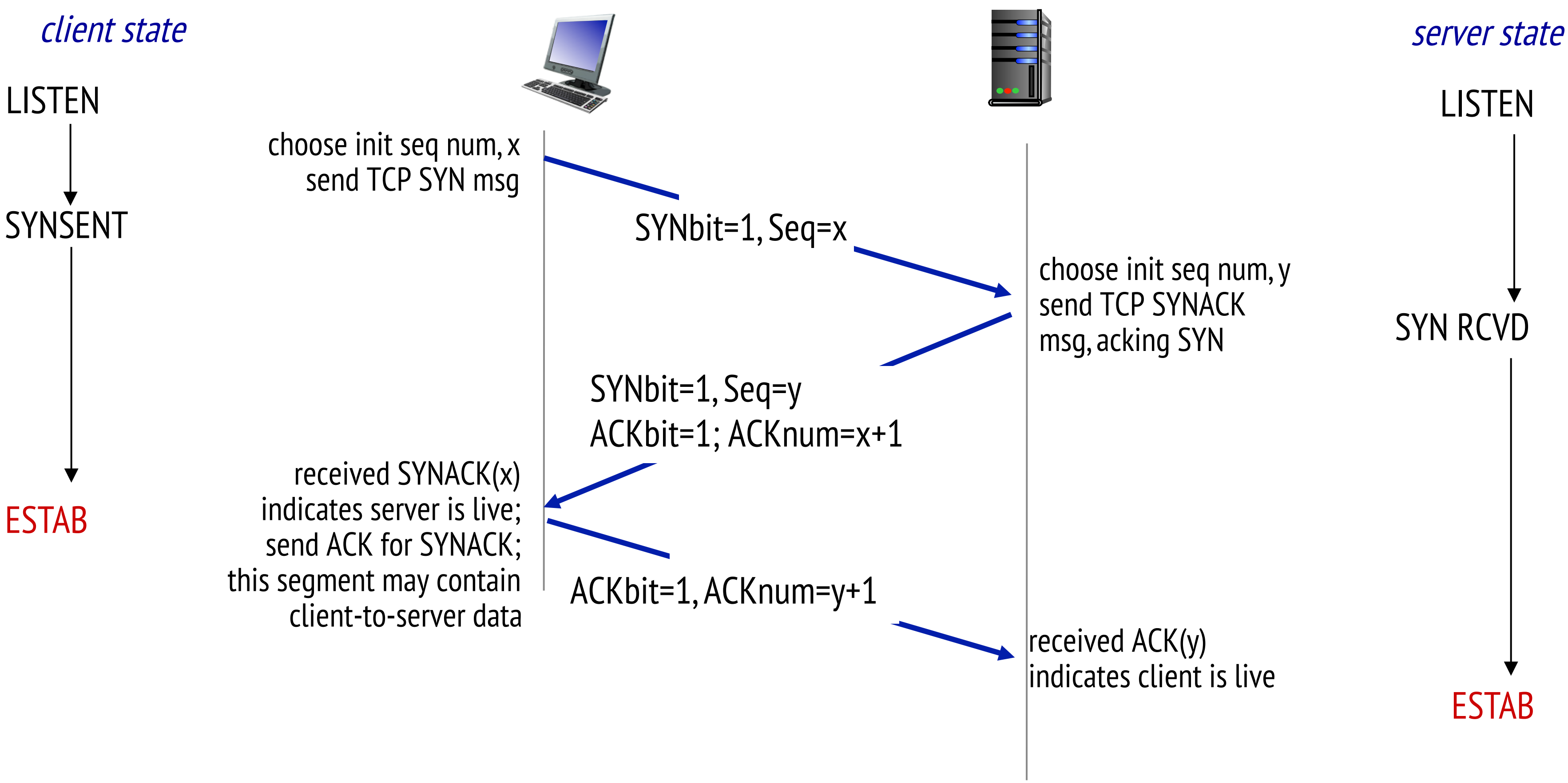
TCP Flow Control

- LastByteRcvd − LastByteRead ⩽ MaxRcvBuffer
- AdvertisedWindow = MaxRcvBuffer − ((NextByteExpected − 1) − LastByteRead)
- LastByteSent − LastByteAcked ⩽ AdvertisedWindow
- EffectiveWindow = AdvertisedWindow − (LastByteSent − LastByteAcked)
- LastByteWritten − LastByteAcked ⩽ MaxSendBuffer
- If the sending process tries to write y bytes to TCP, but

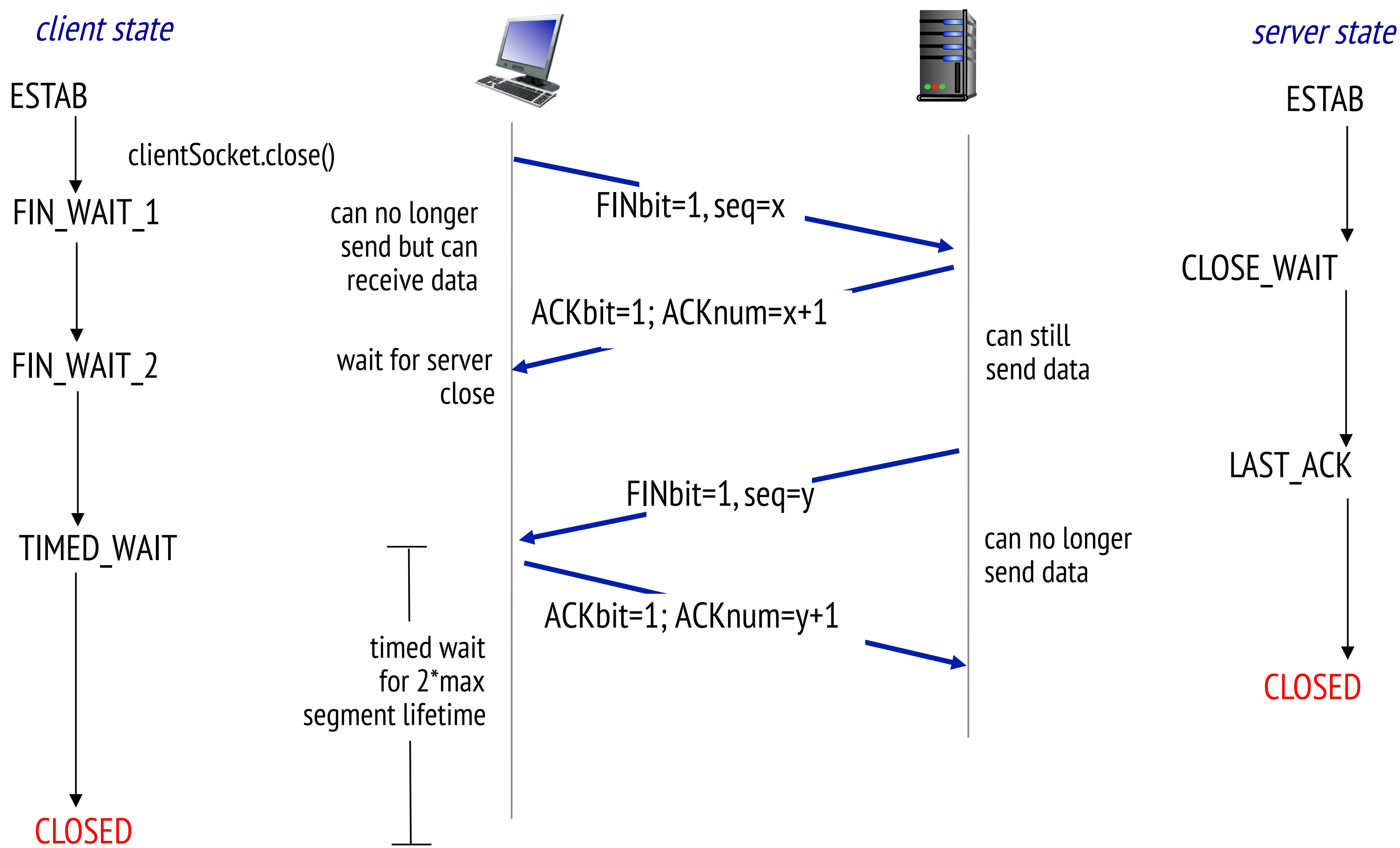
  (LastByteWritten − LastByteAcked) + y > MaxSendBuffer

  then TCP blocks the sending process and does not allow it to generate more data.

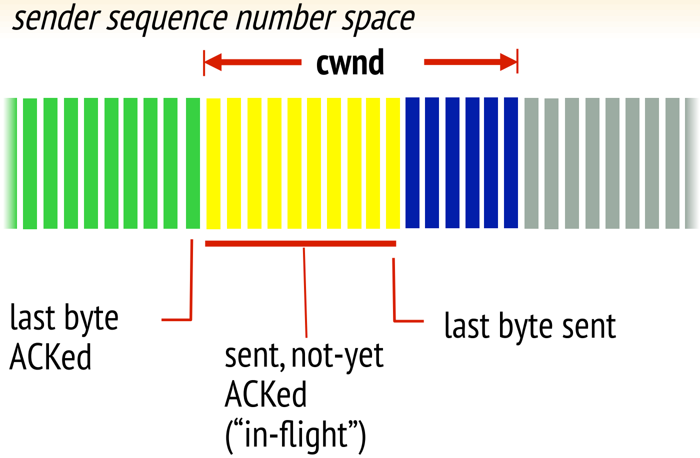
TCP 3-way handshake



TCP: closing a connection



TCP Congestion Control: details



TCP sending rate:

- *roughly:* send cwnd bytes, wait RTT for ACKS, then send more bytes

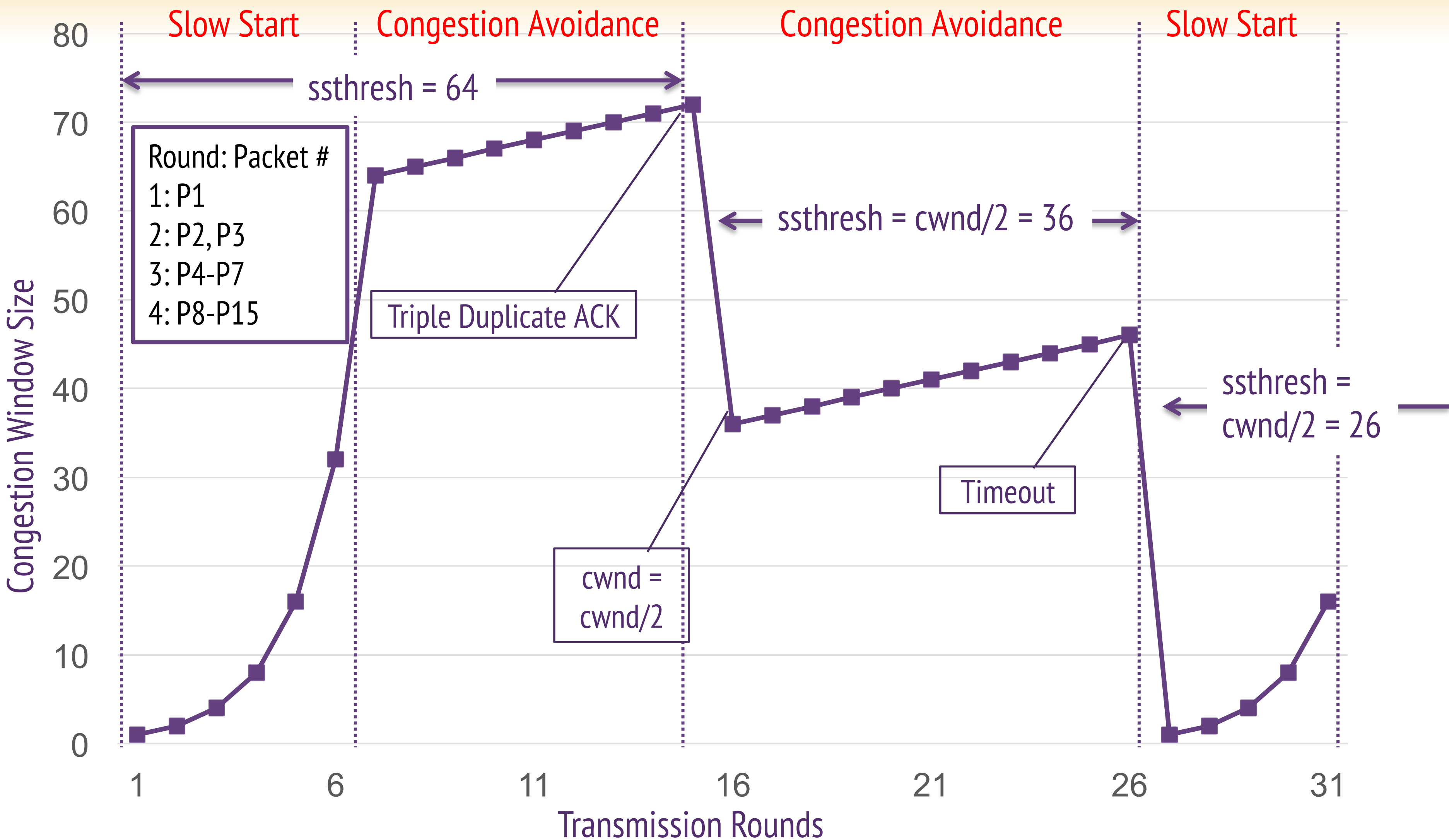
$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

- sender limits transmission:

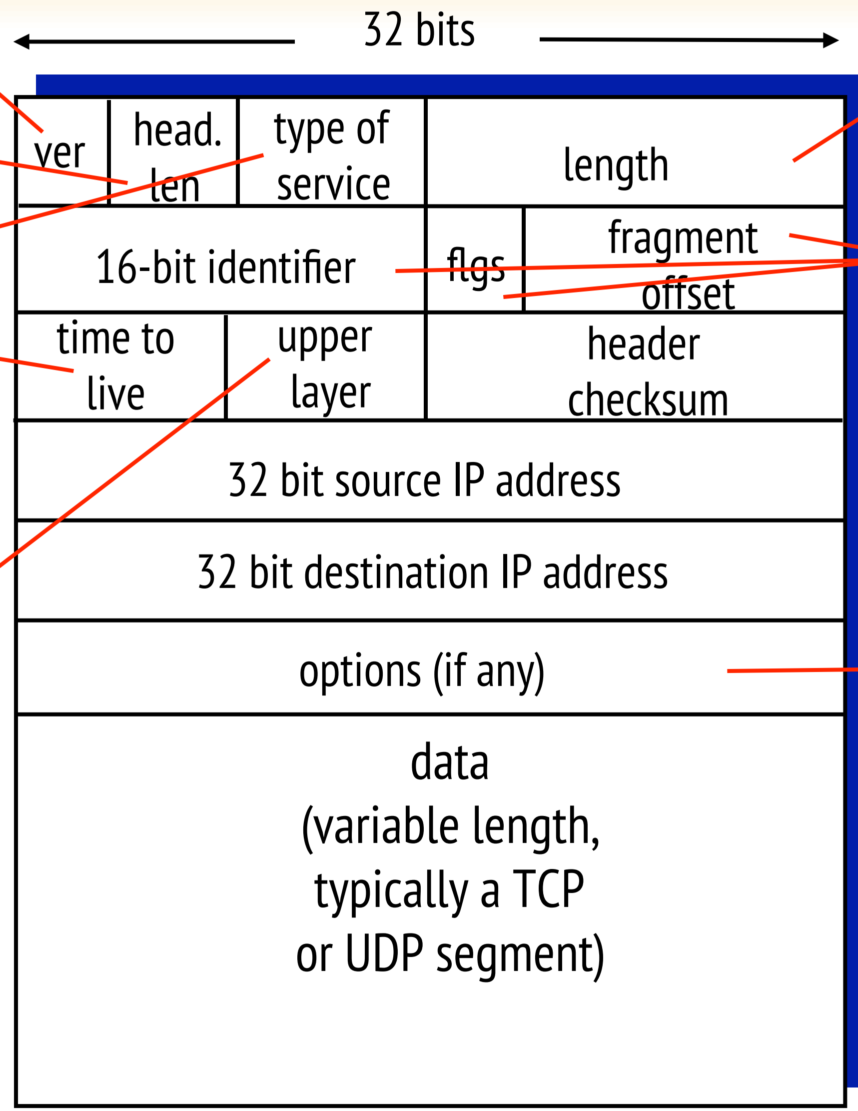
LastByteSent – LastByteAcked <= cwnd

- **cwnd** is dynamic, function of perceived network congestion

TCP Reno


Slow Start
Congestion Avoidance
Congestion Avoidance
Slow Start
ssthresh = 64
Round: Packet #
1: P1
2: P2, P3
3: P4-P7
4: P8-P15
Triple Duplicate ACK
ssthresh = cwnd/2 = 36
ssthresh =
cwnd/2 = 26
Timeout
cwnd =
cwnd/2
Congestion Window Size
Transmission Rounds
0
10
20
30
40
50
60
70
80
1
6
11
16
21
26
31

IP datagram format

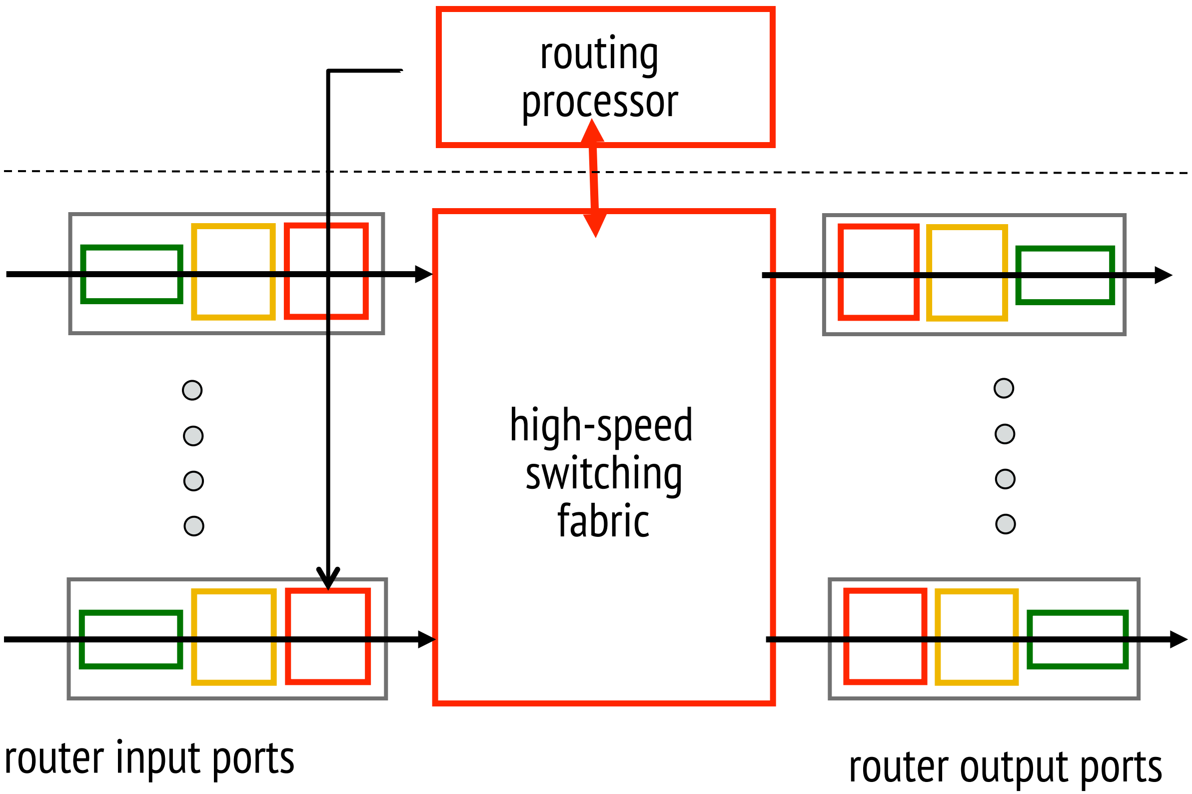


how much overhead?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

Router architecture overview

- high-level view of generic router architecture:



routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operttes in nanosecond timeframe

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link Interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

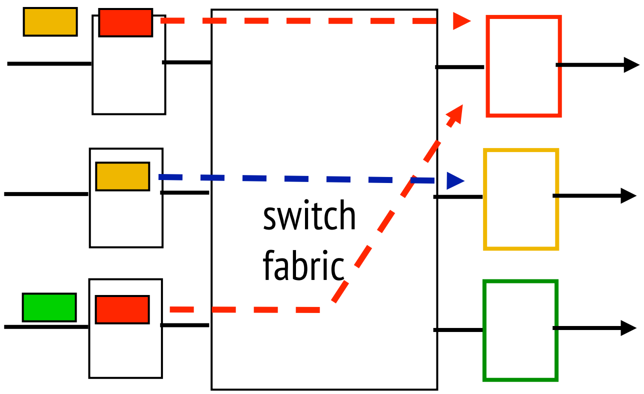
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

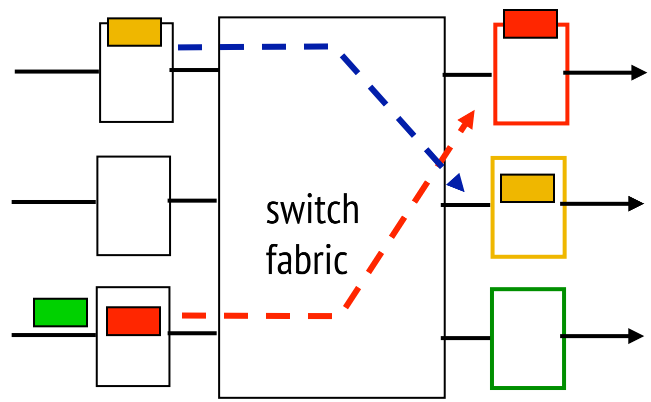
DA: 11001000 00010111 00011000 10101010 which interface?

Input port queuing

- fabric slower than input ports combined -> queueing may occur at input queues
  - *queueing delay and loss due to input buffer overflow!*
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

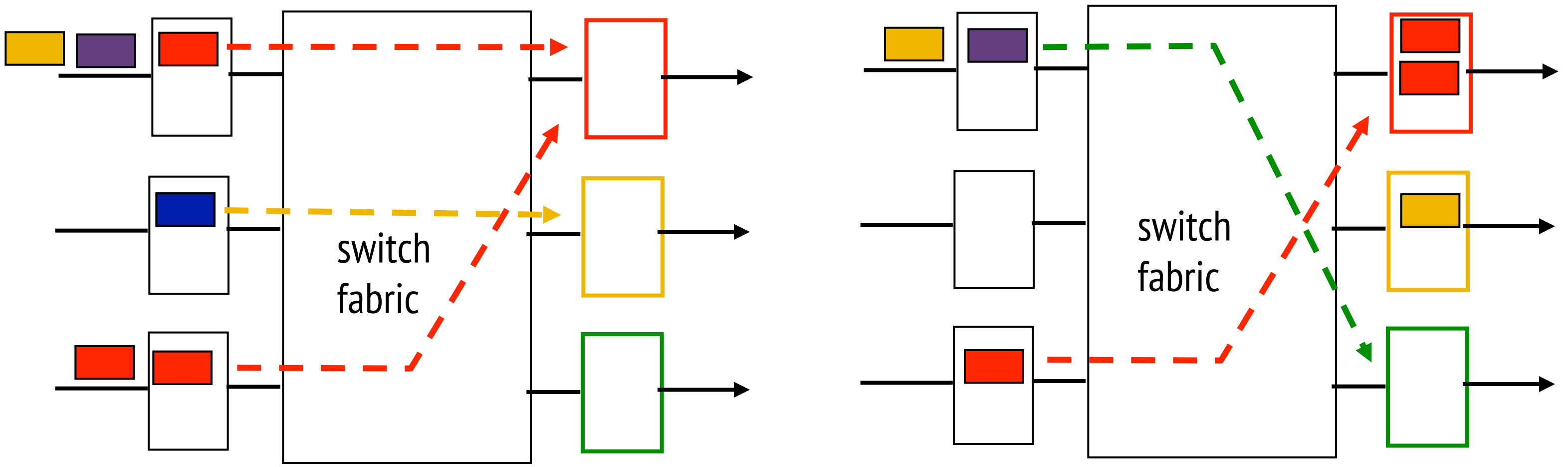


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet experiences HOL blocking

Output port queueing

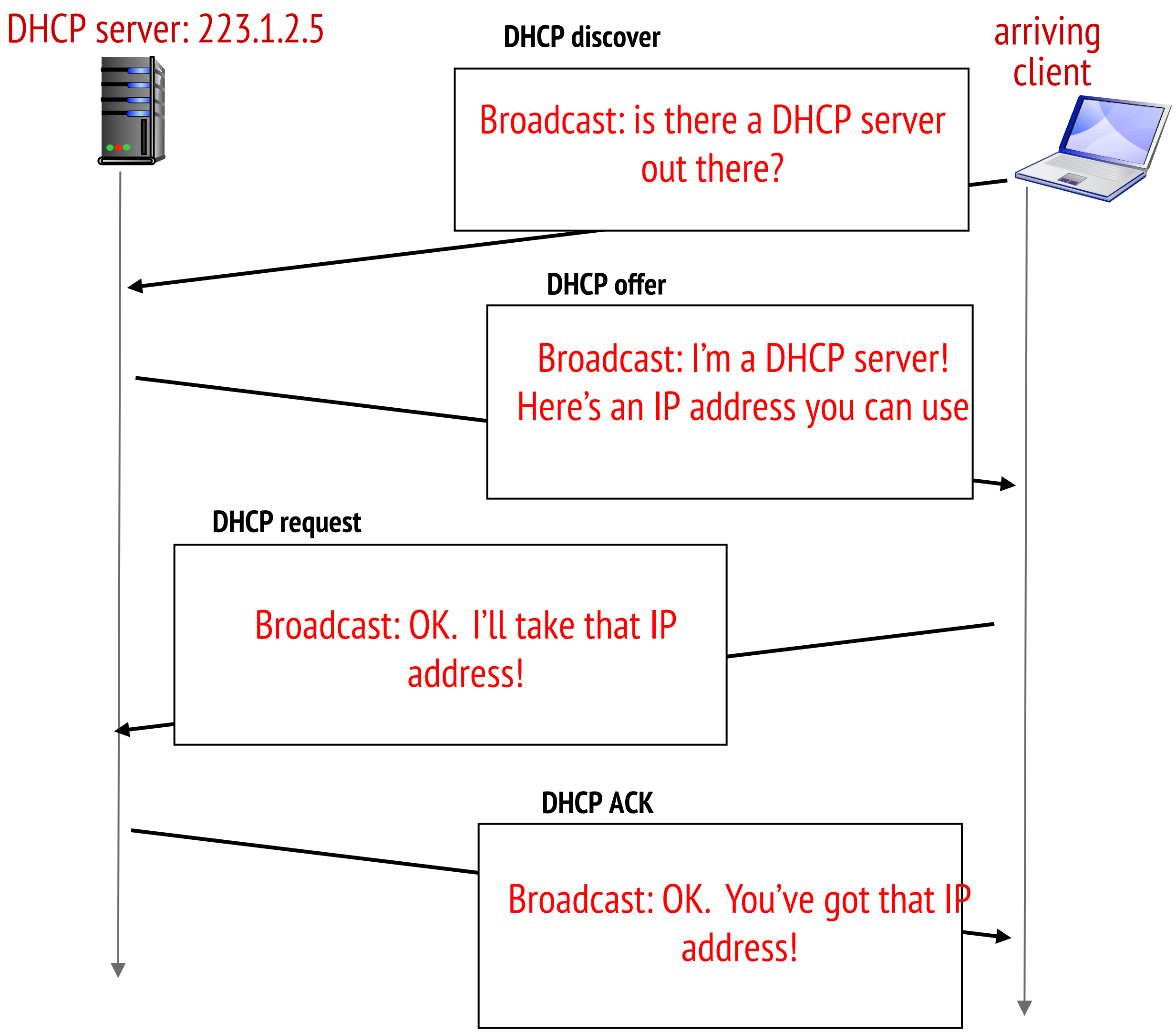


at *t*, packets more from input to output

one packet time later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

DHCP client-server scenario



Internet Control Message Protocol (ICMP)

- Defines a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully
  - Destination host unreachable due to link /node failure
  - Reassembly process failed
  - TTL had reached 0 (so datagrams don't cycle forever)
  - IP header checksum failed

- ICMP-Redirect
  - From router to a source host
  - With a better route information

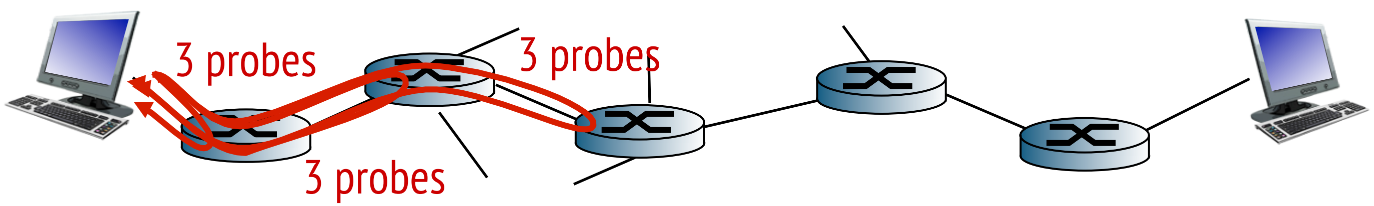
Traceroute and ICMP

- source sends series of UDP segments to destination
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when datagram in *n*th set arrives to nth router:
  - router discards datagram and sends source ICMP message (type 11, code 0)
  - ICMP message include name of router & IP address

when ICMP message arrives, source records RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops

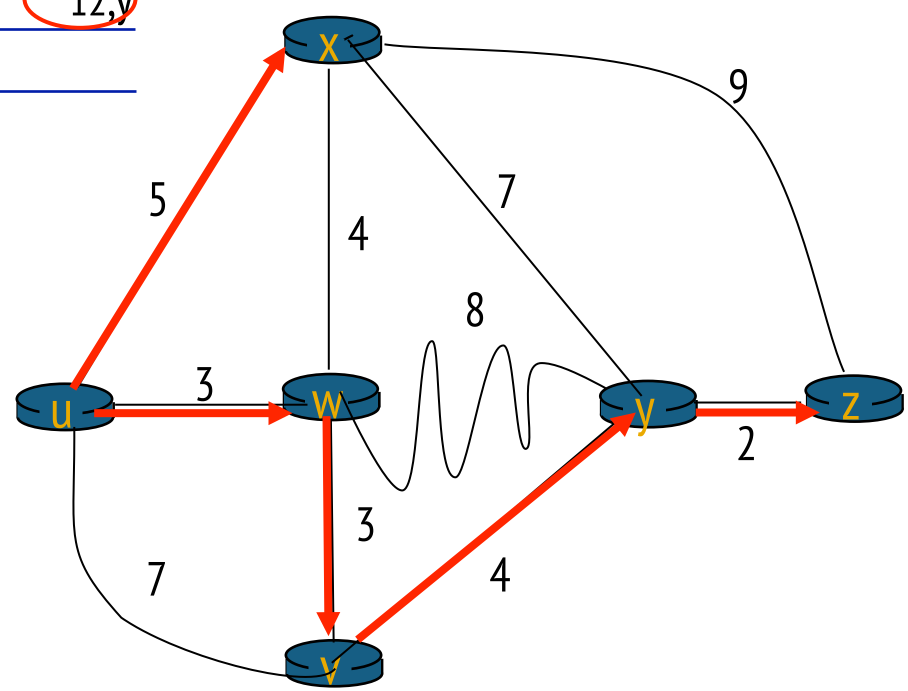


Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ , 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ , 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

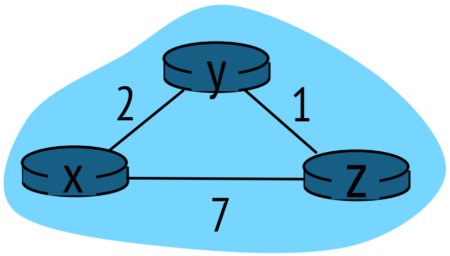
node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

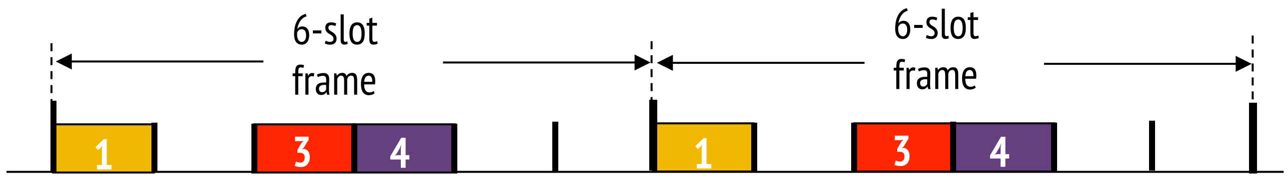


time

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

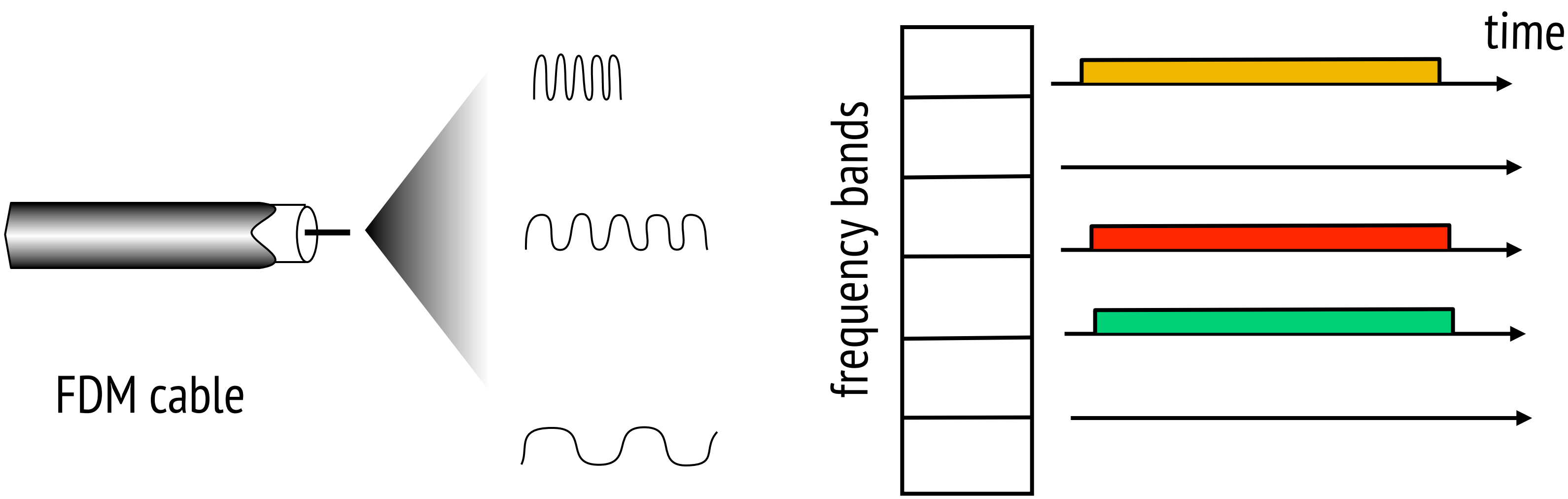
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



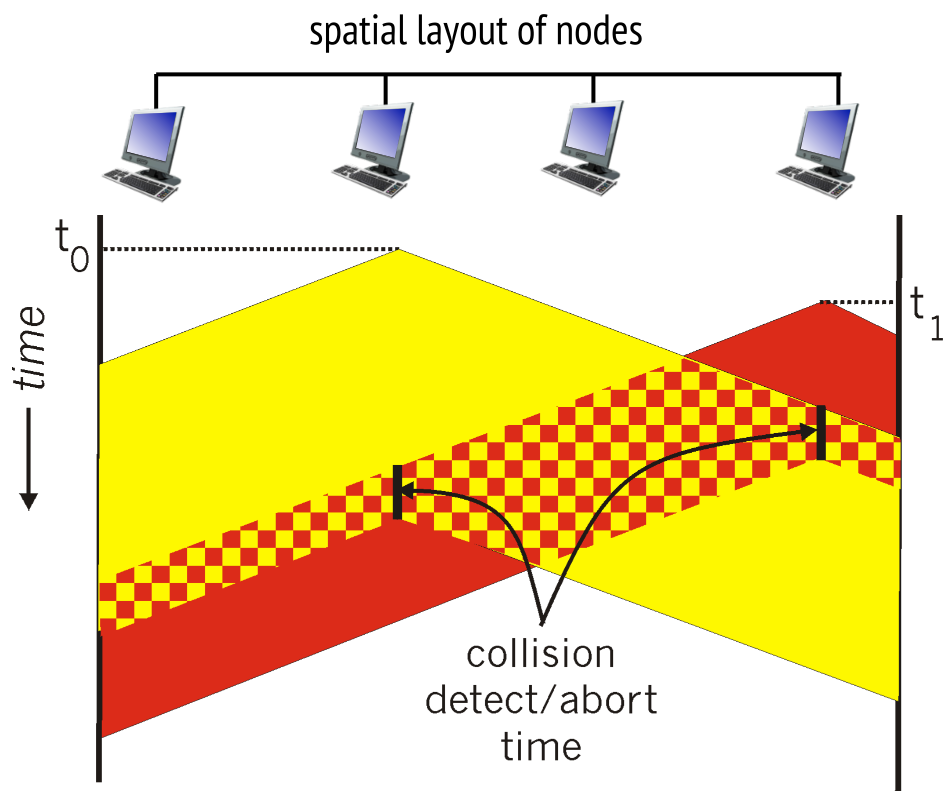
Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have packet to send, frequency bands 2,5,6 idle



CSMA/CD (collision detection)

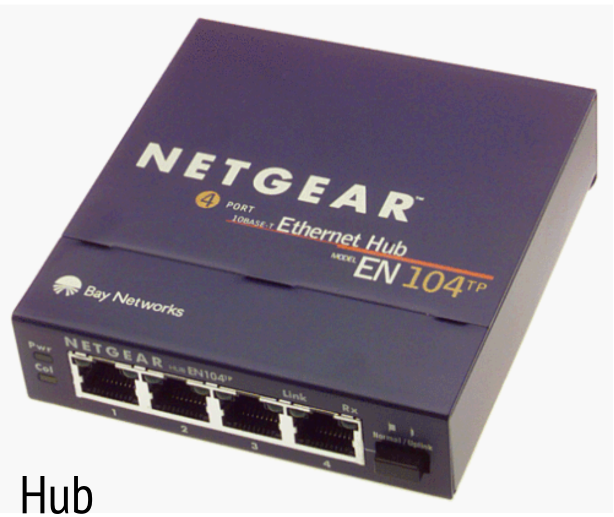


Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters *binary (exponential) backoff:*
   - after *m*th collision, NIC chooses *K* at random from $\{0,1,2, \ldots, 2^m-1\}$. NIC waits K·512 bit times, returns to Step 2
   - longer backoff interval with more collisions

Popular Interconnection Devices

	Hub	Switch	Router
Traffic Isolation	No	Yes	Yes
Plug and Play	Yes	Yes	No
Optimal Routing	No	No	Yes



Hub

Switch

Router

Maximum Data Rate of a Channel

- Nyquist's theorem (1924) relates the data rate to the bandwidth (B) and number of signal levels (V):

  $$\boxed{\text{Max. data rate} = 2B \log_2 V \text{ bits/sec}}$$

- Shannon's theorem (1948) relates the data rate to the bandwidth (B) and signal strength (S) relative to the noise (N):

  $$\boxed{\text{Max. data rate} = B \log_2(1 + S/N) \text{ bits/sec}}$$

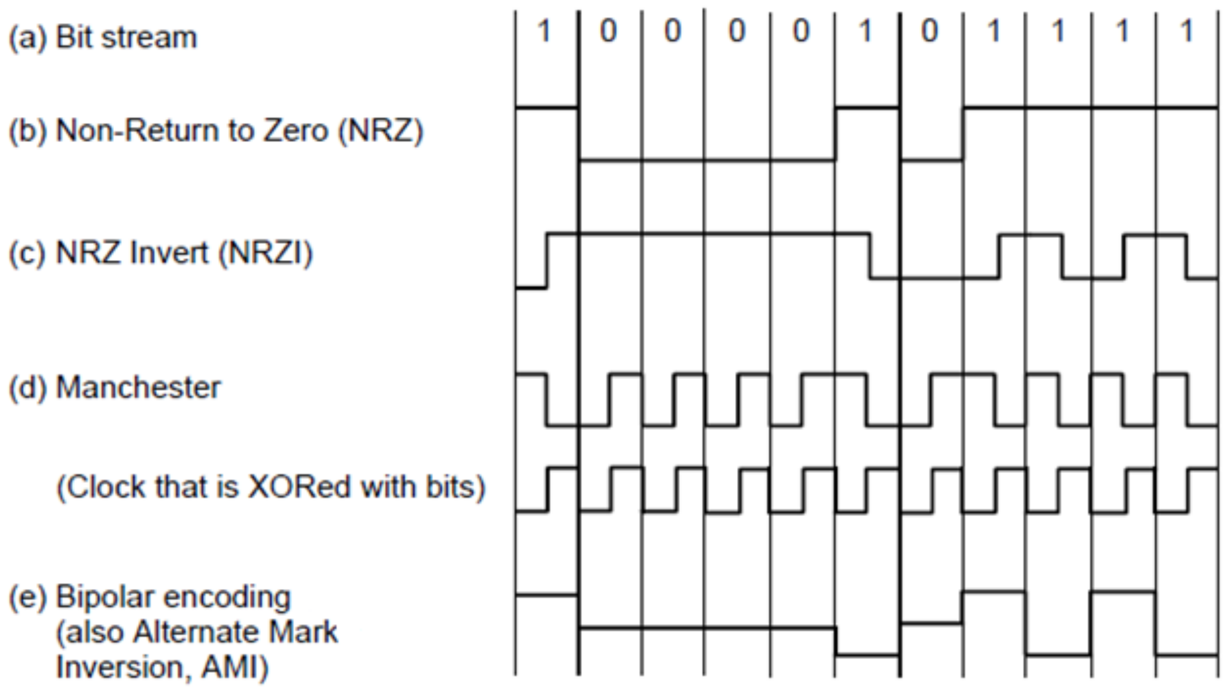
- Signal to Noise Ratio:

  $$\boxed{\text{SNR} = 10 \log_{10}(S/N) \text{ dB}}$$

  dB = decibels ➔ deci = 10; 'bel' chosen after Alexander Graham Bell

Baseband Transmission

- Line codes send <u>symbols</u> that represent one or more bits
  - NRZ is the simplest, literal line code (+1V="1", -1V="0")
  - Other codes tradeoff bandwidth and signal transitions



Four different line codes

Clock Recovery

- To decode the symbols, signals need sufficient transitions
  - Otherwise long runs of 0s (or 1s) are confusing, e.g.:

1 0 0 0 0 0 0 0 0 0 0 um, 0? er, 0?

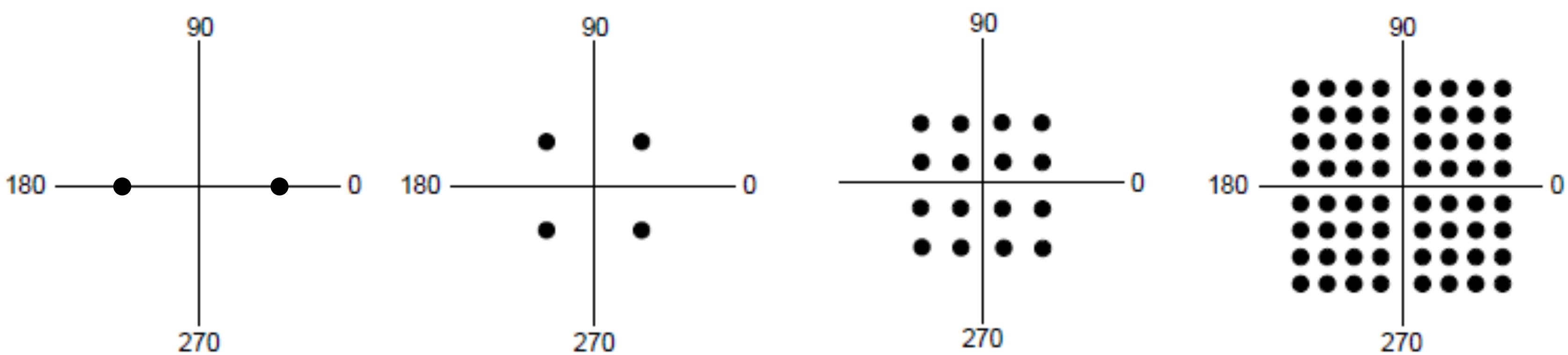
- Strategies:
  - Manchester coding, mixes clock signal in every symbol
  - 4B/5B maps 4 data bits to 5 coded bits with 1s and 0s:

Data	Code	Data	Code	Data	Code	Data	Code
0000	11110	0100	01010	1000	10010	1100	11010
0001	01001	0101	01011	1001	10011	1101	11011
0010	10100	0110	01110	1010	10110	1110	11100
0011	10101	0111	01111	1011	10111	1111	11101

  - Scrambler XORs tx/rx data with pseudorandom bits

Modulation

- *Constellation diagrams* are a shorthand to capture the amplitude and phase modulations of symbols:



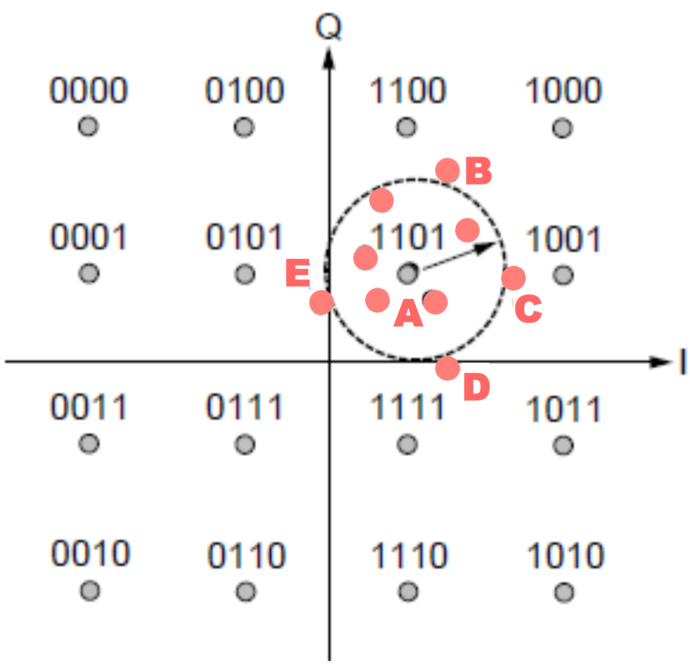
BPSK	QPSK	QAM-16	QAM-64
2 symbols	4 symbols	16 symbols	64 symbols
1 bit/symbol	2 bits/symbol	4 bits/symbol	6 bits/symbol

BPSK/QPSK varies only phase

QAM varies amplitude and phase

Gray Coding

- Gray-coding assigns bits to symbols so that small symbol errors cause few bit errors:

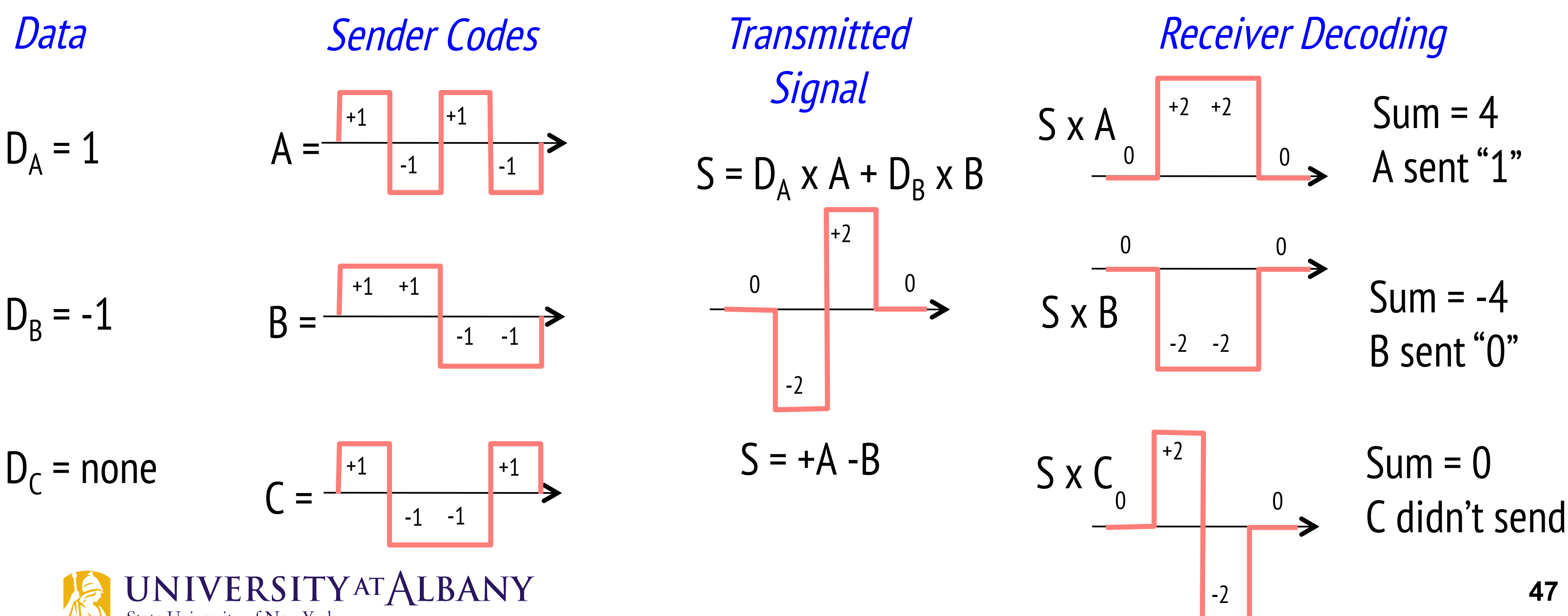


When 1101 is sent:

Point	Decodes as	Bit errors
A	1101	0
B	1100	1
C	1001	1
D	1111	1
E	0101	1

Code Division Multiple Access (CDMA)

- CDMA shares the channel by giving users a code
  - Codes are orthogonal; can be sent at the same time
  - Widely used as part of 3G networks
  - Gold code (GPS Signals), Walsh-Hadamard code, Zadoff-chu sequence



What is network security?

- *confidentiality:* only sender, intended receiver should "understand" message contents
  - Method – encrypt at sender, decrypt at receiver
  - A protocol that prevents an adversary from understanding the message contents is said to provide *confidentiality*.
  - Concealing the quantity or destination of communication is called *traffic confidentiality*.
- *message integrity:* sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
  - A protocol that detects message tampering provides *data integrity.*
  - The adversary could alternatively transmit an extra copy of your message in a *replay attack*.
  - A protocol that detects message tampering provides *originality.*
  - A protocol that detects delaying tactics provides *timeliness*.

What is network security?

- *authentication:* sender, receiver want to confirm identity of each other
  - A protocol that ensures that you really are talking to whom you think you're talking is said to provide *authentication*.
  - Example: DNS Attack [correct URL gets converted to malicious IP]
- *access and availability*: services must be accessible and available to users
  - A protocol that ensures a degree of access is called *availability.*
  - Denial of Service (DoS) Attack
  - Example: SYN Flood attack (Client not transmitting 3rd message in TCP 3-way handshake, thus consuming server's resource)
  - Example: Ping Flood (attacker transmits ICMP Echo Request packets)

Simple encryption scheme

substitution cipher: substituting one thing for another

- *monoalphabetic* cipher: substitute one letter for another

plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq

e.g.:

Plaintext: bob. i love you. alice

ciphertext: nkn. s gktc wky. mgsbc

Encryption key: mapping from set of 26 letters to set of 26 letters

Polyalphabetic Cipher

Plaintext letter:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
$C_1(k = 5)$:	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
$C_2(k = 19)$:	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s

- n substitution ciphers, $C_1, C_2, \ldots, C_n$
- cycling pattern:
  - e.g., n=4 [C_1-C_4], k=key length=5: C_1,C_3,C_4,C_3,C_2; C_1,C_3,C_4,C_3,C_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from C_1, o from C_3, g from C_4

  Encryption key: n substitution ciphers, and cyclic pattern

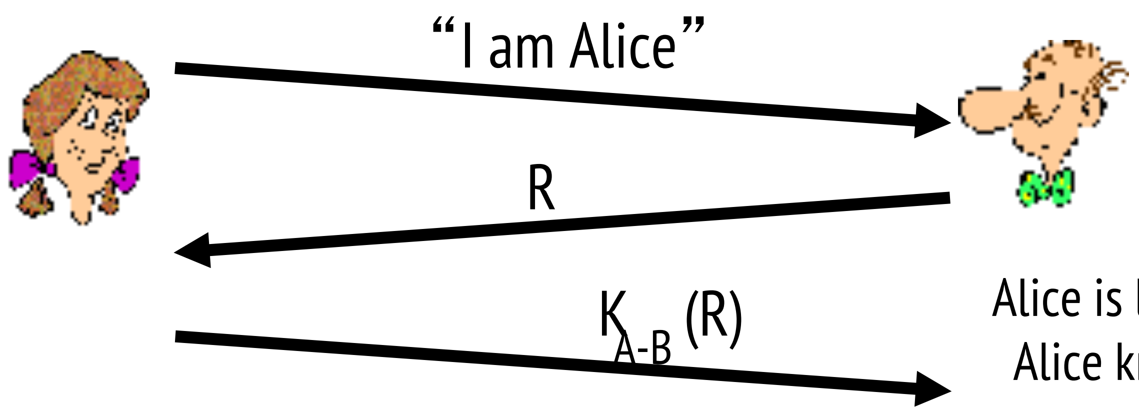
  - key need not be just n-bit pattern

Authentication: yet another try

Goal: avoid playback attack

nonce: number (R) used only *once-in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

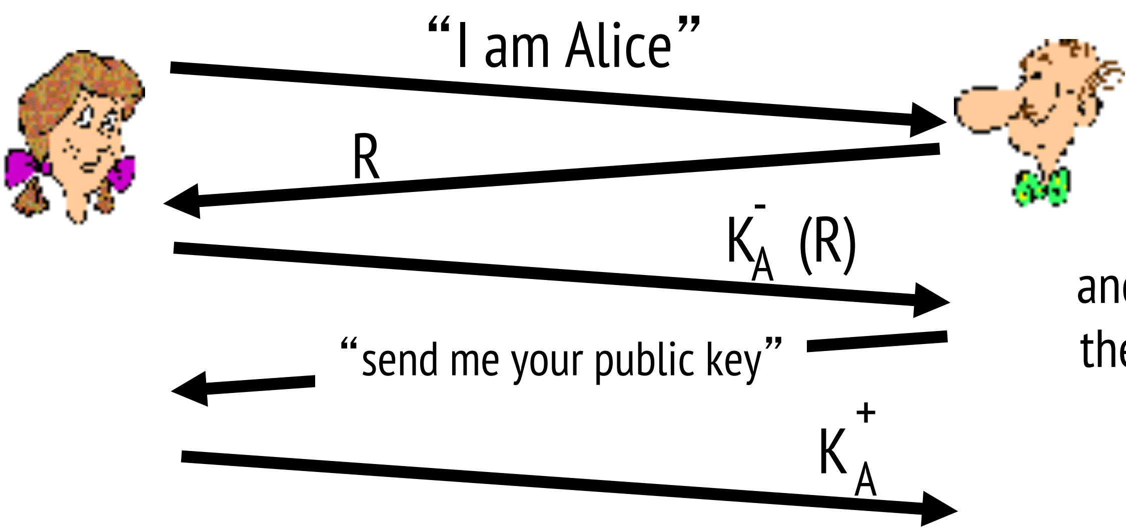
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



Bob computes

$$K_A^+(K_A^-(R)) = R$$

and knows only Alice could have the private key, that encrypted R such that

$$K_A^+(K_A^-(R)) = R$$

Firewalls: why do we need it?

- prevent denial of service attacks:
  - SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections
- prevent illegal modification/access of internal data
  - e.g., attacker replaces CIA’s homepage with something else
- allow only authorized access to inside network
  - set of authenticated users/hosts
- three types of firewalls:
  - stateless packet filters
  - stateful packet filters
  - application gateways

Stateless packet filtering: more examples

Policy	*Firewall Setting*
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Good Luck!!!

UNIVERSITY AT ALBANY
State University of New York

Please provide your feedback in online course evaluation.