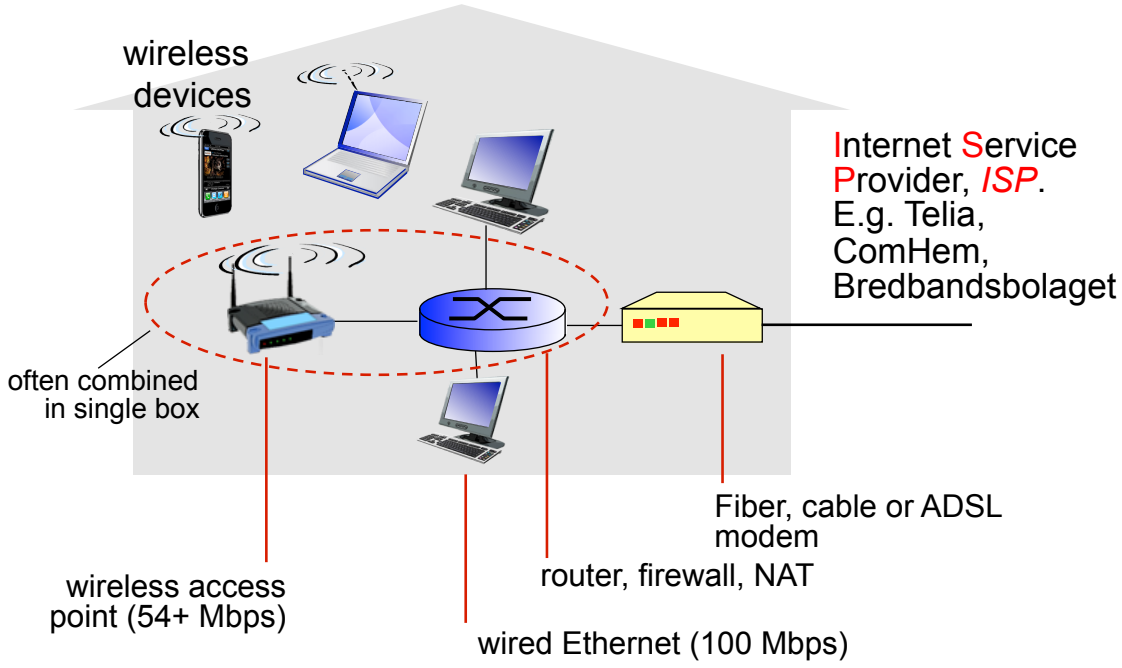
IDT066
Distributed Information Systems

Course Review
Important Slides

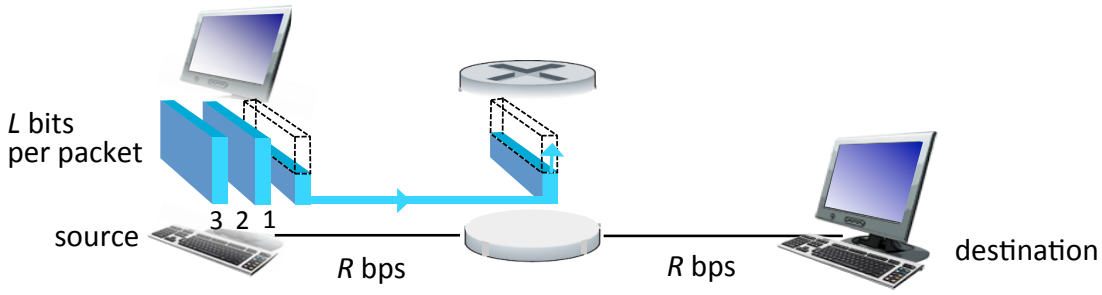
IDT066
Distributed Information Systems

Chapter 1
Introduction

Access net: home network



Packet-switching: store-and-forward



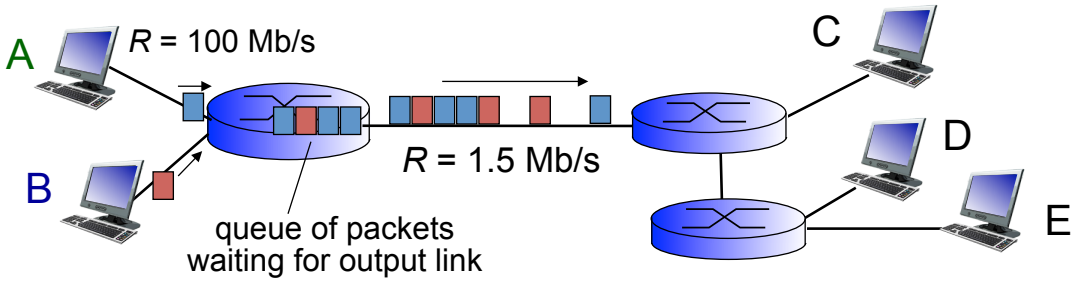
- takes L/R seconds to transmit (push out) L-bit packet into link at R bps
- store and forward: entire packet must arrive at router before it can be transmitted on next link
- end-to-end (E2E) delay = 2L/R (assuming zero propagation delay)

one-hop numerical example:

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

more on delay shortly …

Packet Switching: queueing delay, loss



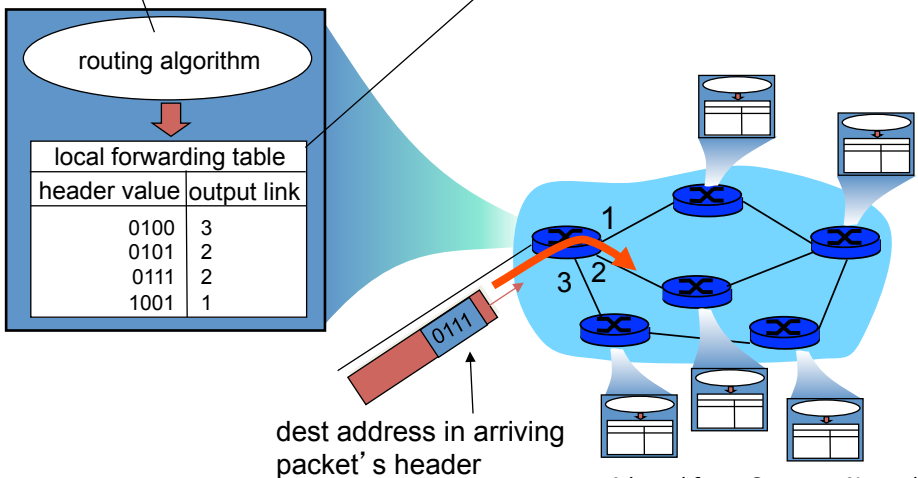
queuing and loss:

- 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

Two key network-core functions

routing: determines source-destination route taken by packets
- routing algorithms

forwarding: move packets from router's input to appropriate router output

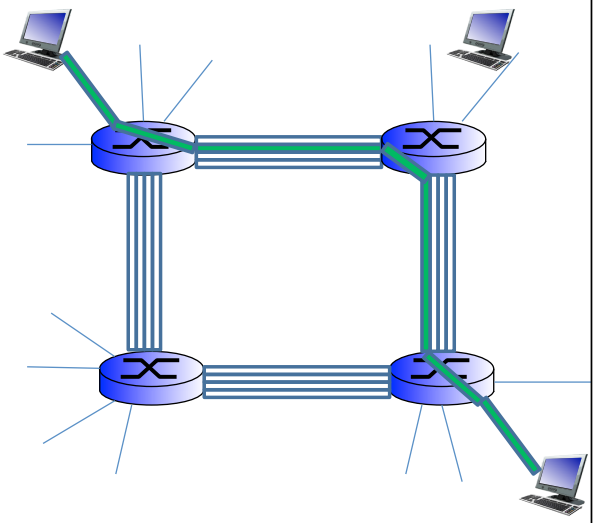


Adapted from: Computer Networking, Kurose/Ross

Circuit switching

end-end resources allocated to, reserved for "call" between source & dest:

- In diagram, each link has four circuits.
  - call gets 2nd circuit in top link and 1st circuit in right link.
- dedicated resources: no sharing
  - circuit-like (guaranteed) performance
- circuit segment idle if not used by call (no sharing)
- Commonly used in traditional telephone networks

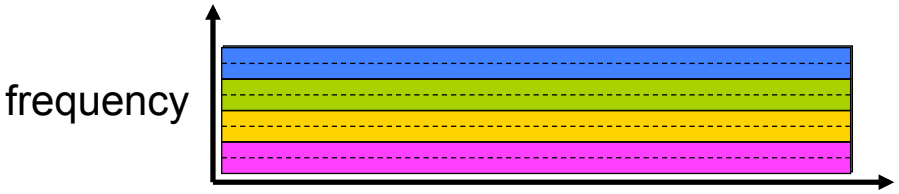


Circuit switching: FDM versus TDM

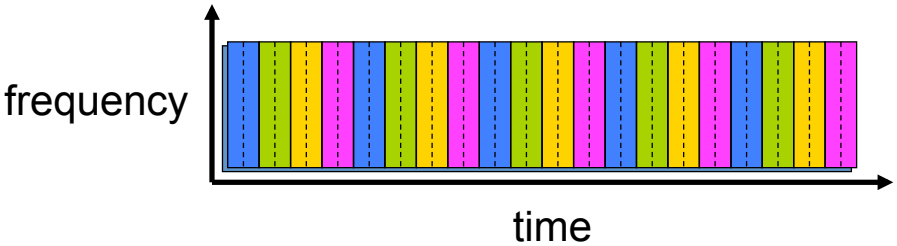
Frequence Division Multiplexing

Example:

4 users



Time Division Multiplexing

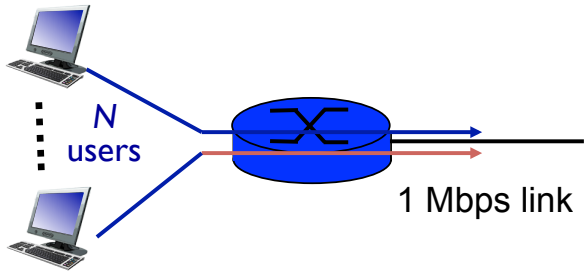


Packet switching versus circuit switching

packet switching allows more users to use network!

example:

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time



- circuit-switching:
  - 10 users
- packet switching:
  - with 35 users, probability > 10 active at same time is less than .0004 *

Q: how did we get value 0.0004?

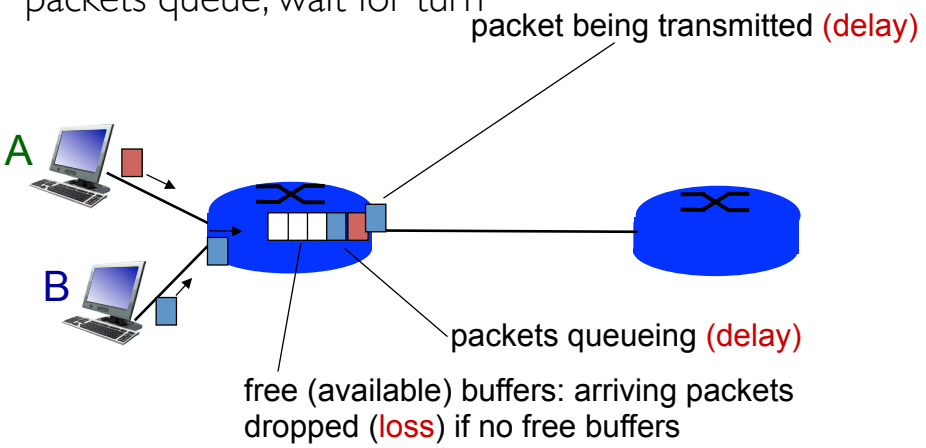
Q: what happens if > 35 users ?

* Check out the online interactive exercises for more examples

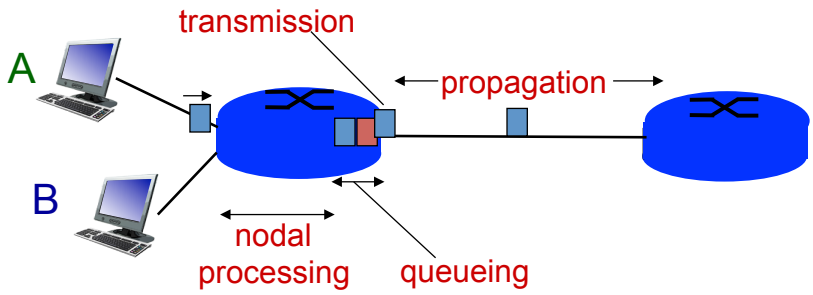
How do loss and delay occur?

packets queue in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn



Four sources of packet delay



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

d_{proc}: nodal processing

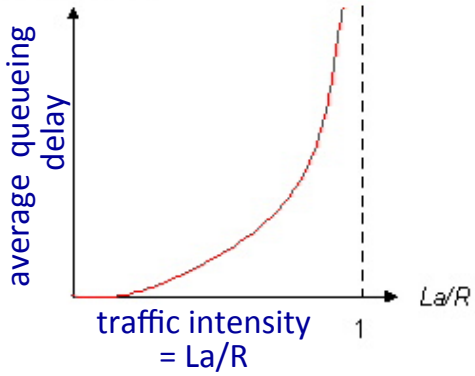
- check bit errors
- determine output link
- typically < msec

d_{queue}: queueing delay

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

Queueing delay (revisited)

- R: link bandwidth (bps)
- L: packet length (bits)
- a: average packet arrival rate



- La/R ~ 0: avg. queueing delay small
- La/R -> 1: avg. queueing delay large
- La/R > 1: more "work" arriving than can be serviced, average delay infinite!

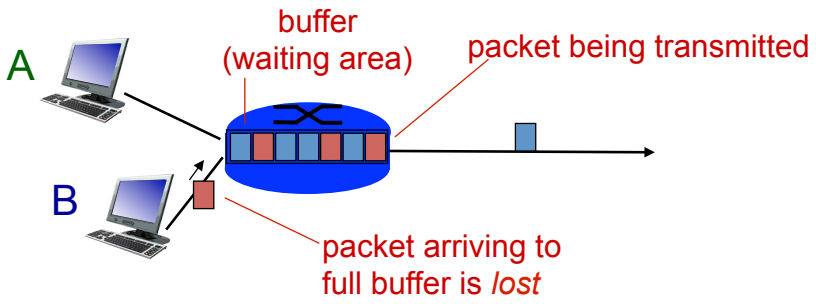
La/R ~ 0

La/R -> 1

* Check out the Java applet for an interactive animation on queuing and loss

Packet loss

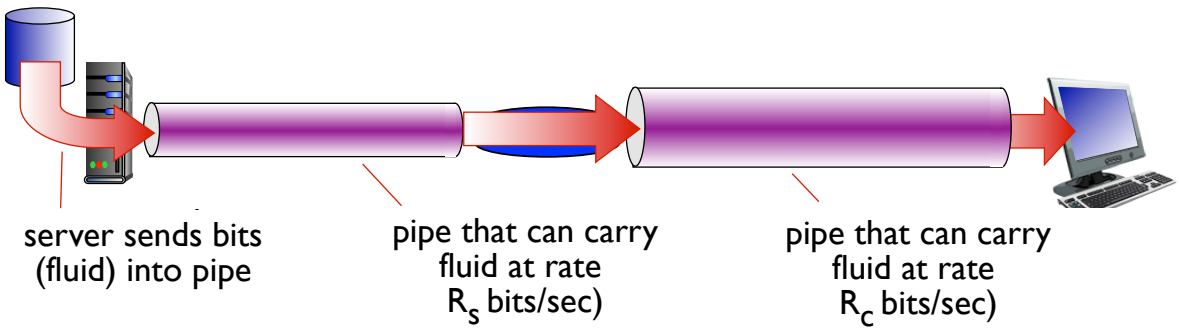
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



* Check out the Java applet for an interactive animation on queuing and loss

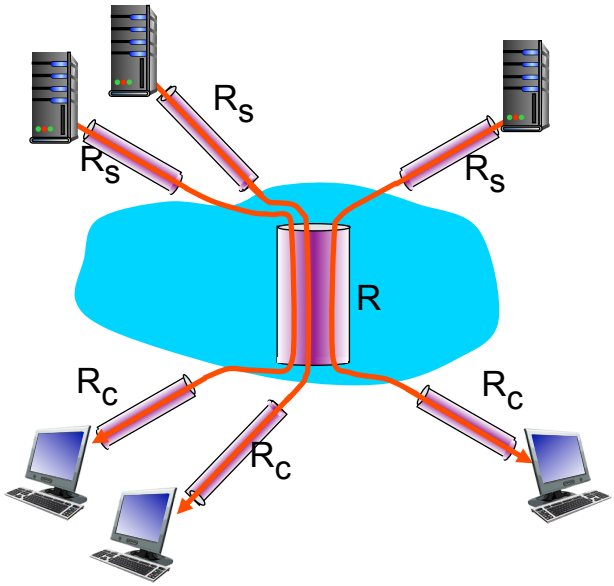
Throughput

- 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



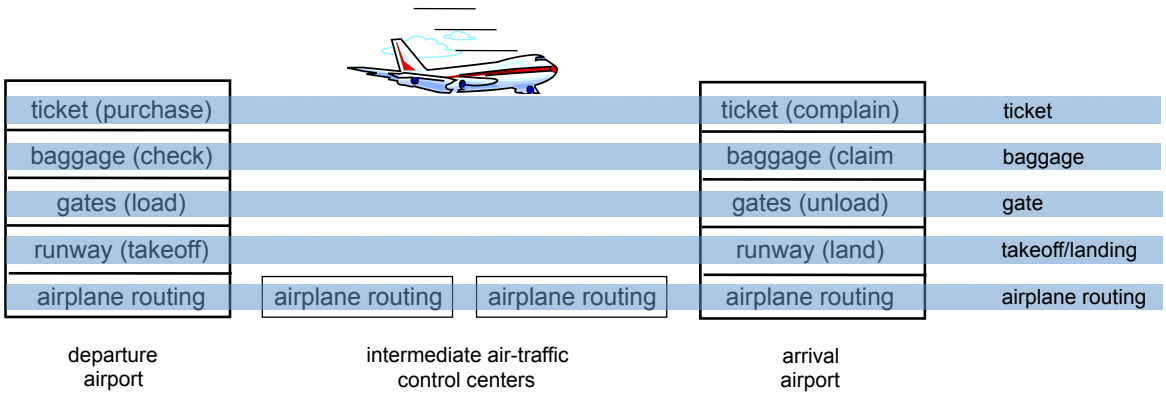
Throughput: Internet scenario

- per-connection end-end throughput: $\min(R_c, R_s, R/10)$
- in practice: R_c or R_s is often bottleneck



10 connections (fairly) share backbone bottleneck link R bits/sec

Layering of airline functionality



layers: each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below

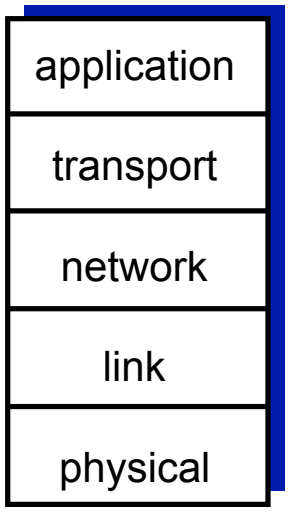
Why layering?

dealing with complex systems:

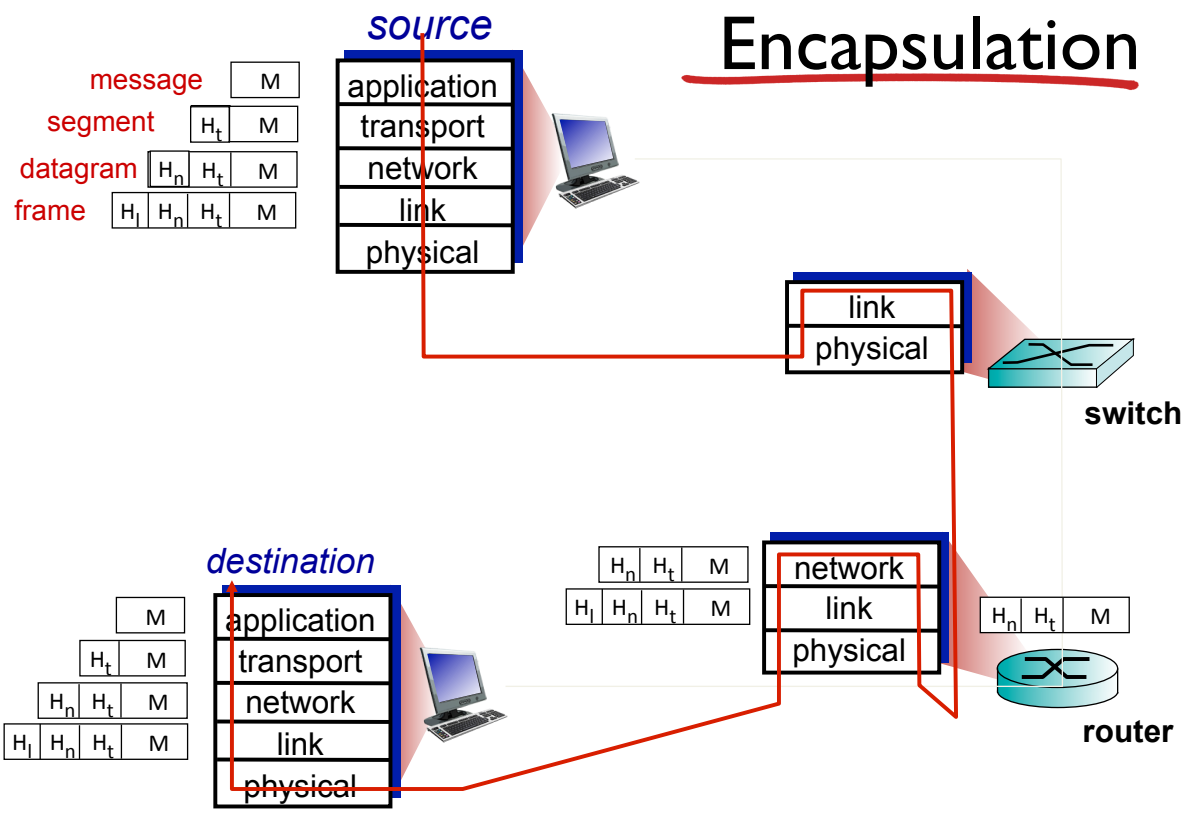
- explicit structure allows identification, relationship of complex system's pieces
  - layered reference model for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest 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, 802.111 (WiFi), PPP
- physical: bits "on the wire"



Encapsulation



UPPSALA UNIVERSITET

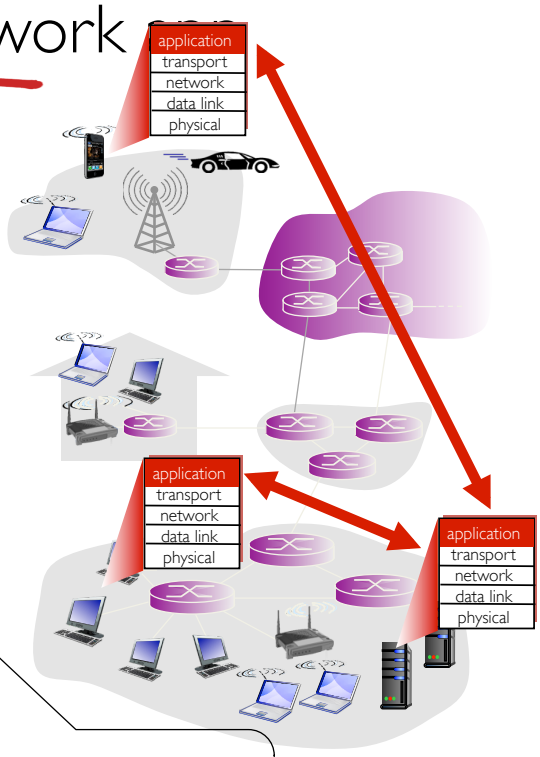
IDT066
Distributed Information Systems

Chapter 2
Applications

Creating a network

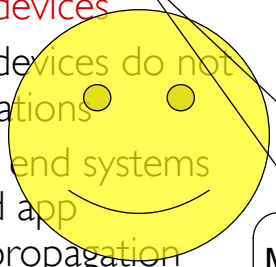
write programs that:

- run on (different) *end systems*
- communicate over networks
- e.g., web server software communicates with browser software



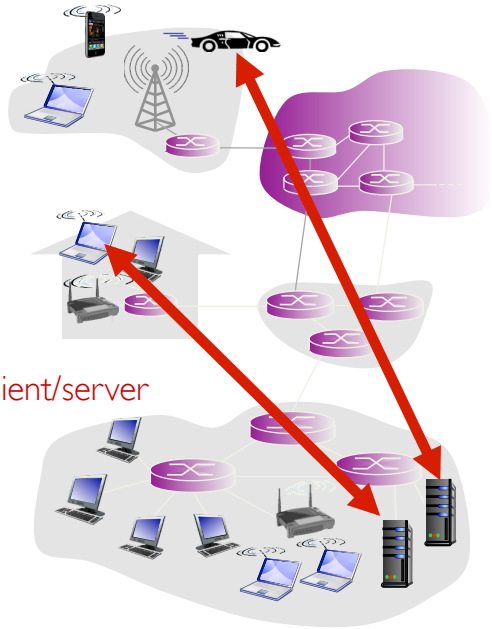
no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation



Main reason for the success of Internet

Client-server architecture



client/server

server:

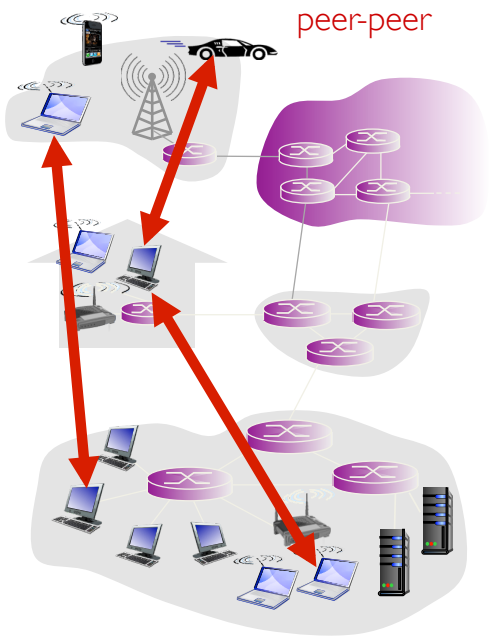
- *always-on* host
- permanent IP address (i.e. "*I know where you are*")
- often at a data center for scaling capacity with demand

client:

- initiates the contact and communicates with server
- may be intermittently connected
- may have dynamic IP addresses. (Why not permanent?)
- clients do not communicate directly with each other

P2P architecture

- end systems (called *peers*) communicate directly
- peers *request* service from other peers and *provide* services in return to the others
  - *self scalability* – new peers bring new service capacity, (as well as new service demands)
- *not* always-on server. May be switched off and may change IP addresses
  - complex management. Why?



Adapted from: Computer Networking, Kurose/Ross, pp 112-113

App-layer protocol defines

- types of messages exchanged,
  - e.g., request, response
- message syntax:
  - different fields in messages, e.g. for application data and control information to other party
- message semantics
  - meaning of information in control fields (not app data)
- rules for when and how processes send & respond to messages

"open"/standard protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

"closed"/proprietary protocols:

- e.g., Skype

Adapted from: Computer Networking, Kurose/Ross, pp 117-122

Transport service requirements: common apps

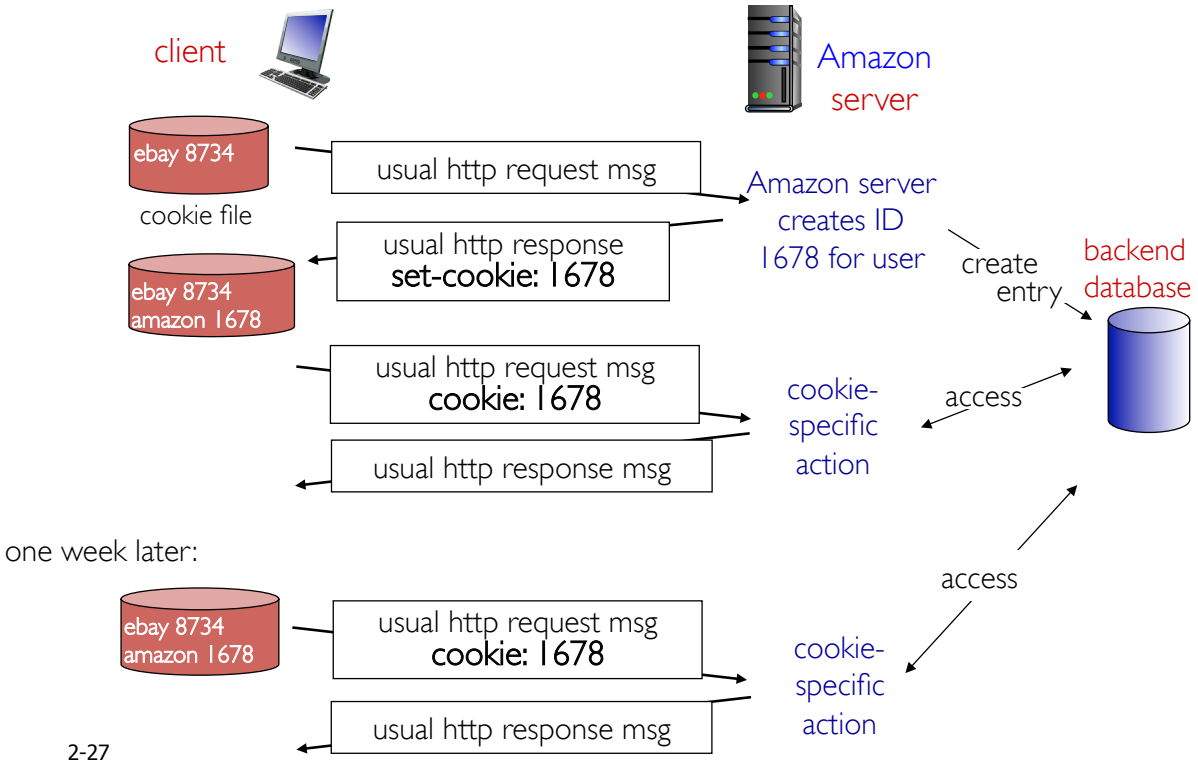
application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
text messaging	no loss	elastic	yes and no

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests

Cookies: keeping "state" between requests



Adapted from: Computer Networking, Kurose/Ross, pp 134-136

DNS: domain name system

Q: how to *map* between IP address and host name, and vice versa ?

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., www.yahoo.com - used by humans

Domain Name System:

- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
  - Note: This is a core Internet function, implemented as application-layer protocol
  - Again - complexity at network's "edge"
- *distributed database* implemented in hierarchy of many *name servers*

Adapted from: Computer Networking, Kurose/Ross, pp 156-158

DNS: services, structure

DNS services

- hostname to IP address translation – demo!
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name. Why and when?
- *why not centralize DNS*

It doesn't scale?

- single point of failure
- traffic volume
- distant centralized database, long delay.
- maintenance

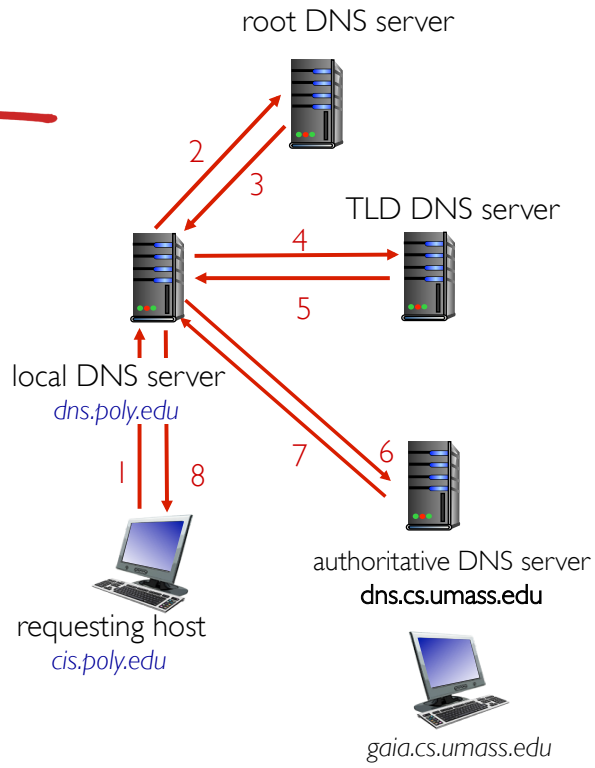
Adapted from: Computer Networking, Kurose/Ross, pp 156-159

DNS name resolution bottom up example

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

iterated query:

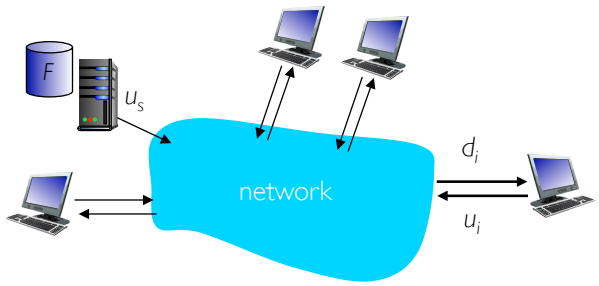
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Adapted from: Computer Networking, Kurose/Ross, pp 160-163

File distribution time: P2P

- *server transmission:* must upload at least one copy
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - min client download time: F/d_{min}



- *clients:* as aggregate must download *NF* bits
  - max upload rate (limting max download rate) is $u_s + \Sigma u_i$

time to distribute F to N clients using P2P approach

$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

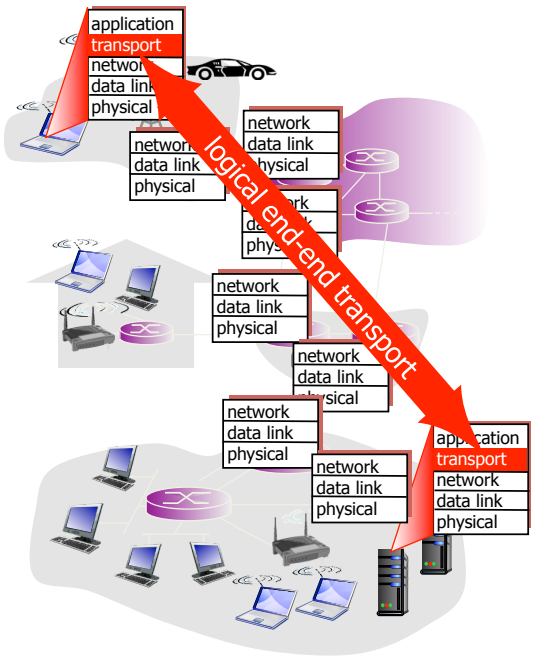
UPPSALA UNIVERSITET

IDT066
Distributed Information Systems

Chapter 3
Transport Layer

Internet transport-layer protocols

- unreliable, unordered delivery: UDP
  - no-frills extension of "best-effort" IP
- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- services not available:
  - delay guarantees
  - bandwidth guarantees



pp 224-230

TCP vs. UDP

Features	TCP (Yes/No)	UDP (Yes/No)
(1) Ordered packets	Yes	No
(2) Connectionless	No	Yes
(3) Reliable data delivery (no packet loss)	Yes	No
(4) Flow control	Yes	No
(5) Congestion control	Yes	No
(6) delay guarantees	No	No
(7) bandwidth guarantees	No	No

UDP: User Datagram Protocol [RFC 768]

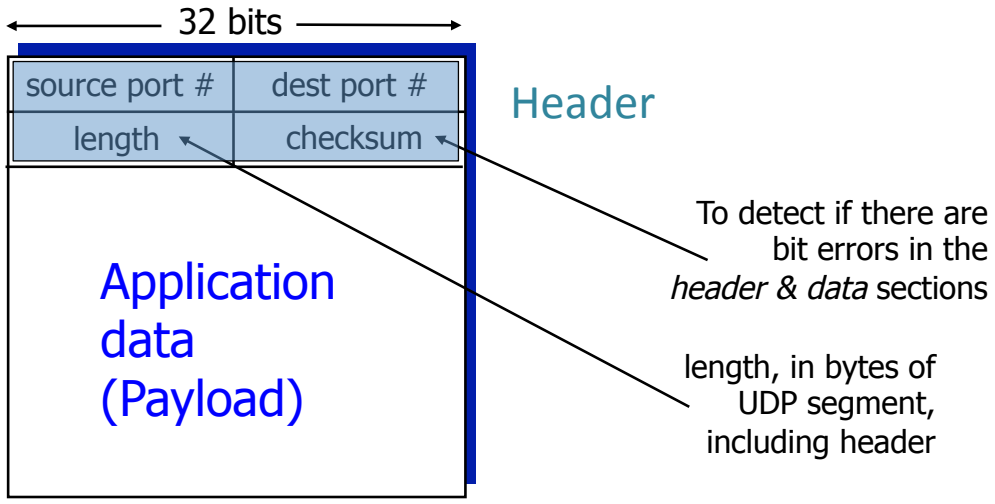
- "no frills," "bare bones" Internet transport protocol
  - each UDP packet (segment) handled independently of others
- *"best effort" service*, UDP segments may be:
  - lost
  - delivered out-of-order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver

- UDP is used by:
  - streaming multimedia apps (loss tolerant, but time sensitive)
  - name look-up services (e.g. the Domain Name Service protocol) Why?
  - apps that *adds* reliability (e.g. remote procedure calls)
  - apps that use specific error recovery (e.g. TV broadcast).

Why is there a UDP

- no connection establishment before sending data (which adds delay)
- simple: no connection state at sender, receiver that has to be maintained.
- no flow control: UDP can blast away segments as fast as desired without waiting for a receiver OK.

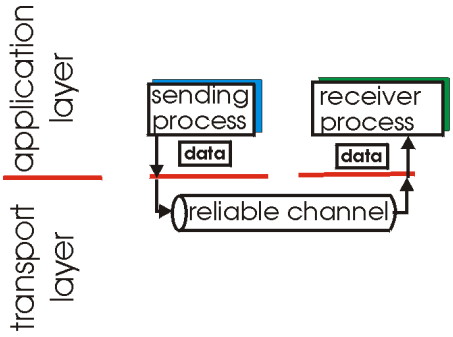
UDP: segment(data packet) header



UDP segment format

Principles of reliable data transfer

- important in application, transport, link layers
  - top-10 list of important networking topics!

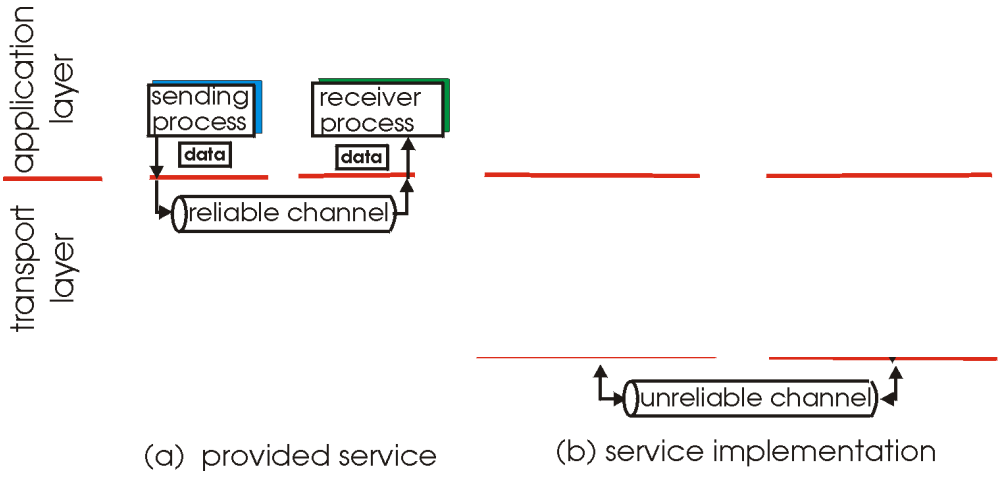


(a) provided service

- *characteristics of unreliable channel* will determine the complexity of a reliable data transfer protocol (rdt)

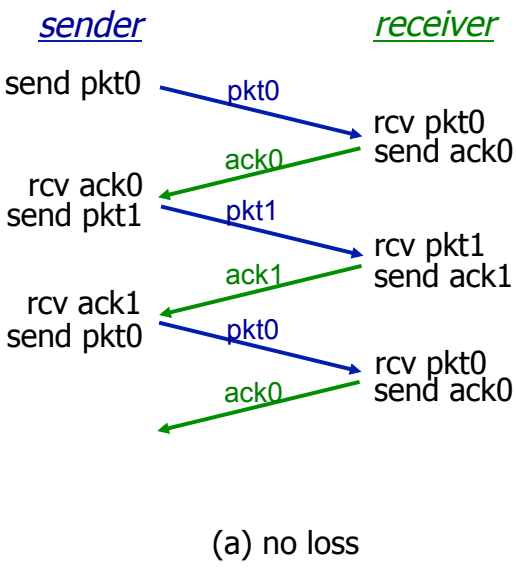
Principles of reliable data transfer

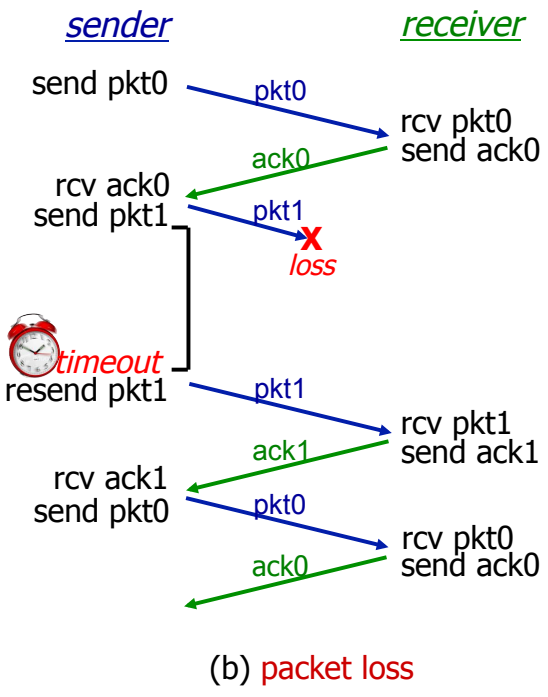
- important in application, transport, link layers
  - top-10 list of important networking topics!



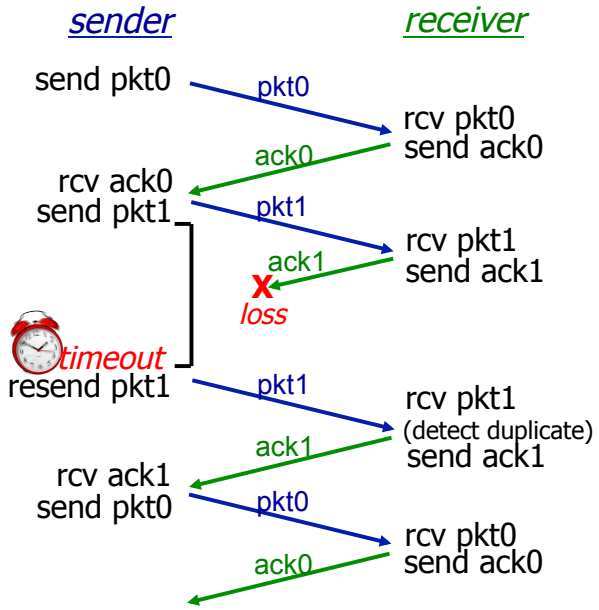
- *characteristics of unreliable channel* will determine the complexity of a reliable data transfer protocol (rdt)

Example of a Reliable Data Transfer Protocol: Packet Loss

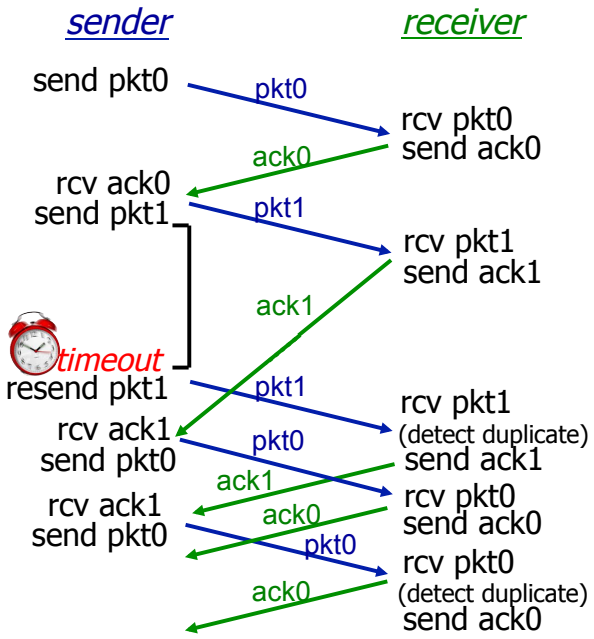




Example of a Reliable Data Transfer Protocol: Ack loss

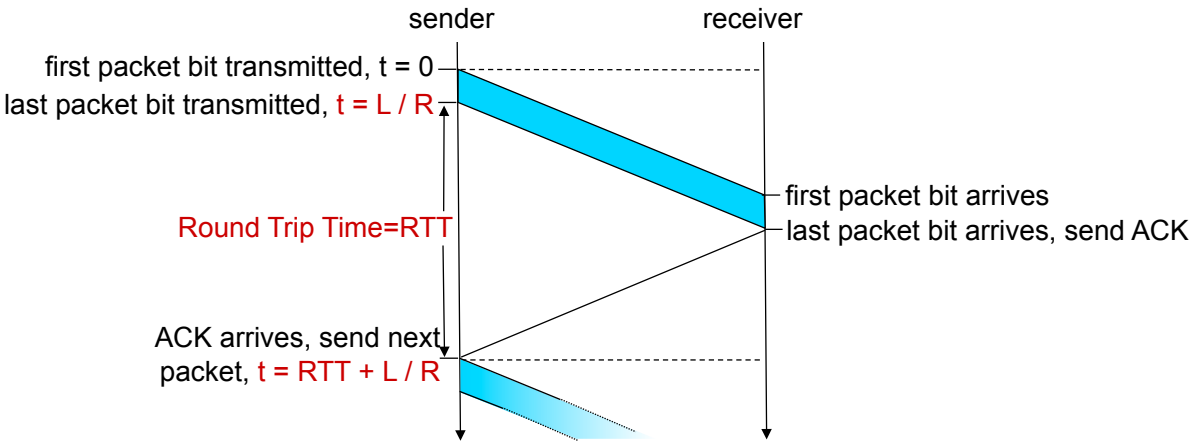


(c) ACK loss



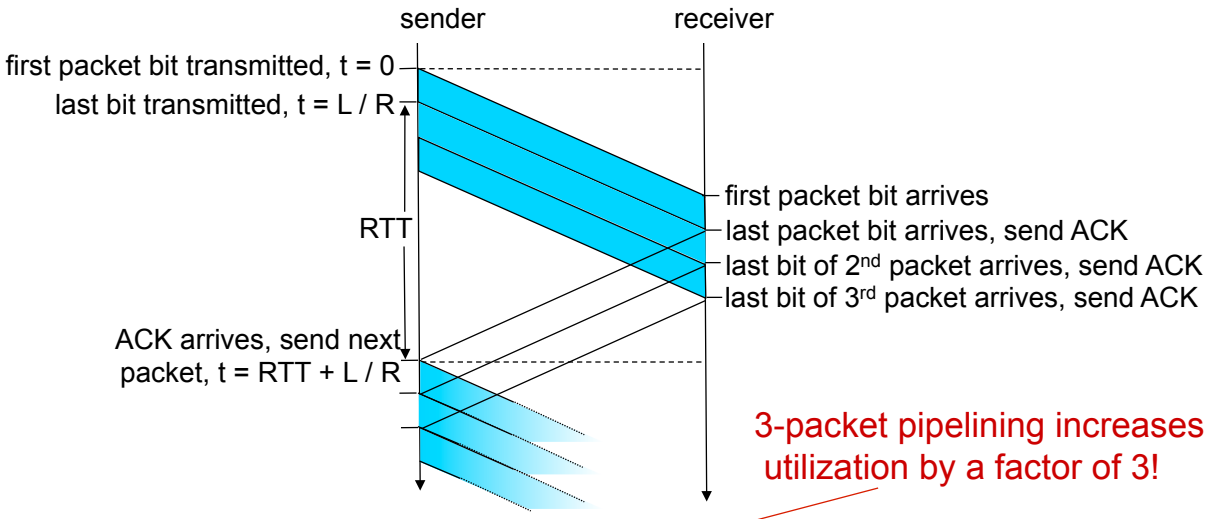
(d) premature timeout/ delayed ACK

stop-and-wait operation



$$U_{sender} = \frac{L / R}{RTT + L / R}$$

Pipelining: increased utilization



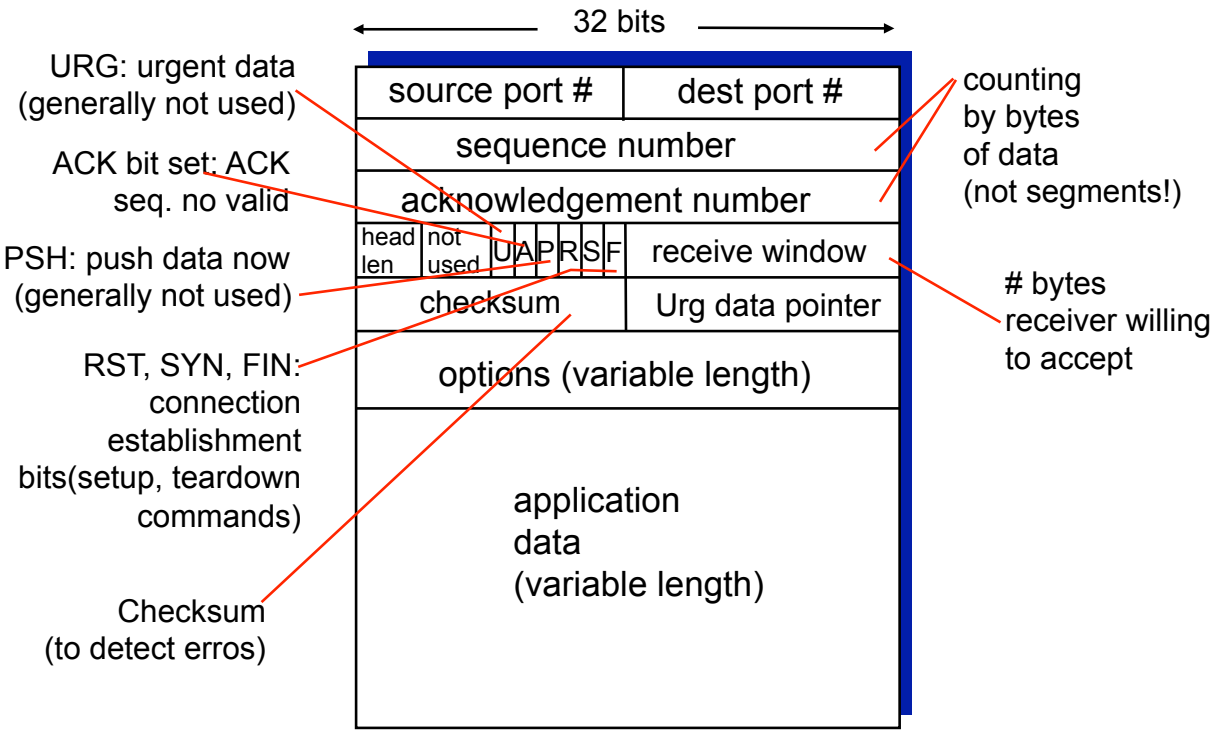
3-packet pipelining increases utilization by a factor of 3!

$$U_{sender} = \frac{3L / R}{RTT + L / R}$$

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

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - 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 (exchange of control msgs) inits sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

TCP segment structure



TCP seq. numbers, ACKs

sequence numbers:

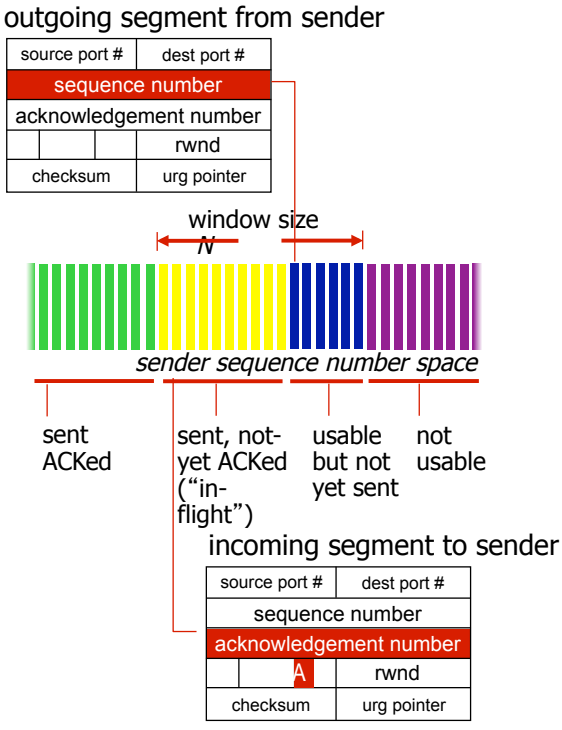
- byte stream "number" of first byte in segment's data

acknowledgements:

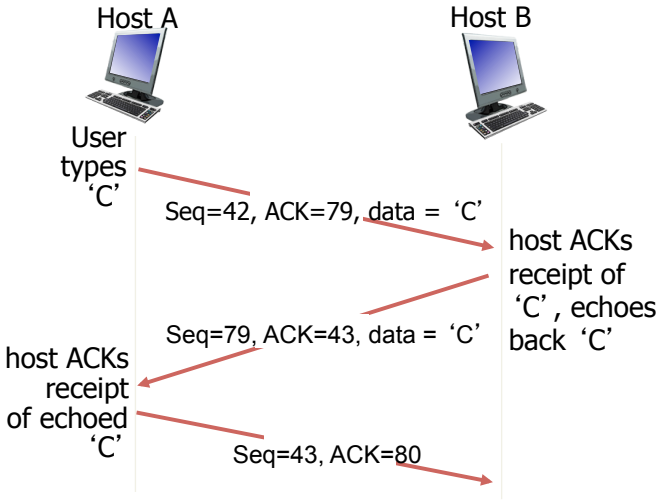
- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

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

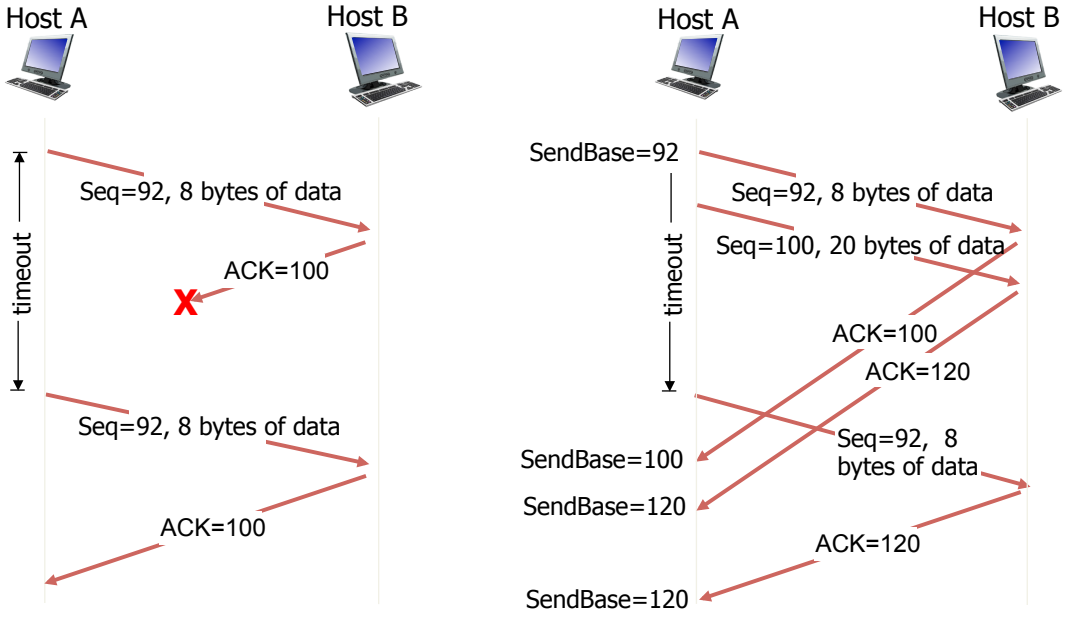


TCP seq. numbers, ACKs



simple telnet scenario

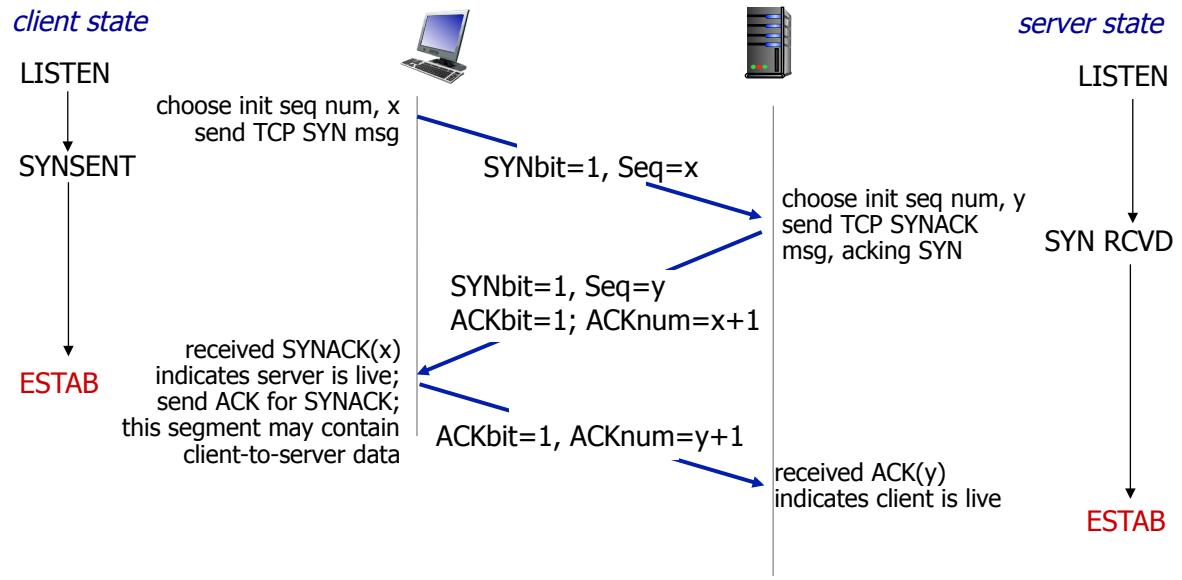
TCP: retransmission scenarios



lost ACK scenario

premature timeout

TCP 3-way handshake



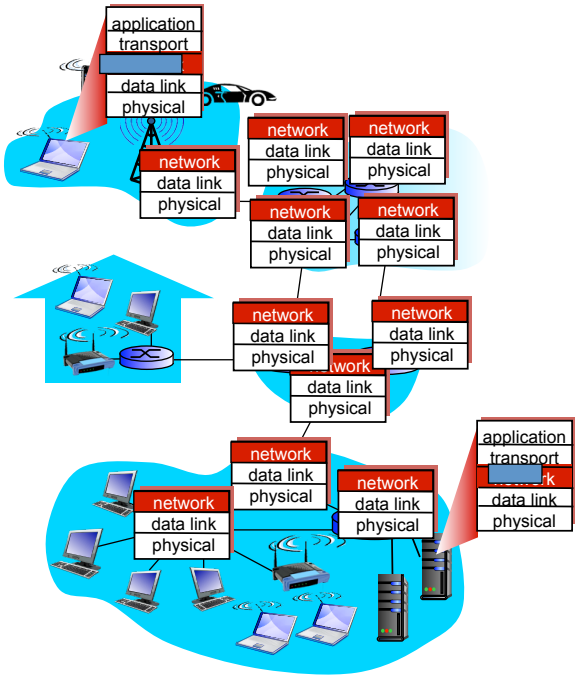
UPPSALA UNIVERSITET

IDT066
Distributed Information Systems

Chapter 4
Network Layer

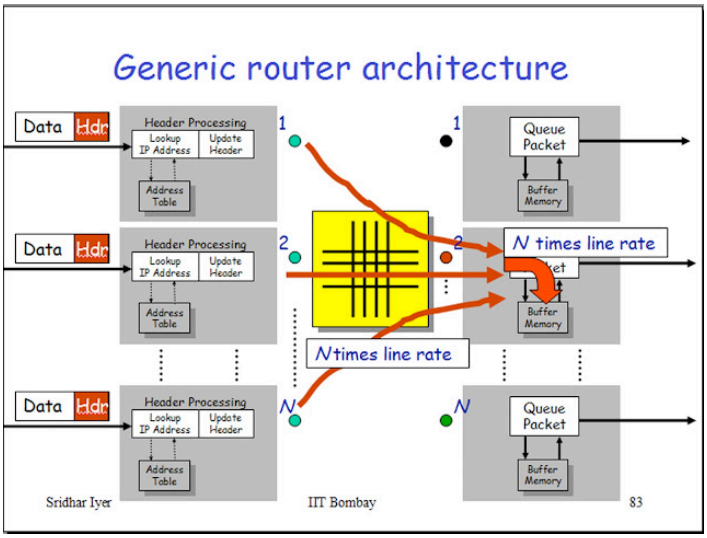
Network layer

- forwards packets from sending to receiving host
- on sending side: encapsulates transport packets into *datagrams*
- on receiving side: delivers packets to transport layer
- network layer protocols exist in ***every*** host & router
- router examines header fields in all *datagrams*

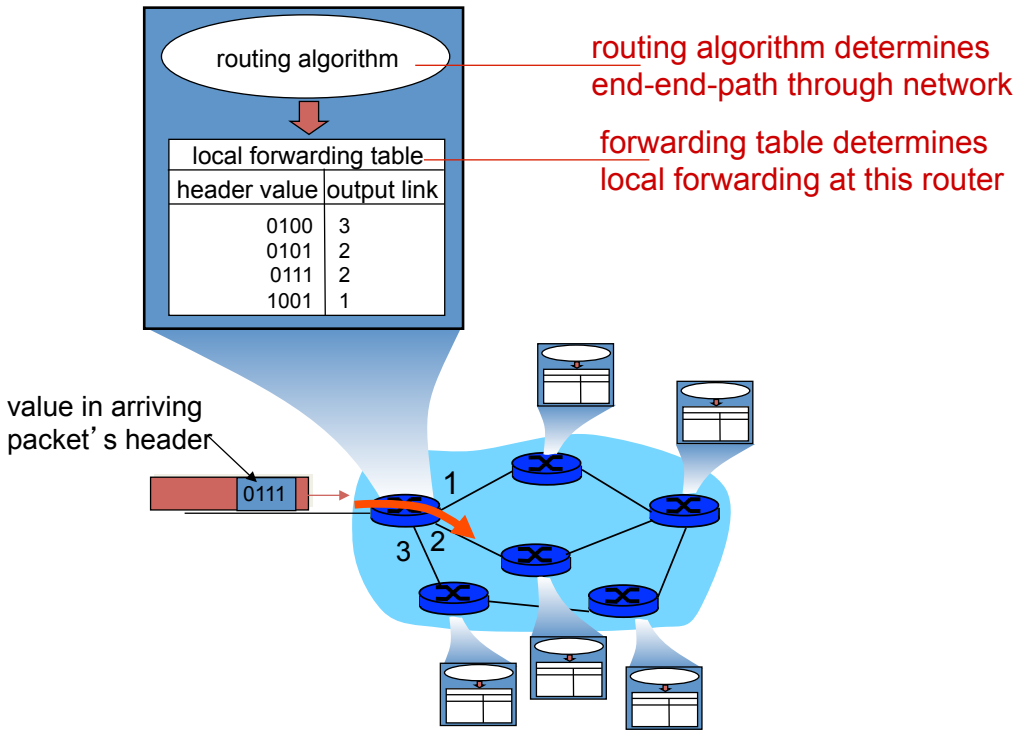


Two key network-layer functions

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.
  - *routing algorithms*



Interplay between routing and forwarding

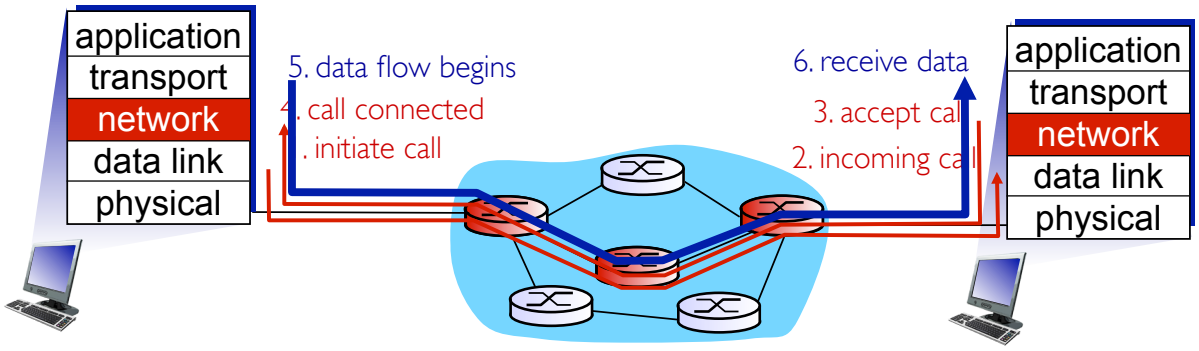


Connection vs connection-less services

- *datagram* network provides network-layer *connectionless* service
- *virtual-circuit* network provides network-layer *connection* service

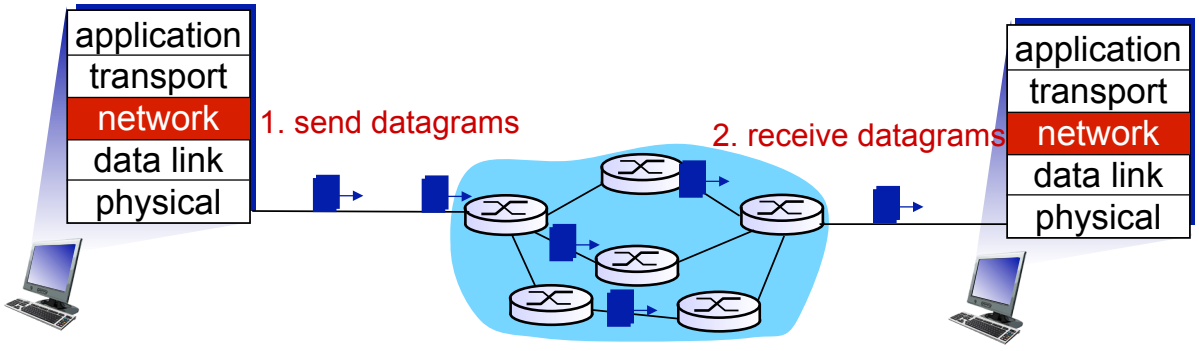
Virtual circuits: Signaling and flow

- Signaling to **setup** a virtual circuit, *reserve resources,* e.g. line capacity and buffers at each router. Establish **state.**
- The flow of data packets starts.
- Signaling to supervise flow (e.g. Route/link failure)
- Signaling to **tear down** circuit & release resources

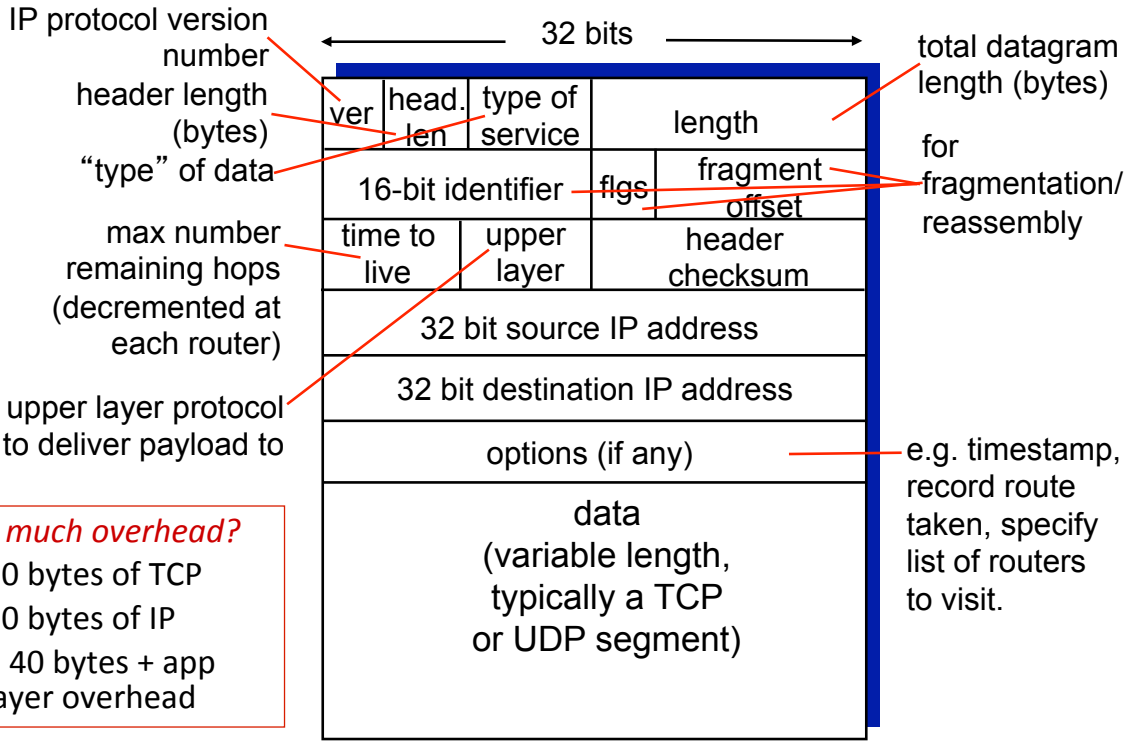


Datagram networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address, looked up at all encountered routers.



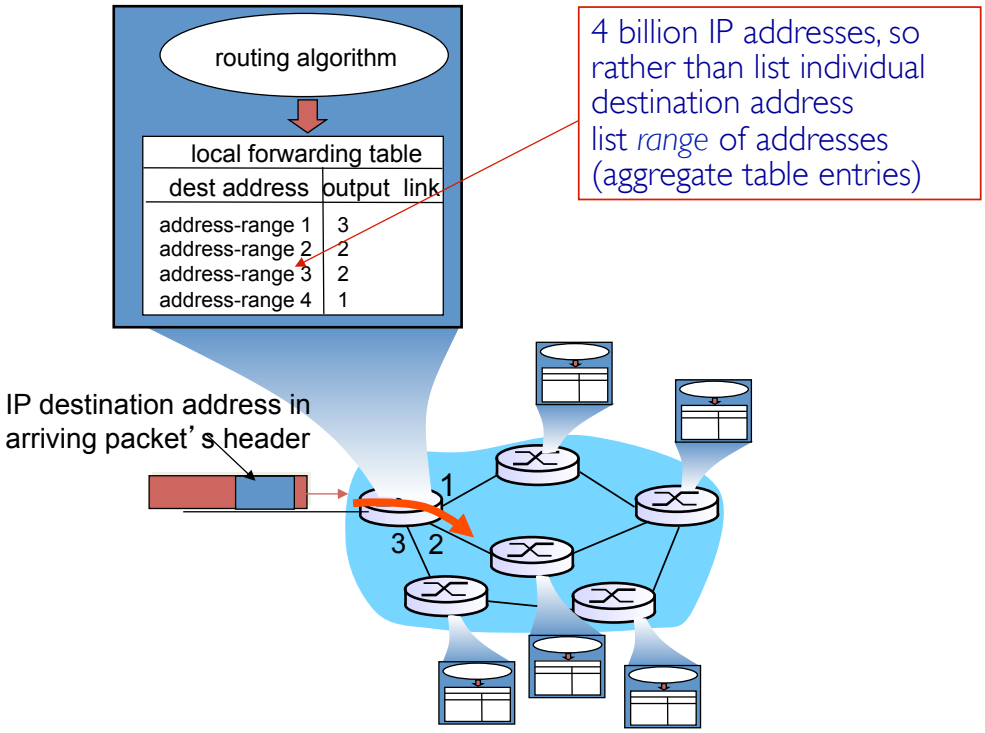
IP datagram format



how much overhead?

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

Datagram forwarding table



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

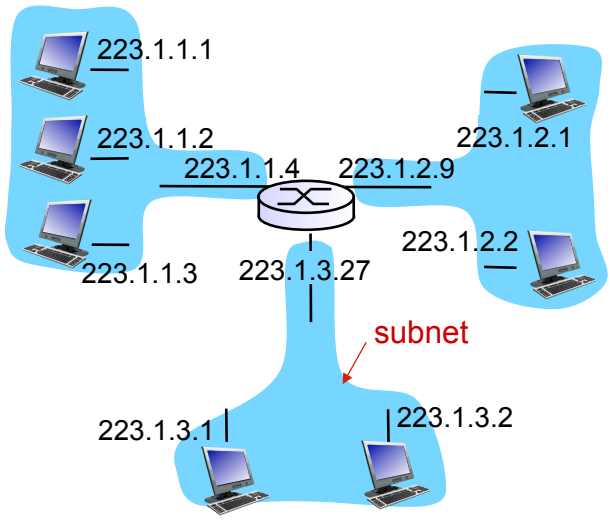
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

Subnets

- IP address:
  - –subnet part - high order bits
  - –host part - low order bits
- *what's a subnet ?*
  - –device interfaces with same subnet part of IP address
  - –can physically reach each other *without intervening router*



network consisting of 3 subnets

subnet part | host part

11011111 00000001 00000011 00000000

223.1.3.0/24

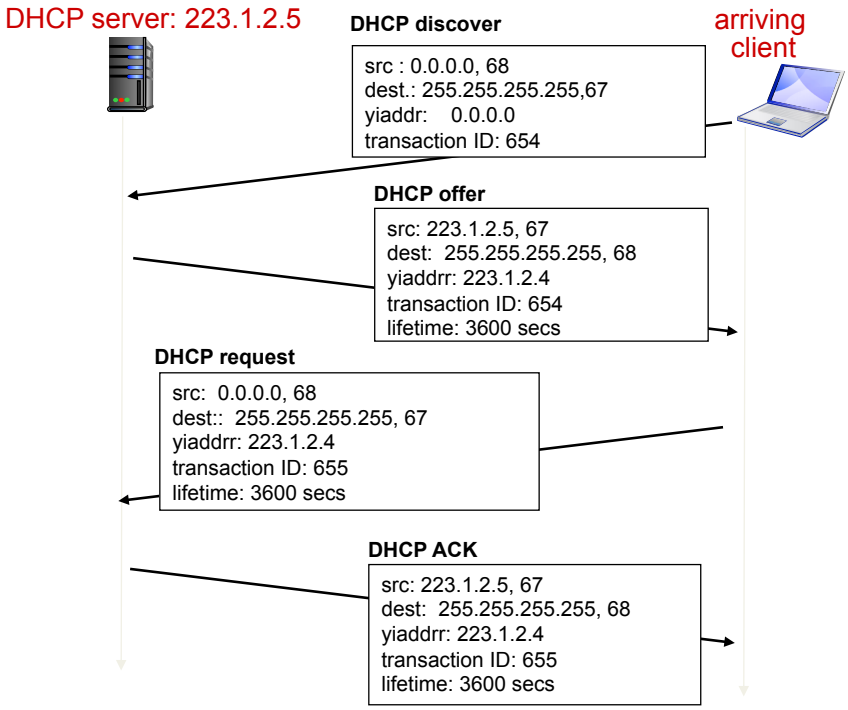
IP addresses: how to get one?

Q: How does a *host* get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from server

pp 371-375

DHCP client-server scenario



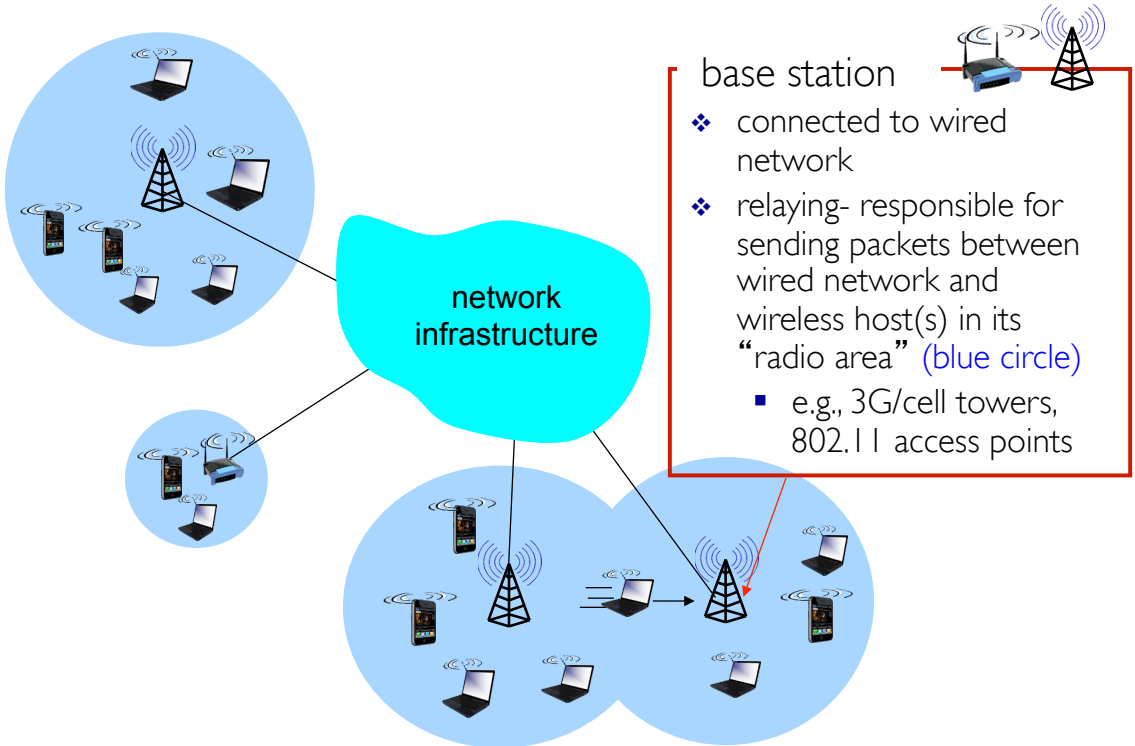
pp 371-375

UPPSALA UNIVERSITET

IDT066
Distributed Information Systems

Chapter 6
Wireless, WiFi and mobility

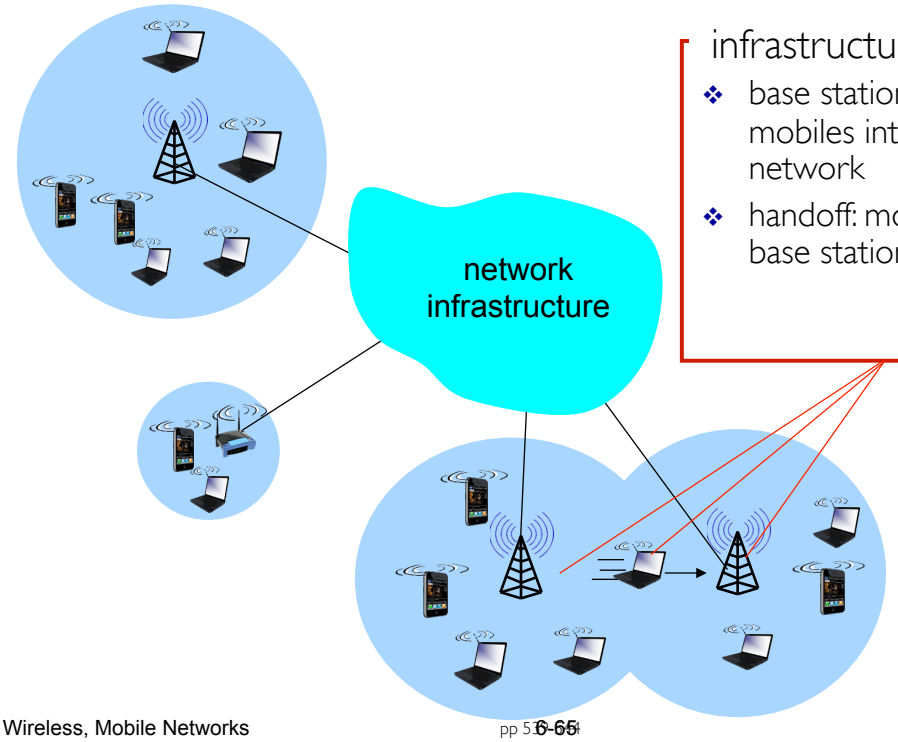
Elements of a wireless network



base station

- connected to wired network
- relaying- responsible for sending packets between wired network and wireless host(s) in its "radio area" (blue circle)
  - e.g., 3G/cell towers, 802.11 access points

Infrastructure mode



infrastructure mode

- base station connects mobiles into wired network
- handoff: mobile changes base station

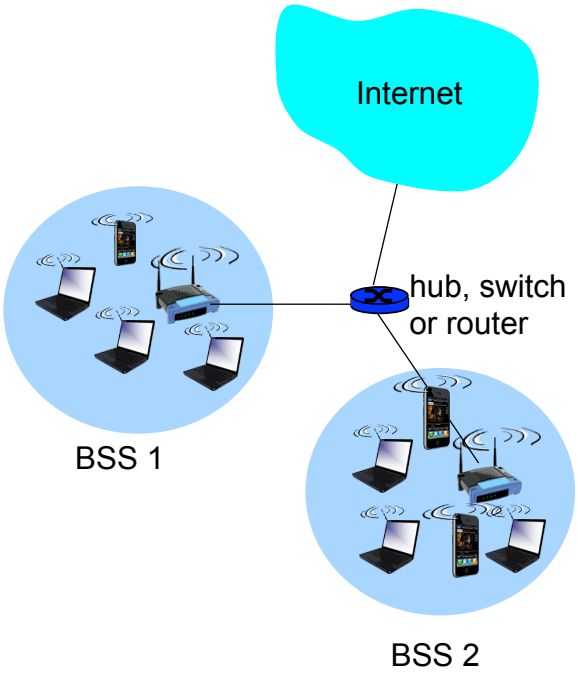
Wireless Link Characteristics

important differences from wired link ….

- *decreased signal strength:* radio signal ***attenuates*** as it propagates through matter (path loss)
- *interference from other sources:* standardized wireless network frequencies (e.g., 2.4 GHz) are shared by other devices (e.g., microwave oven). Electrical devices, such as electrical motors, interfere as well.
- *multipath propagation:* radio signal reflects off objects, the ground, atmosphere, etc. Reflections arrive at destination at slightly different times

…. make communication across (even a point to point) wireless link much more "difficult" compared to a wire.

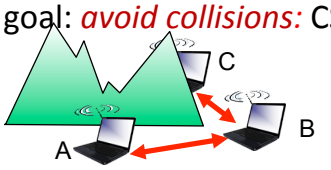
802.11 LAN architecture

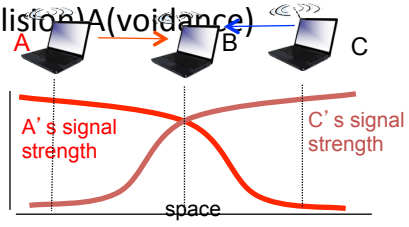


- wireless host communicates with base station
  - base station = access point (AP)
- Basic Service Set (BSS) (aka "cell") in infrastructure mode contains:
  - wireless hosts
  - access point (AP): base station

IEEE 802.11: Sharing the radio channel

- Many nodes can independently chose to send at the same time
- 802.11: Carrier Sense Multiple Access – host *senses (listen)* radio channel if busy before transmitting
  - Don't transmit and collide with ongoing transmission by other node
- 802.11: *difficult to detect a collision*!
  - difficult to receive (sense collisions) when transmitting due to weak received signals
  - can't sense all collisions in any case: *hidden terminal,*
  - goal: *avoid collisions:* CSMA/C(ollision)A(voidance)





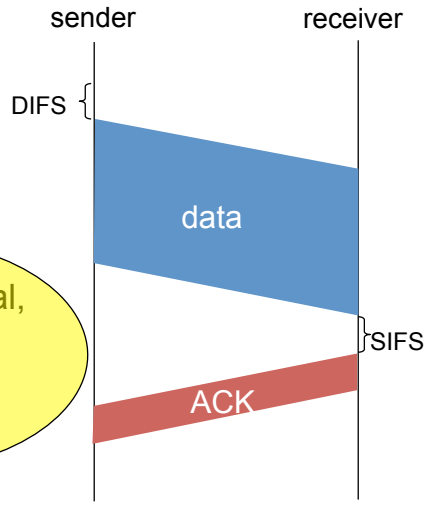
IEEE 802.11 MAC Protocol: CSMA/CA

802.11 sender

1 if sense channel idle for **DIFS** then
transmit entire frame (no CD)

2 if sense channel busy then
start random backoff time
timer counts down while channel idle
transmit when timer expires
if no ACK, increase random backoff interval, repeat 2

Draw this figure!



802.11 receiver

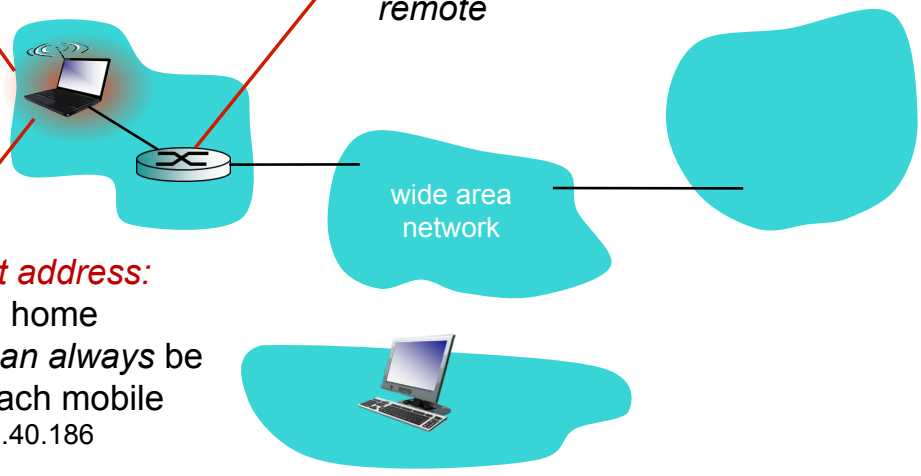
- if frame received OK

return ACK after **SIFS** (ACK needed due to hidden terminal problem)

Mobility: vocabulary

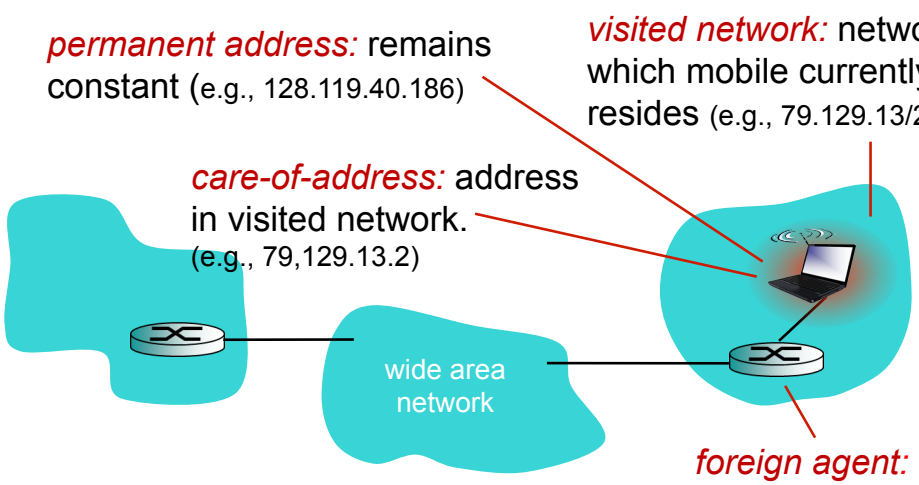
home network: permanent "home" of mobile
(e.g., 128.119.40/24)

home agent: entity that will perform mobility functions on behalf of mobile, when mobile is remote

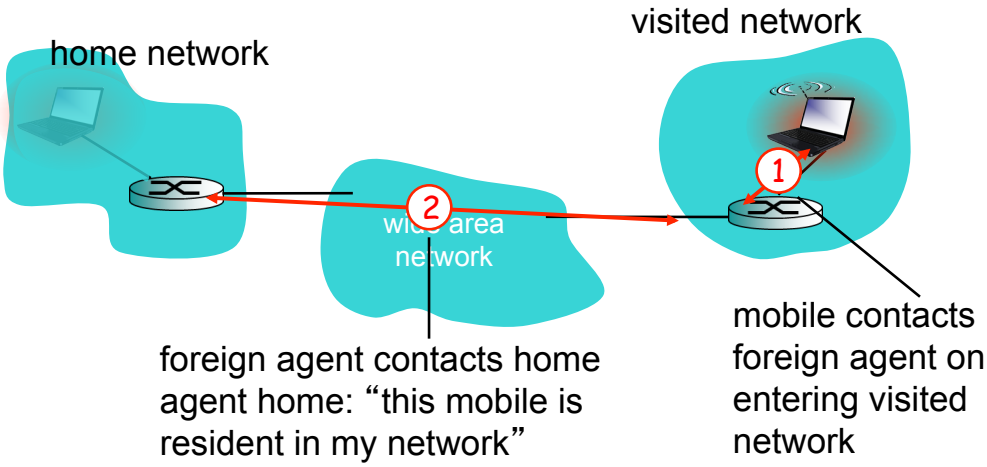


permanent address: address in home network, *can always* be used to reach mobile
e.g., 128.119.40.186

Mobility: more vocabulary



Mobility: registration



end result:

- foreign agent knows about *visiting mobile*
- home agent knows *location* of mobile

Mobility via indirect routing