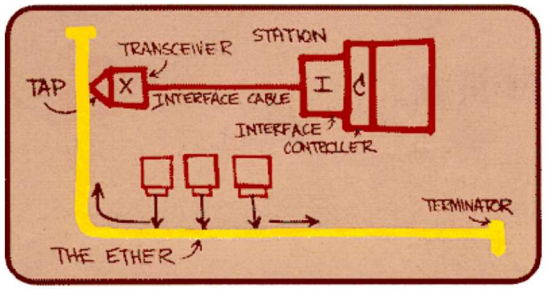
Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP

Ethernet

"dominant" wired LAN technology:

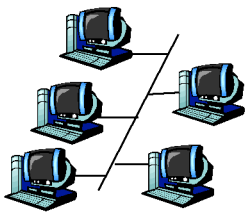
- cheap $20 for NIC
- first widely used LAN technology
- Simpler and cheaper
- kept up with speed race: 10 Mbps – 10 Gbps



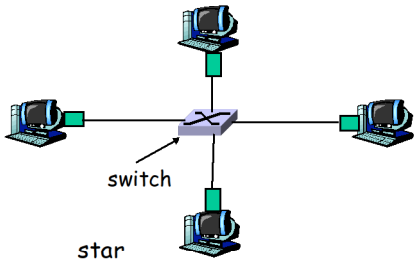
Metcalfe's Ethernet sketch

Star topology

- bus topology popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- today: star topology prevails
  - active *switch* in center



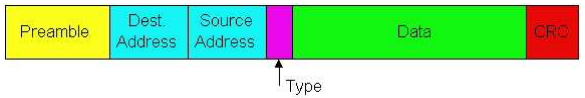
bus: coaxial cable



star

Ethernet Frame Structure

Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame



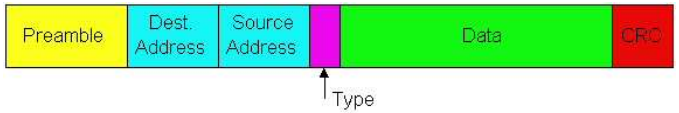
Preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

Preamble = 序文；电报报头；先兆

Ethernet Frame Structure (more)

- Addresses: 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- Type: indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- CRC: checked at receiver, if error is detected, frame is dropped



Ethernet: Unreliable, connectionless

- connectionless: ***No handshaking*** between sending and receiving NICs
- unreliable: receiving NIC doesn't send acks or nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP

  (reliability 由 TCP 层负责)
  - otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted CSMA/CD

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 **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

类似于全双工：一边发送，一遍可以侦听
退避机制

Ethernet's CSMA/CD (more)

Jam Signal: make sure all other transmitters are aware of collision; 48 bits

Bit time: .1 microsec for 10 Mbps Ethernet ; for K=1023, wait time is about 50 msec

如果冲突，双方退避等待
等多久？随机时间

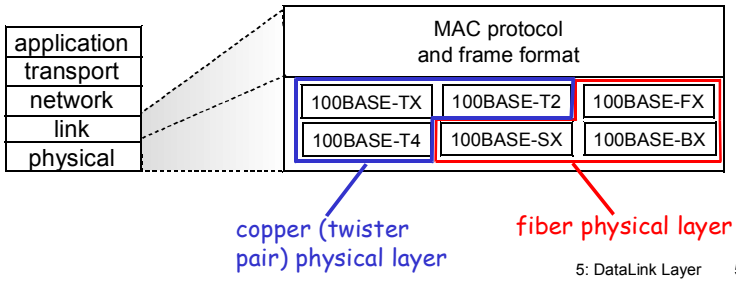
Exponential Backoff:

- *Goal*: adapt retransmission attempts to estimated current load
  - **If heavy load: random wait will be longer**
- first collision: choose K from {0,1}; delay is K· 512 bit transmission times
- after second collision: choose K from {0,1,2,3}…
- after ten collisions, choose K from {0,1,2,3,4,…,1023}

802.3 Ethernet Standards: Link & Physical Layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - different physical layer media: fiber, cable

Common link layer, different PHY layers

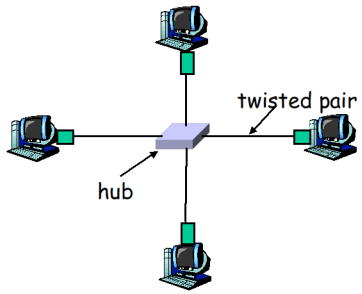


Link Layer

Hubs

… physical-layer ("dumb") repeaters (简单转发):

- bits coming in one link go out *all* other links at same rate
- all nodes connected to hub can collide with one another
- no frame buffering
- no CSMA/CD at hub: host NICs detect collisions

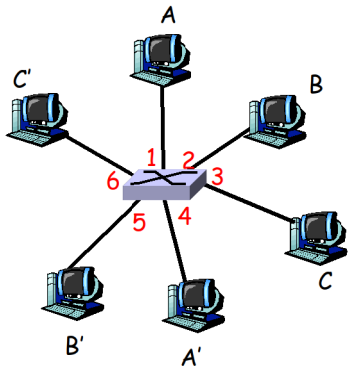


Switch (选择性转发，基于 MAC 地址，和路由器区别)

- link-layer device: smarter than hubs, take *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

Switch: allows *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub

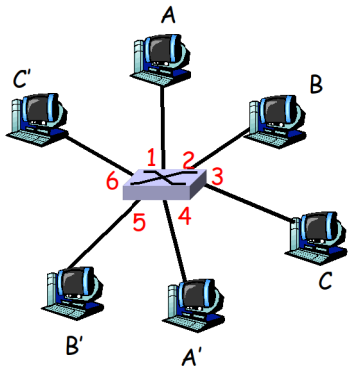


switch with six interfaces (1,2,3,4,5,6)

Switch Table

当 frame 到达 switch
记录其 MAC 地址和到达的接口

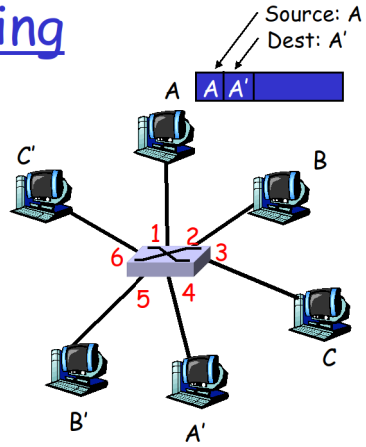
- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- *A:* each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- *Q:* how are entries created, maintained in switch table?
  - something like a routing protocol?



switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Switch: frame filtering/forwarding

When frame received:

1. record link associated with sending host
2. index switch table using MAC dest address
3. if entry found for destination
   then {
   if dest on segment from which frame arrived
   then drop the frame
   else forward the frame on interface indicated
   }
   else flood

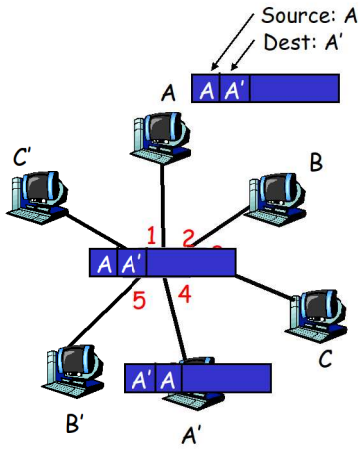
表里已有 dest 接口
则转发 dest 接口

如果表里没有 dest 接口，则群发

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

- frame destination unknown: *flood*
- destination A location known: *selective send*

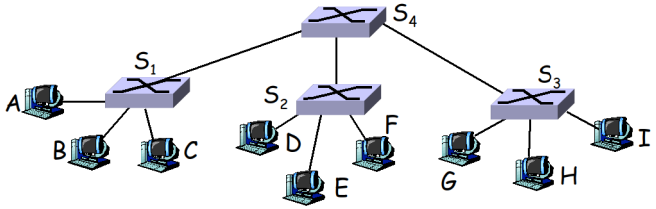


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

Interconnecting switches

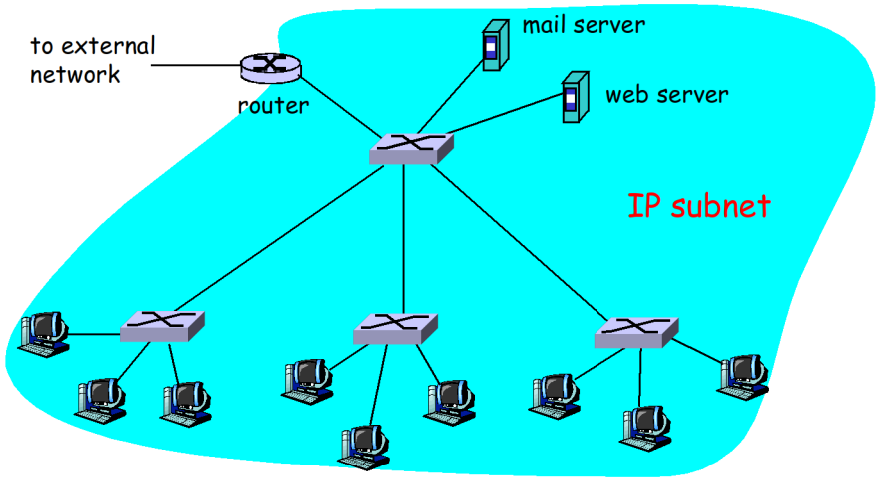
- switches can be connected together



- *Q:* sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?
- *A:* self learning! (works exactly the same as in single-switch case!)

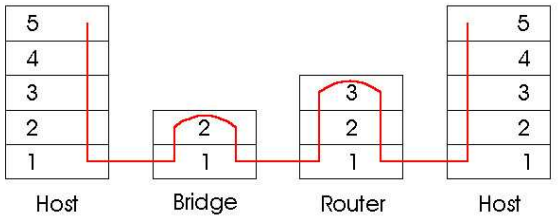
Institutional network

内部不需要 router
可以做到快速转发



Switches vs. Routers

- both **store-and-forward** devices
  - routers: network layer devices (examine network layer headers)
  - switches are link layer devices
- routers maintain routing tables, implement **routing** algorithms
- switches maintain switch tables, implement **filtering**, **learning** algorithms



Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Hubs and switches
- 5.7 PPP

Point to Point Data Link Control

- one sender, one receiver, **one link: easier than broadcast link**:
  - no Media Access Control
  - no need for explicit MAC addressing
  - e.g., **dialup** link, ISDN line
- popular point-to-point DLC protocols:
  - PPP (point-to-point protocol)
  - HDLC: High level data link control (Data link used to be considered "high layer" in protocol stack!)

例：拨号上网 link

PPP Design Requirements [RFC 1557]

- packet framing: encapsulation of network-layer datagram in data link frame
  - carry network layer data of any network layer protocol (not just IP) *at same time*
- ***bit transparency: must carry any bit pattern in the data field***
- error detection (no correction)
- connection liveness: detect, signal link failure to network layer
- network layer address negotiation: endpoint can learn/configure each other's network address

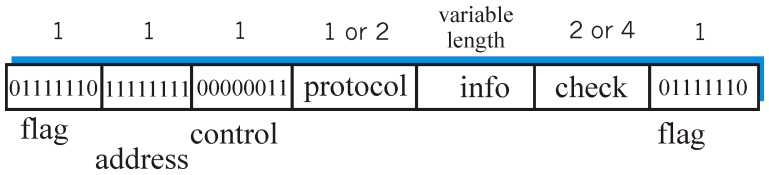
PPP non-requirements

- no error correction/recovery
- no flow control
- out of order delivery OK
- no need to support multipoint links (e.g., polling)

Error recovery, flow control, data re-ordering all relegated to higher layers!

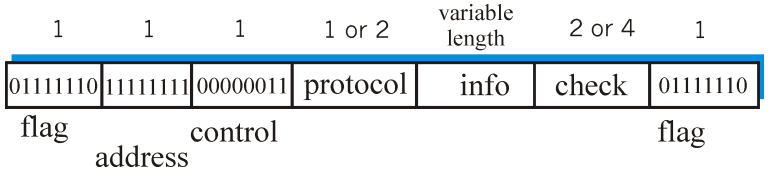
PPP Data Frame

- Flag: delimiter (framing)
- Address: does nothing (only one option)
- Control: does nothing; in the future possible multiple control fields
- Protocol: upper layer protocol to which frame delivered (eg, PPP-LCP, IP, IPCP, etc)



PPP Data Frame

- info: upper layer data being carried
- check: cyclic redundancy check for error detection



Chapter 5: Summary

- principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies
  - Ethernet
  - switched LANS
  - PPP

Chapter 5: let's take a breath

- journey down protocol stack *complete* (except PHY)
- solid understanding of networking principles, practice
- ..... could stop here .... but *lots* of interesting topics!
  - wireless
  - multimedia
  - security
  - network management

Chapter 5: Link layer

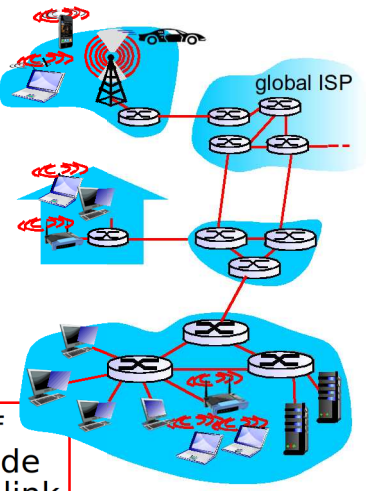
our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet
- instantiation, implementation of various link layer technologies

Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs
- layer-2 packet: frame, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link

Link layer services

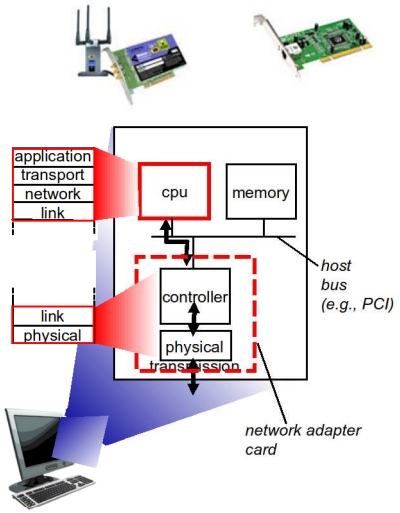
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?

Link layer services (more)

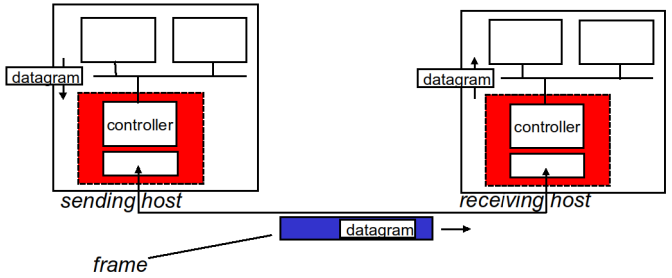
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Adaptors communicating

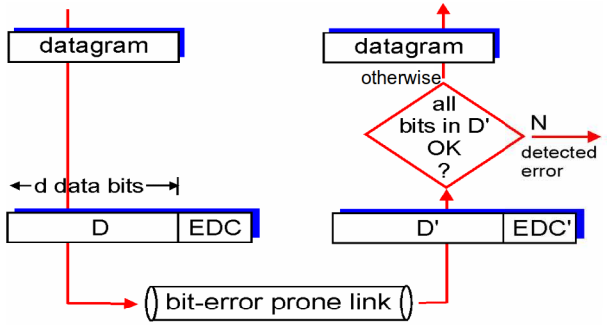


- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Error detection

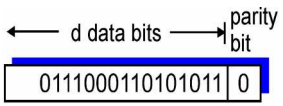
EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

• Error detection not 100% reliable!
  • protocol may miss some errors, but rarely
  • larger EDC field yields better detection and correction

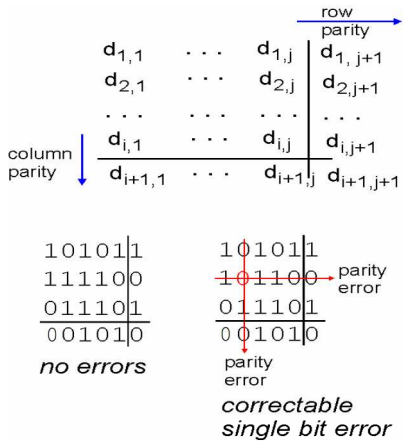


Parity checking

single bit parity:
- *d*etect single bit errors



two-dimensional bit parity:
- detect and correct single bit errors



Internet checksum (review)

goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer *only*)

sender:

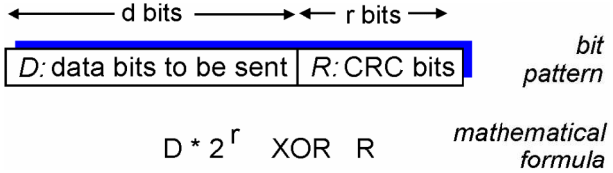
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

Cyclic redundancy check

- more powerful error-detection coding
- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



CRC example

want:

$D \cdot 2^r$ XOR R = nG

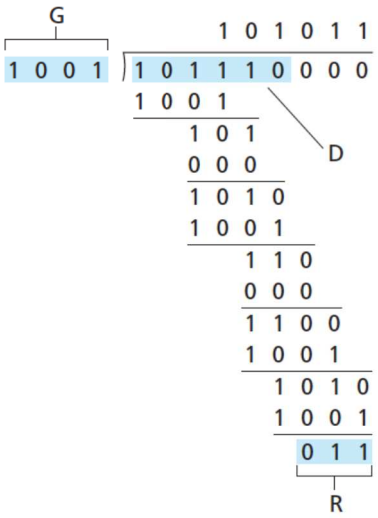
equivalently:

$D \cdot 2^r$ = nG XOR R

equivalently:

if we divide $D \cdot 2^r$ by G, want remainder R to satisfy:

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$



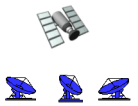
Multiple access links, protocols

two types of "links":

- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

MAC protocols: taxonomy

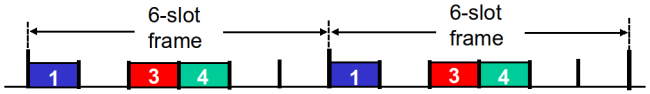
three broad classes:

- *channel partitioning*
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use
- *random access*
  - channel not divided, allow collisions
  - “recover” from collisions
- *“taking turns”*
  - nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

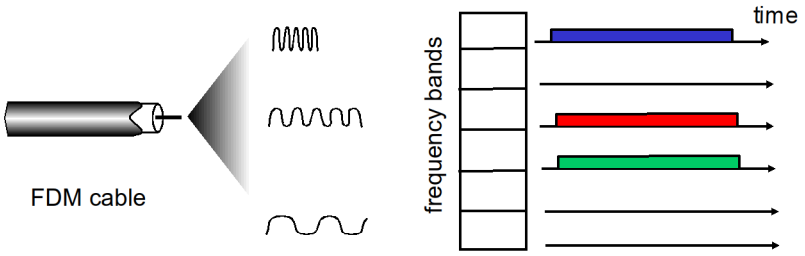
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

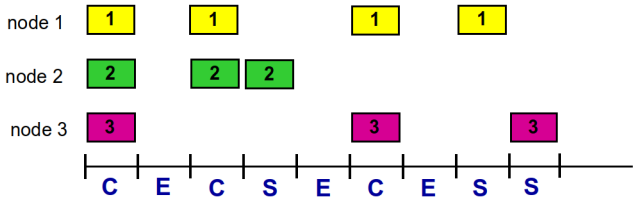
- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes → “collision”,
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA



Pros:

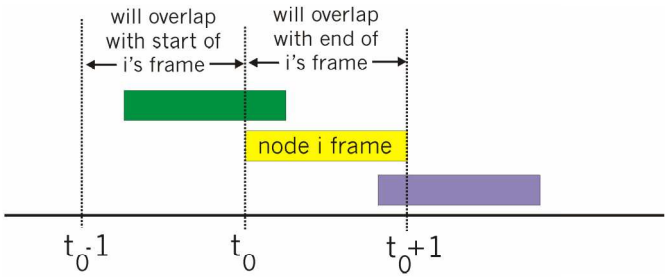
- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at t_0 collides with other frames sent in $[t_0-1,t_0+1]$



CSMA (carrier sense multiple access)

CSMA: listen before transmit:

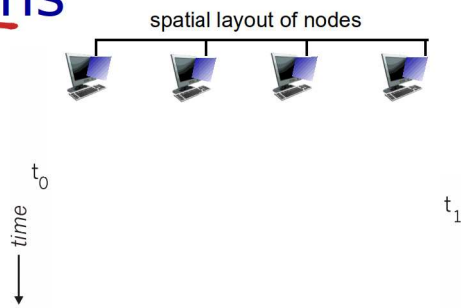
if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

- human analogy: don't interrupt others!

CSMA collisions

- collisions *can* still occur: propagation delay means two nodes may not hear each other's transmission



- collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability

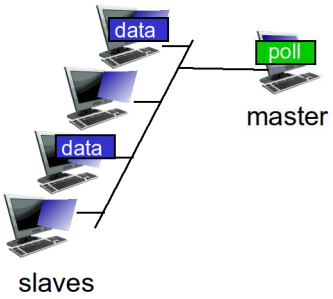
CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- After aborting, NIC enters *binary (exponential) backoff:*
  - after *m*th collision, NIC chooses *K* at random from *{0,1,2, …, 2^m-1}*. NIC waits K·512 bit times, returns to Step 2
  - longer backoff interval with more collisions
- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

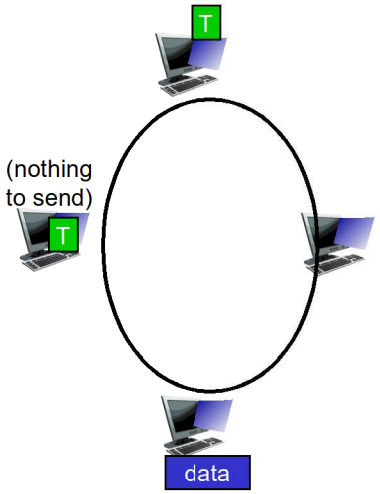
"Taking turns" MAC protocols

polling:
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"Taking turns" MAC protocols

token passing:
- control *token* passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



Summary of MAC protocols

- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
  - share channel *efficiently* and *fairly* at high load
  - inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - efficient at low load: single node can fully utilize channel
  - high load: collision overhead
- *taking turns*
  - polling from central site, token passing
  - bluetooth, FDDI, token ring

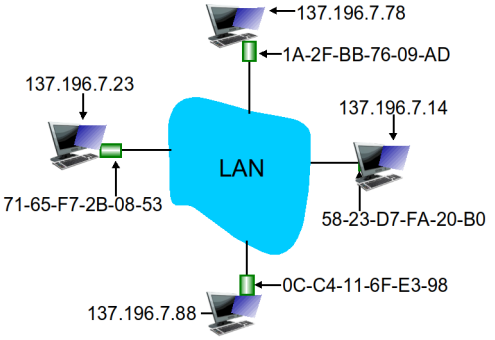
MAC addresses and ARP

- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - function: *used 'locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
    - e.g.: 1A-2F-BB-76-09-AD

hexadecimal (base 16) notation
(each "number" represents 4 bits)

ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:
  < IP address; MAC address; TTL>
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

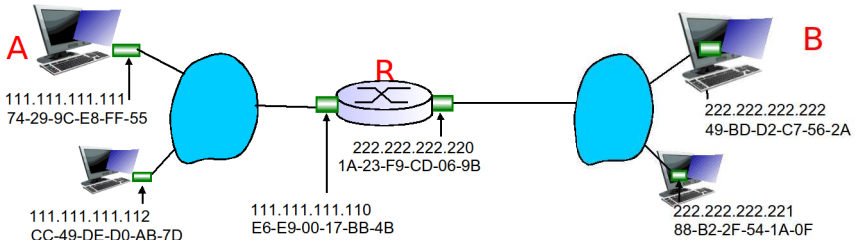
ARP protocol: same LAN

- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

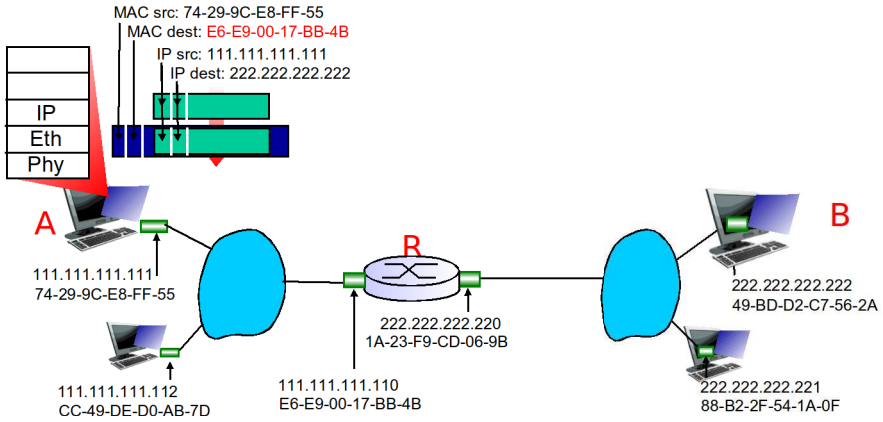
walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R's MAC address (how?)



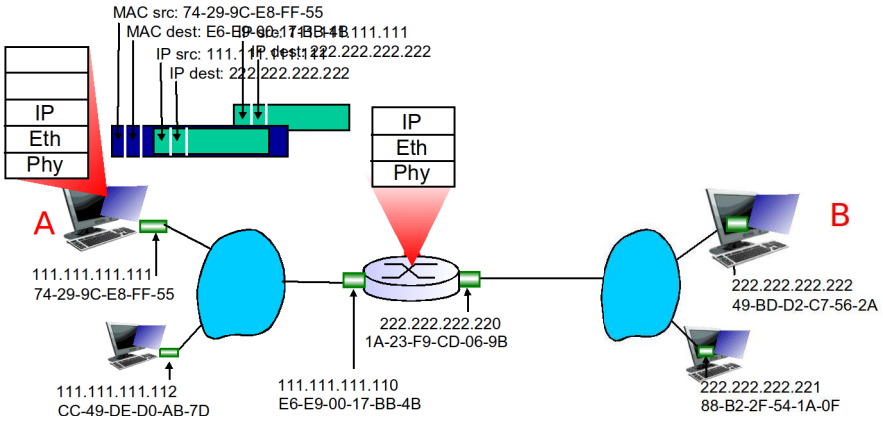
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



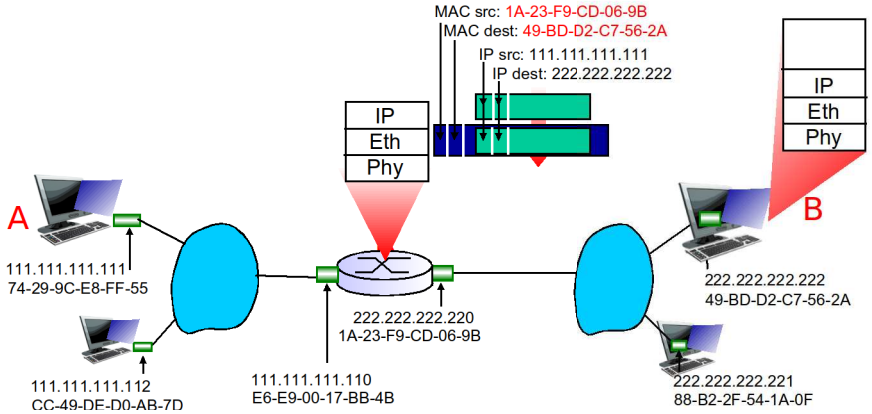
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



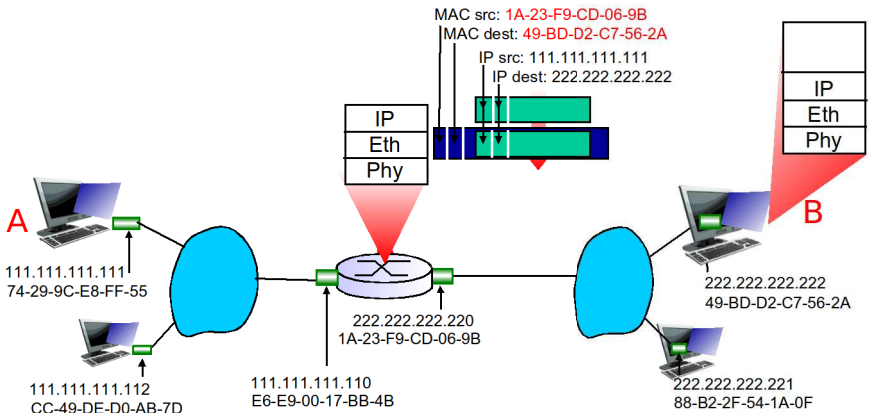
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



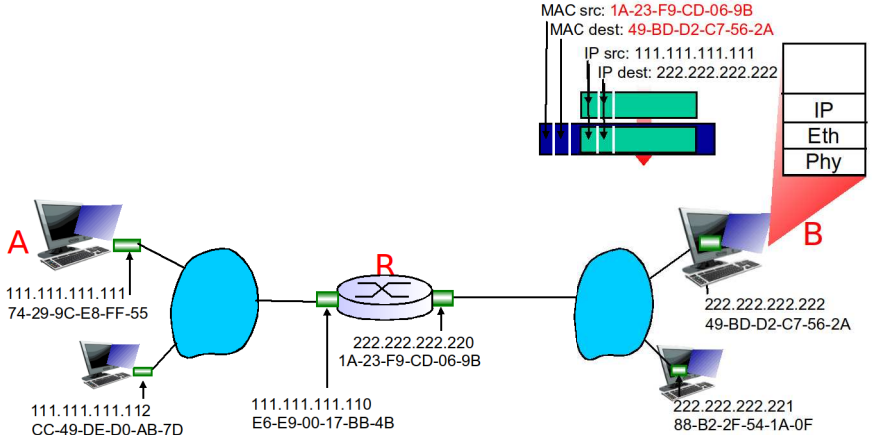
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Ethernet frame structure

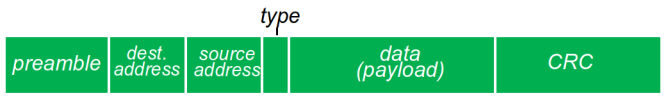
sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

Ethernet frame structure (more)

- *addresses:* 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- *type:* indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- *CRC:* cyclic redundancy check at receiver
  - error detected: frame is dropped



Ethernet: unreliable, connectionless

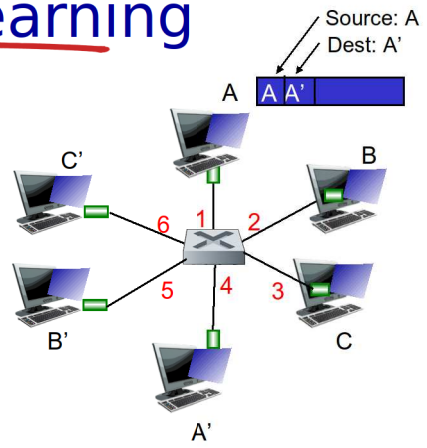
- *connectionless:* no handshaking between sending and receiving NICs
- *unreliable:* receiving NIC doesnt send acks or nacks to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet's MAC protocol: unslotted *CSMA/CD wth binary backoff*

Ethernet switch

- link-layer device: takes an *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

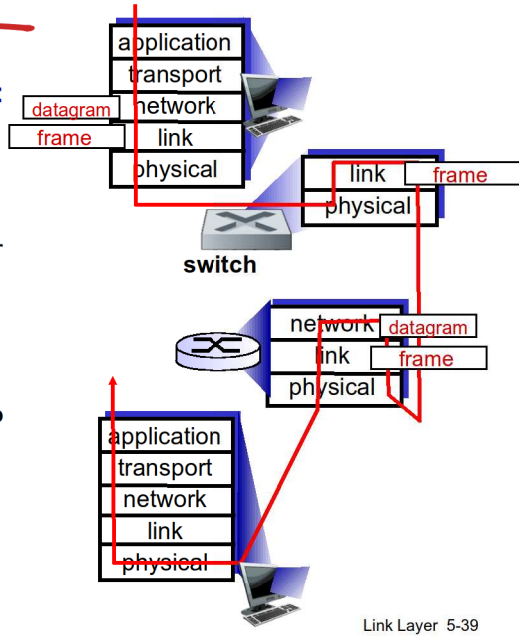
Switches vs. routers

both are store-and-forward:

- *routers:* network-layer devices (examine network-layer headers)
- *switches:* link-layer devices (examine link-layer headers)

both have forwarding tables:

- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



Data center networks

- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
  - e-business (e.g. Amazon)
  - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
  - search engines, data mining (e.g., Google)

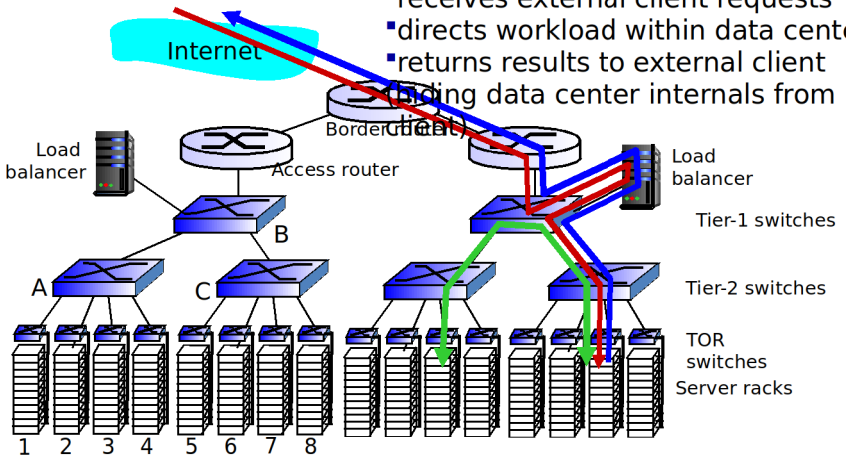
- challenges:
  - multiple applications, each serving massive numbers of clients
  - managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Data center networks

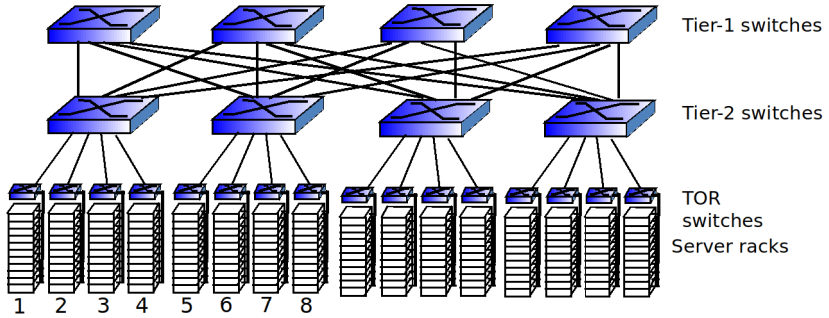
load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)

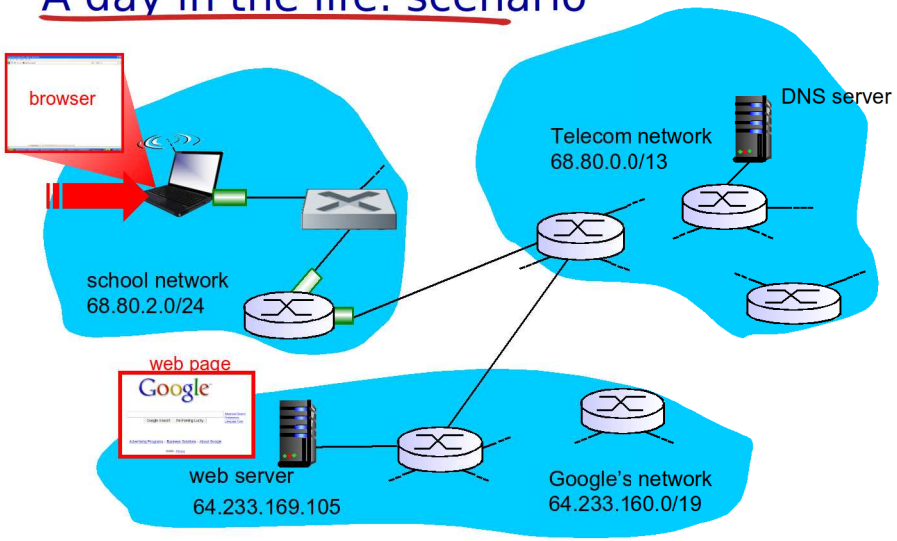


Data center networks

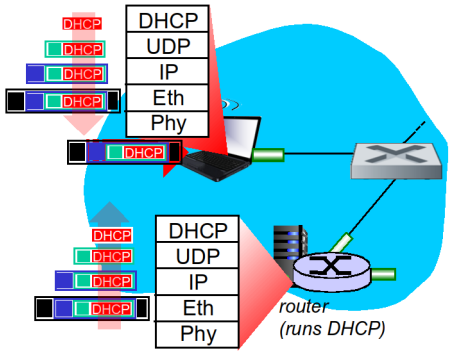
- rich interconnection among switches, racks:
  - increased throughput between racks (multiple routing paths possible)
  - increased reliability via redundancy



A day in the life: scenario

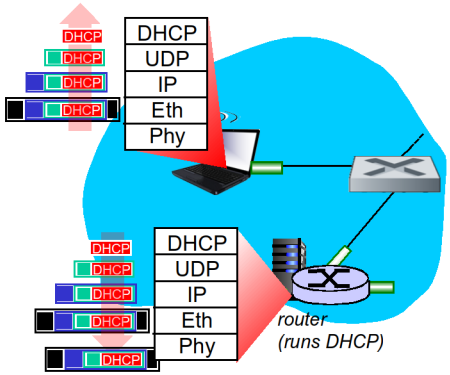


A day in the life... connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*
- DHCP request *encapsulated* in *UDP*, encapsulated in *IP*, encapsulated in *802.3* Ethernet
- Ethernet frame *broadcast* (dest: FFFFFFFFFFFF) on LAN, received at router running *DHCP* server
- Ethernet *demuxed* to IP demuxed, UDP demuxed to DHCP

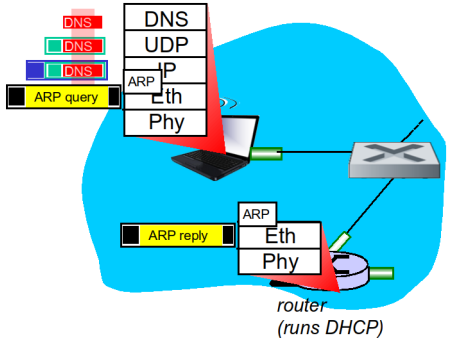
A day in the life… connecting to the Internet



- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (*switch learning*) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

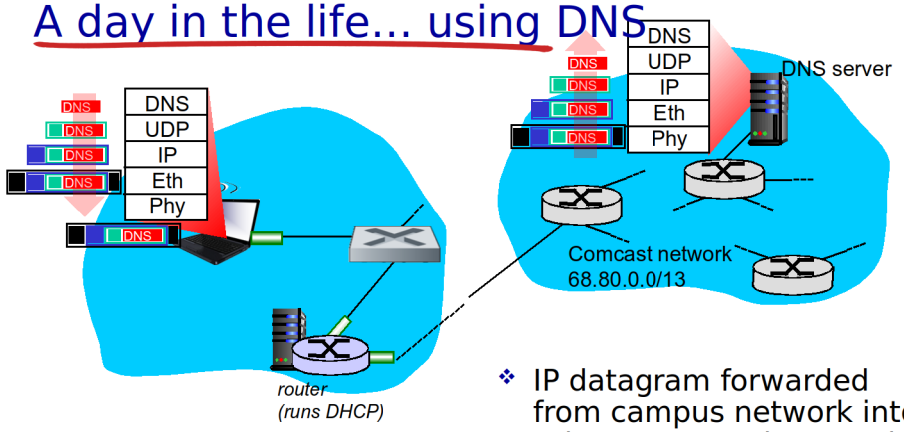
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life… ARP (before DNS, before HTTP)



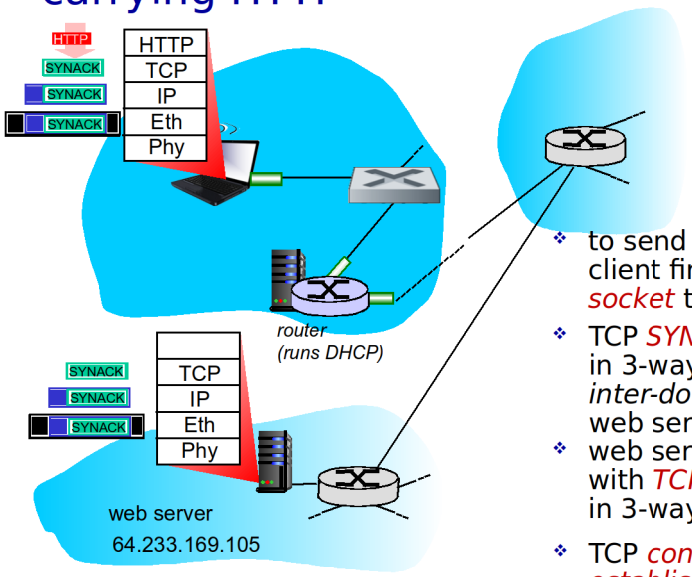
- before sending *HTTP* request, need IP address of www.google.com: *DNS*
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: *ARP*
- *ARP query* broadcast, received by router, which replies with *ARP reply* giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life… using DNS



- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into Telecom network, routed (tables created by *RIP, OSPF, IS-IS* and/or *BGP* routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



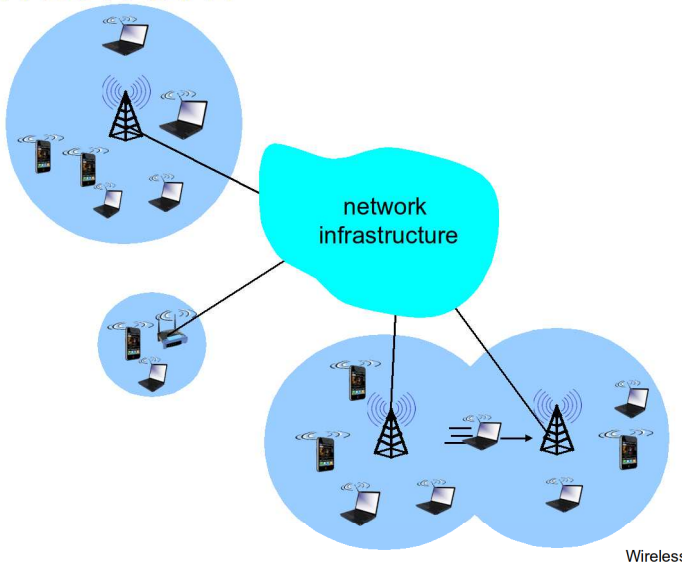
- to send HTTP request, client first opens *TCP socket* to web server
- TCP *SYN segment* (step 1 in 3-way handshake) *inter-domain routed* to web server
- web server responds with *TCP SYNACK* (step 2 in 3-way handshake)
- TCP *connection established!*

Ch. 6: Wireless and Mobile Networks

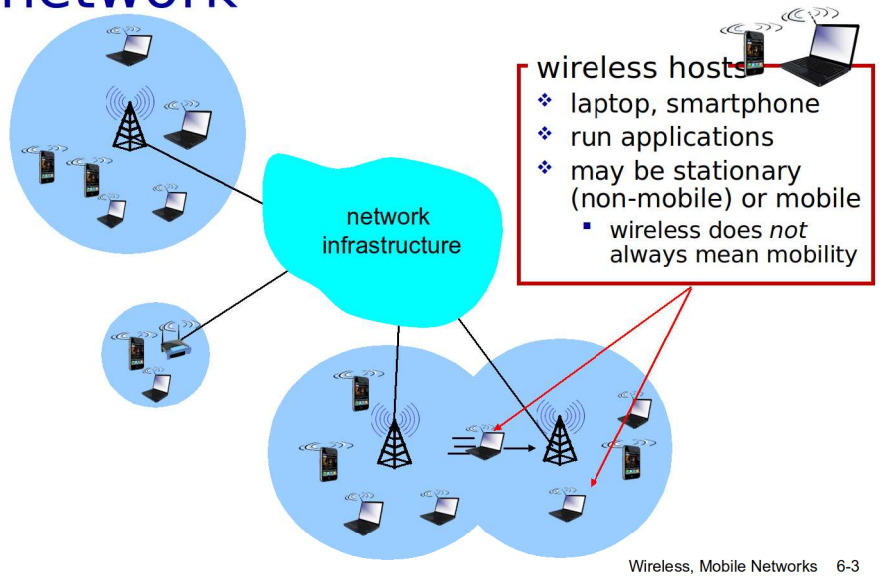
Background:

- # wireless (mobile) phone subscribers now exceeds # wired phone subscribers (5-to-1)!
- # wireless Internet-connected devices equals # wireline Internet-connected devices
  - laptops, Internet-enabled phones promise anytime untethered Internet access
- two important (but different) challenges
  - *wireless:* communication over wireless link
  - *mobility:* handling the mobile user who changes point of attachment to network

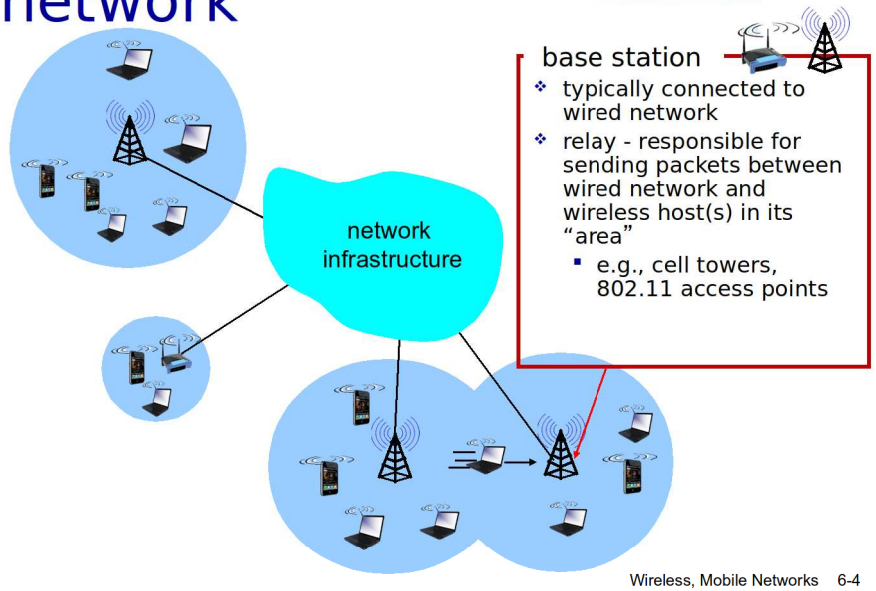
Elements of a wireless network



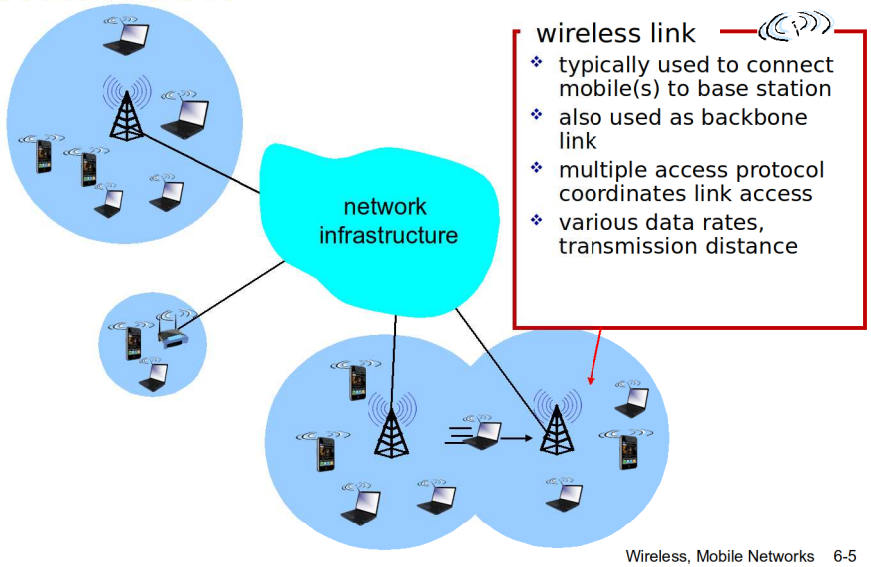
Elements of a wireless network



Elements of a wireless network

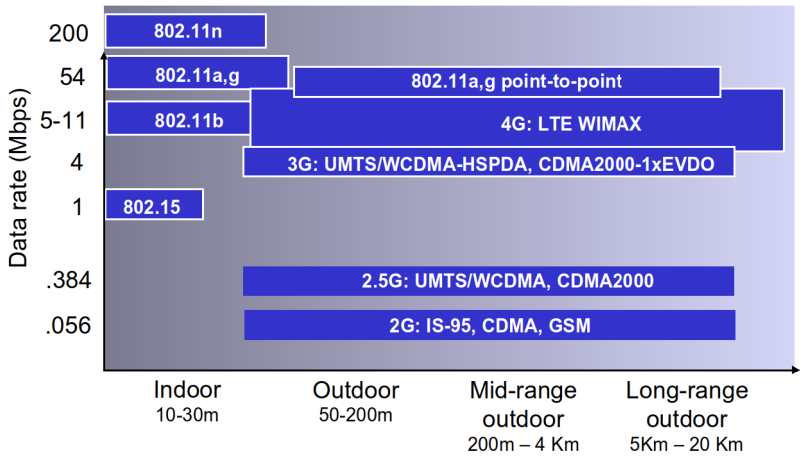


Elements of a wireless network

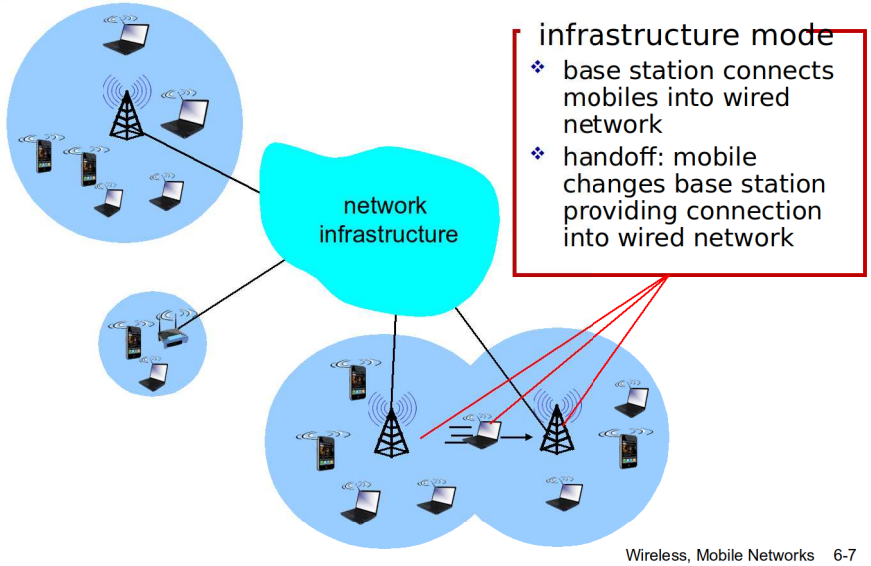


wireless link
- typically used to connect mobile(s) to base station
- also used as backbone link
- multiple access protocol coordinates link access
- various data rates, transmission distance

Characteristics of selected wireless links

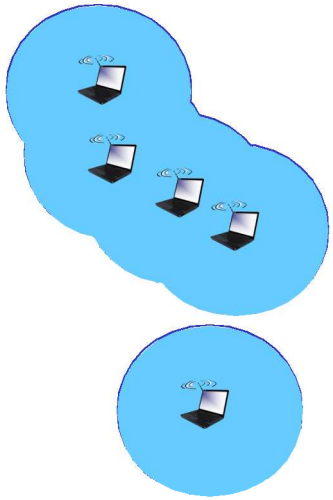


Elements of a wireless network



infrastructure mode
- base station connects mobiles into wired network
- handoff: mobile changes base station providing connection into wired network

Elements of a wireless network



ad hoc mode
- no base stations
- nodes can only transmit to other nodes within link coverage
- nodes organize themselves into a network: route among themselves

Wireless Link Characteristics

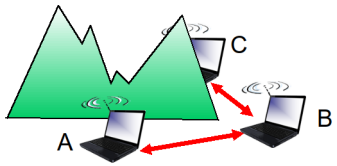
important differences from wired link ....

- *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) shared by other devices (e.g., phone); devices (motors) interfere as well
- *multipath propagation:* radio signal reflects off objects ground, arriving ad destination at slightly different times

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

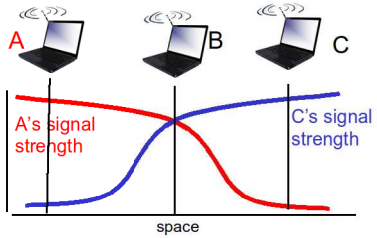
Wireless network characteristics

Multiple wireless senders and receivers create additional problems (beyond multiple access):



Hidden terminal problem

- B, A hear each other
- B, C hear each other
- A, C can not hear each other means A, C unaware of their interference at B



Signal attenuation:

- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B

Code Division Multiple Access (CDMA)

- unique "code" assigned to each user; i.e., code set partitioning
  - all users share same frequency, but each user has own "chipping" sequence (i.e., code) to encode data
  - allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")
- *encoded signal* = (original data) X (chipping sequence)
- *decoding:* inner-product of encoded signal and chipping sequence

IEEE 802.11 Wireless LAN

802.11b

- 2.4-5 GHz unlicensed spectrum
- up to 11 Mbps
- direct sequence spread spectrum (DSSS) in physical layer
  - all hosts use same chipping code

802.11a

- 5-6 GHz range
- up to 54 Mbps

802.11g

- 2.4-5 GHz range
- up to 54 Mbps

802.11n: multiple antennae

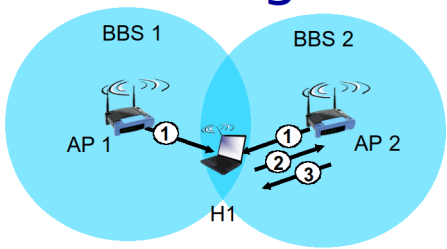
- 2.4-5 GHz range
- up to 200 Mbps

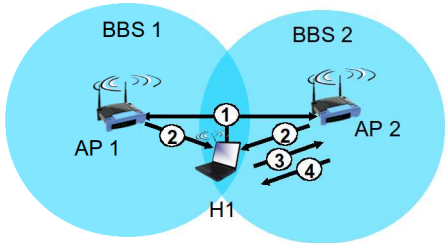
- all use CSMA/CA for multiple access
- all have base-station and ad-hoc network versions

802.11: Channels, association

- 802.11b: 2.4GHz-2.485GHz spectrum divided into 11 channels at different frequencies
  - AP admin chooses frequency for AP
  - interference possible: channel can be same as that chosen by neighboring AP!
- host: must *associate* with an AP
  - scans channels, listening for *beacon frames* containing AP's name (SSID) and MAC address
  - selects AP to associate with
  - may perform authentication
  - will typically run DHCP to get IP address in AP's subnet

802.11: passive/active scanning





passive scanning:

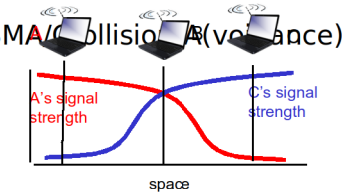
(1) beacon frames sent from APs
(2) association Request frame sent: H1 to selected AP
(3) association Response frame sent from selected AP to H1

active scanning:

(1) Probe Request frame broadcast from H1
(2) Probe Response frames sent from APs
(3) Association Request frame sent: H1 to selected AP
(4) Association Response frame sent from selected AP to H1

IEEE 802.11: multiple access

- avoid collisions: 2+ nodes transmitting at same time
- 802.11: CSMA - sense before transmitting
  - don't collide with ongoing transmission by other node
- 802.11: *no* collision detection!
  - difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
  - can't sense all collisions in any case: hidden terminal, fading
  - goal: *avoid collisions:* CSMA/C(ollision)A(voidance)



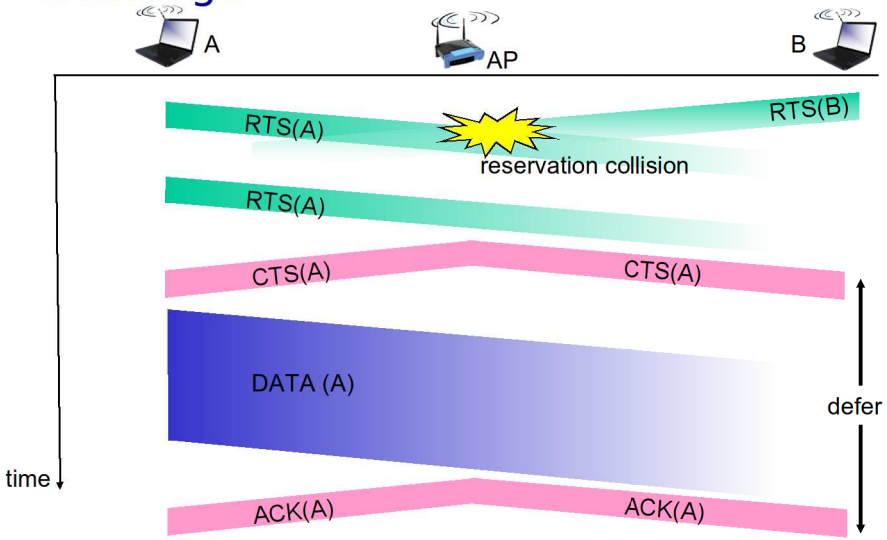
Avoiding collisions

idea: allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames

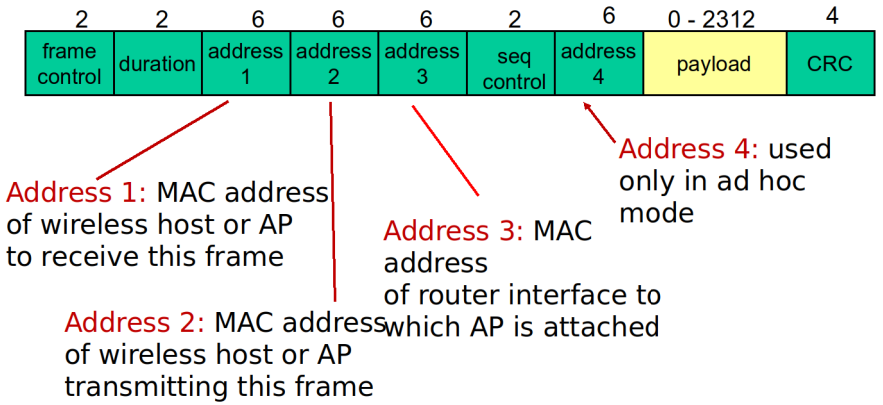
- sender first transmits *small* request-to-send (RTS) packets to BS using CSMA
  - RTSs may still collide with each other (but they're short)
- BS broadcasts clear-to-send CTS in response to RTS
- CTS heard by all nodes
  - sender transmits data frame
  - other stations defer transmissions

avoid data frame collisions completely using small reservation packets!

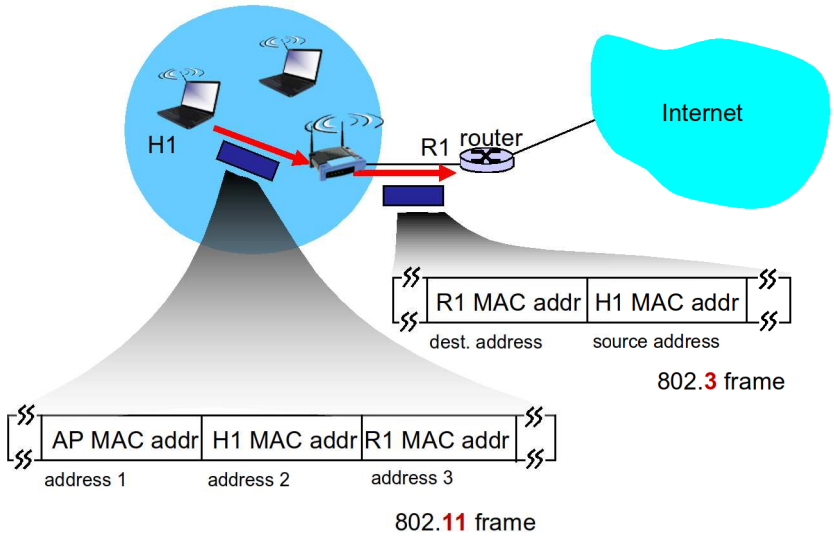
Collision Avoidance: RTS-CTS exchange



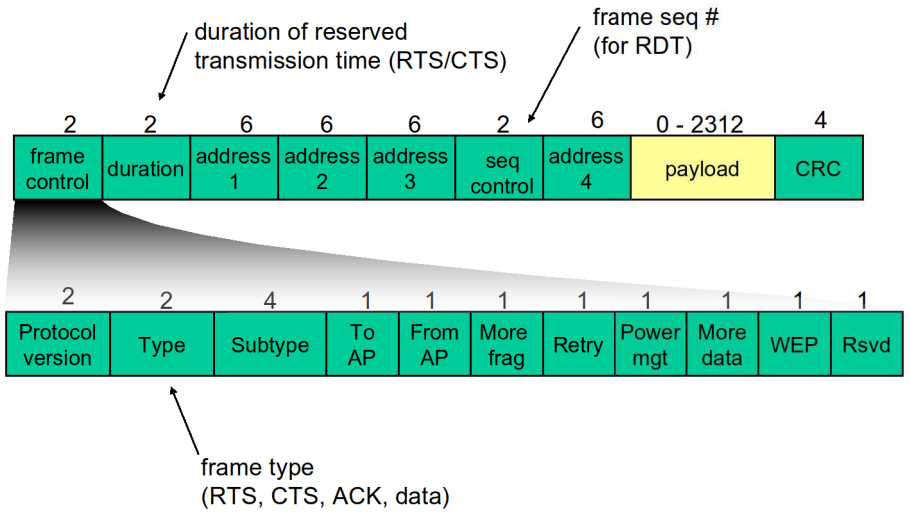
802.11 frame: addressing



802.11 frame: addressing

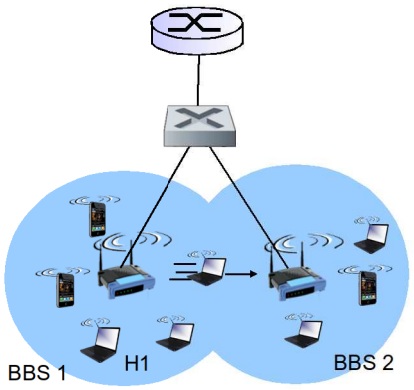


802.11 frame: more



802.11: mobility within same subnet

- H1 remains in same IP subnet: IP address can remain same
- switch: which AP is associated with H1?
  - self-learning (Ch. 5): switch will see frame from H1 and "remember" which switch port can be used to reach H1



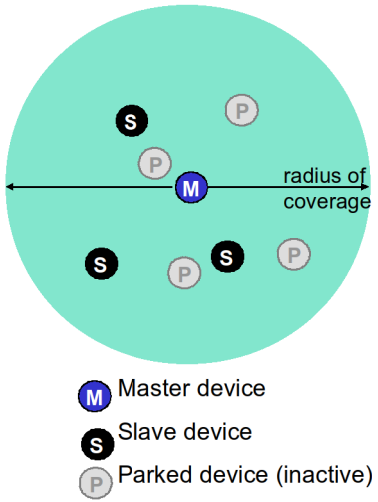
802.11: advanced capabilities

power management

- node-to-AP: "I am going to sleep until next beacon frame"
  - AP knows not to transmit frames to this node
  - node wakes up before next beacon frame
- beacon frame: contains list of mobiles with AP-to-mobile frames waiting to be sent
  - node will stay awake if AP-to-mobile frames to be sent; otherwise sleep again until next beacon frame

802.15: personal area network

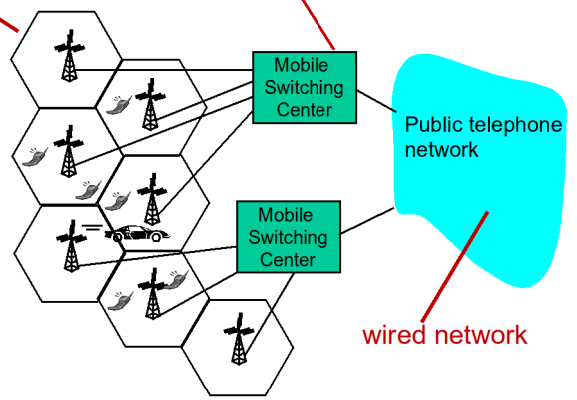
- less than 10 m diameter
- replacement for cables (mouse, keyboard, headphones)
- ad hoc: no infrastructure
- master/slaves:
  - slaves request permission to send (to master)
  - master grants requests
- 802.15: evolved from Bluetooth specification
  - 2.4-2.5 GHz radio band
  - up to 721 kbps



components of cellular network architecture

MSC
- connects cells to wired tel. net.
- manages call setup (more later!)
- handles mobility (more later!)

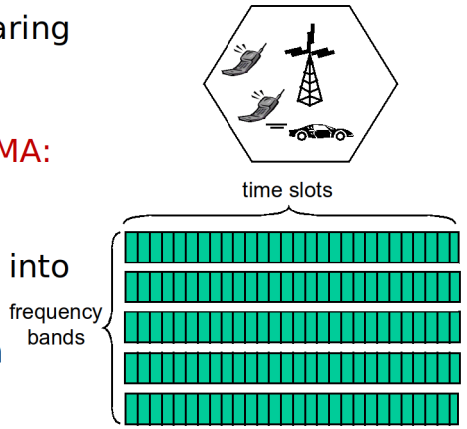
cell
- covers geographical region
- *base station* (BS) analogous to 802.11 AP
- *mobile users* attach to network through BS
- *air-interface:* physical and link layer protocol between mobile and BS



Cellular networks: the first hop

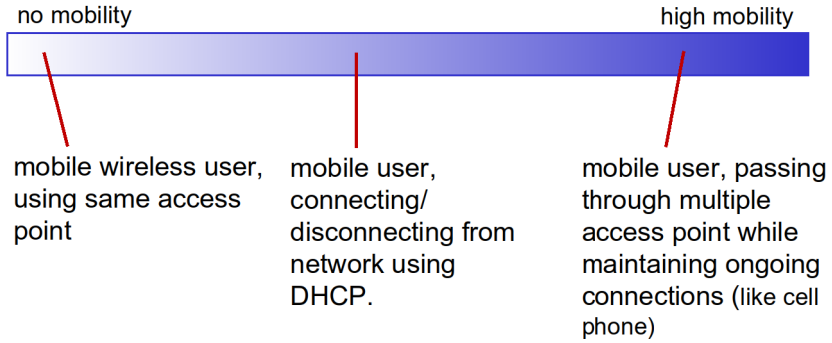
Two techniques for sharing mobile-to-BS radio spectrum

- combined FDMA/TDMA: divide spectrum in frequency channels, divide each channel into time slots
- CDMA: code division multiple access



What is mobility?

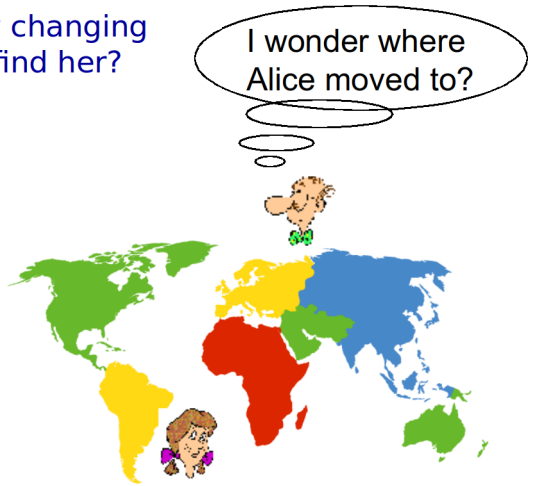
- spectrum of mobility, from the *network* perspective:



How do *you* contact a mobile friend:

Consider friend frequently changing addresses, how do you find her?

- search all phone books?
- call her parents?
- expect her to let you know where he/she is?



Mobility: approaches

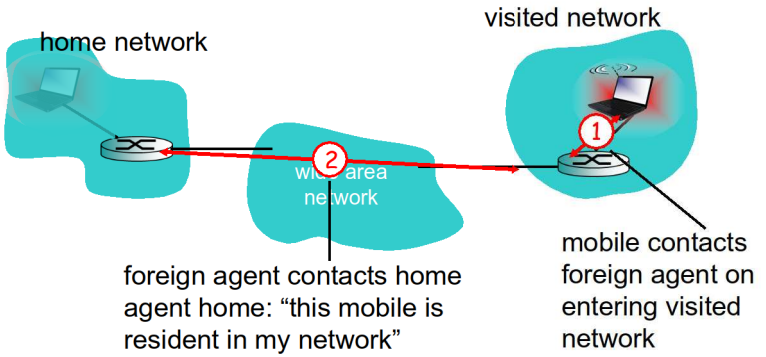
- *let routing handle it:* routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
- *let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

Mobility: approaches

- *let routing handle it:* routers advertise permanent address of mobile-nodes in residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
- *let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

not scalable to millions of mobiles

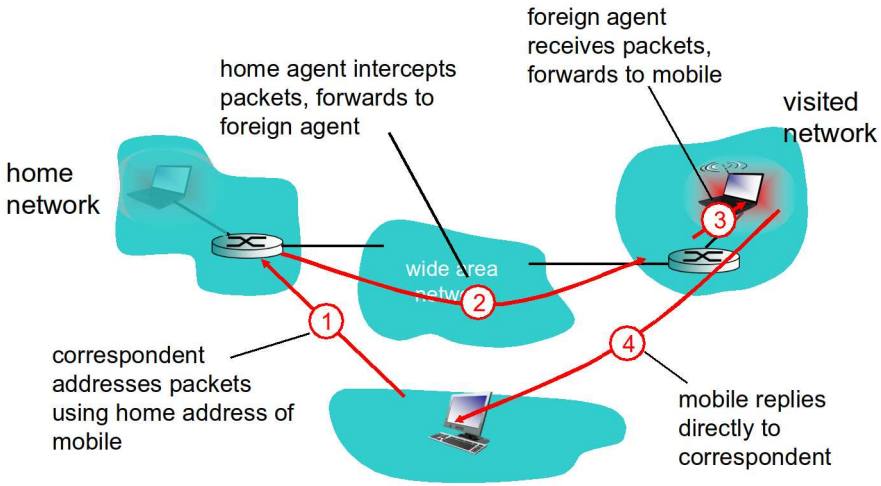
Mobility: registration



end result:

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

Mobility via indirect routing



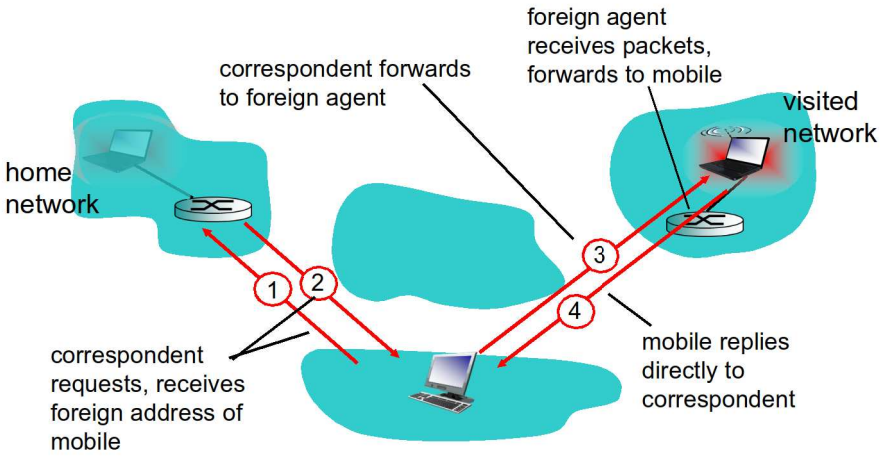
Indirect Routing: comments

- mobile uses two addresses:
  - permanent address: used by correspondent (hence mobile location is *transparent* to correspondent)
  - care-of-address: used by home agent to forward datagrams to mobile
- foreign agent functions may be done by mobile itself
- triangle routing: correspondent-home-network-mobile
  - inefficient when correspondent, mobile are in same network

Indirect routing: moving between networks

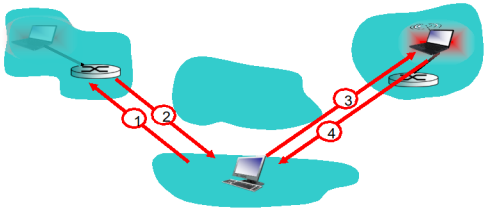
- suppose mobile user moves to another network
  - registers with new foreign agent
  - new foreign agent registers with home agent
  - home agent update care-of-address for mobile
  - packets continue to be forwarded to mobile (but with new care-of-address)
- mobility, changing foreign networks transparent: *on going connections can be maintained!*

Mobility via direct routing



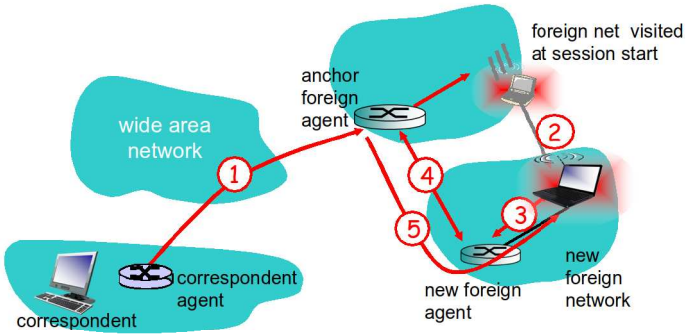
Mobility via direct routing: comments

- overcome triangle routing problem
- *non-transparent to correspondent:* correspondent must get care-of-address from home agent
  - what if mobile changes visited network?



Accommodating mobility with direct routing

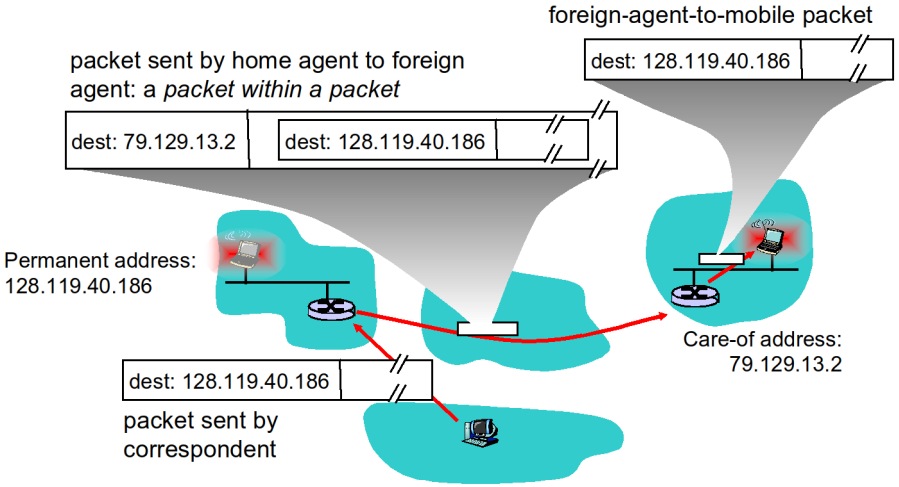
- anchor foreign agent: FA in first visited network
- data always routed first to anchor FA
- when mobile moves: new FA arranges to have data forwarded from old FA (chaining)



Mobile IP

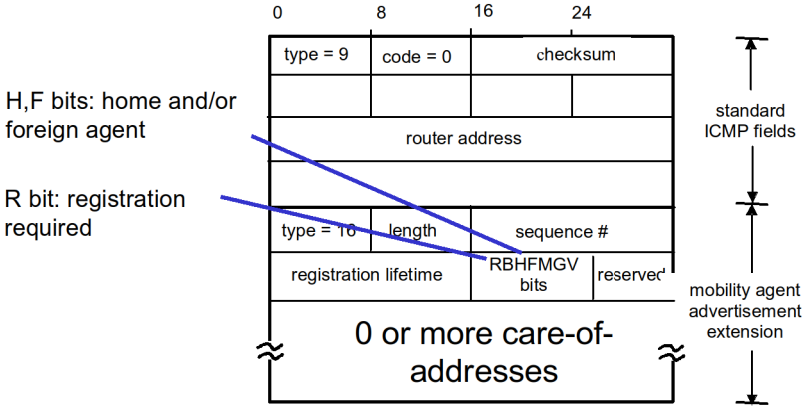
- RFC 3344
- has many features we've seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - indirect routing of datagrams
  - agent discovery
  - registration with home agent

Mobile IP: indirect routing

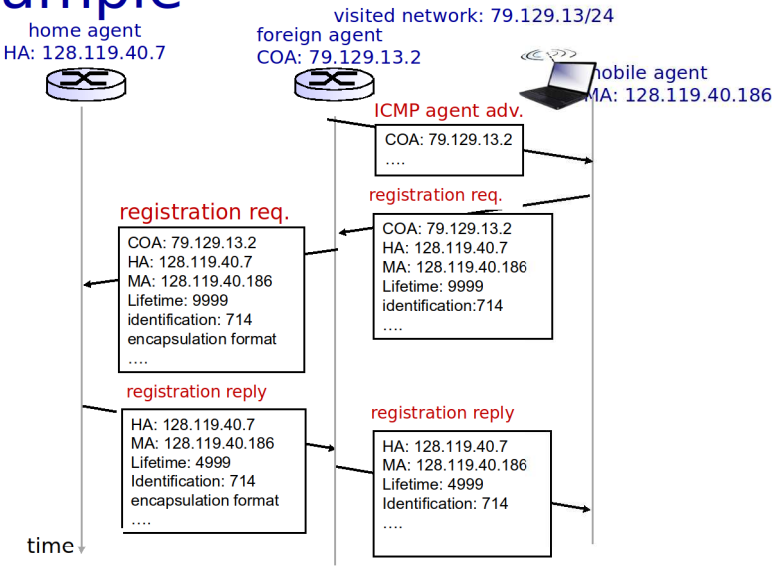


Mobile IP: agent discovery

- *agent advertisement:* foreign/home agents advertise service by broadcasting ICMP messages (typefield = 9)



Mobile IP: registration example



Chapter 1: introduction

our goal:

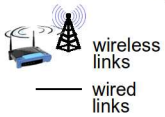
- get "feel" and terminology
- more depth, detail *later* in course
- approach:
  - use Internet as example

overview:

- what's the Internet?
- what's a protocol?
- network edge; hosts, access net, physical media
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput
- security
- protocol layers, service models
- history

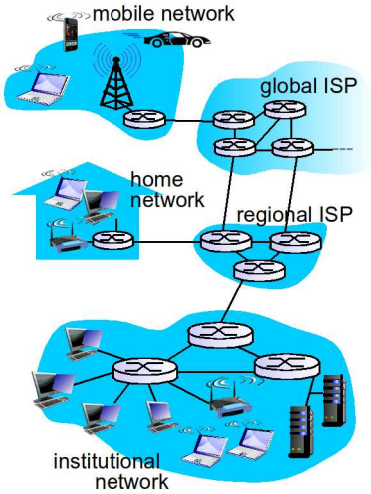
What's the Internet: "nuts and bolts" view

- millions of connected computing devices:
  - *hosts* = *end systems*
  - running *network apps*



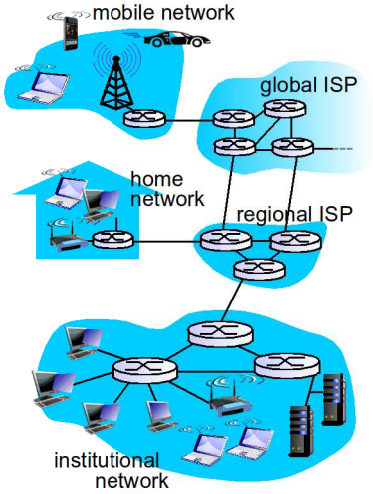
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate: *bandwidth*

- *Packet switches:* forward packets (chunks of data)
  - *routers* and *switches*



What's the Internet: "nuts and bolts" view

- *Internet:* "network of networks"
  - Interconnected ISPs
- *protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- *Internet standards*
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



What's a protocol?

human protocols:

- "what's the time?"
- "I have a question"
- introductions

… specific msgs sent

… specific actions taken when msgs received, or other events

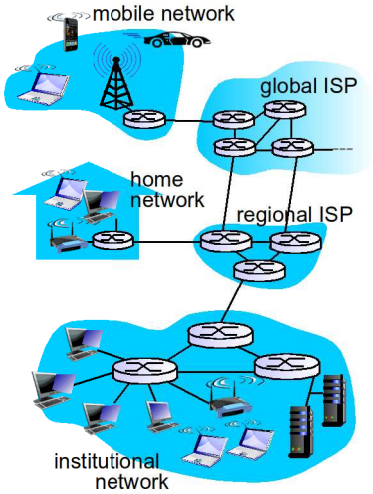
network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt

A closer look at network structure:

- *network edge:*
  - hosts: clients and servers
  - servers often in data centers
- *access networks, physical media:* wired, wireless communication links
- *network core:*
  - interconnected routers
  - network of networks



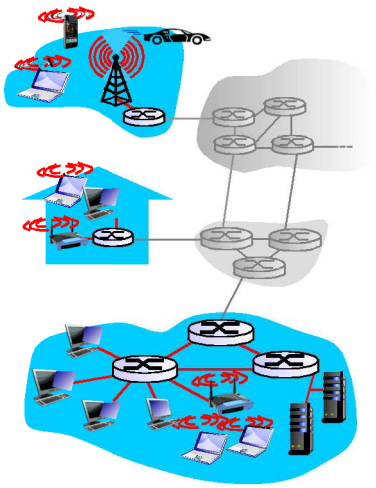
Access networks and physical media

Q: How to connect end systems to edge router?

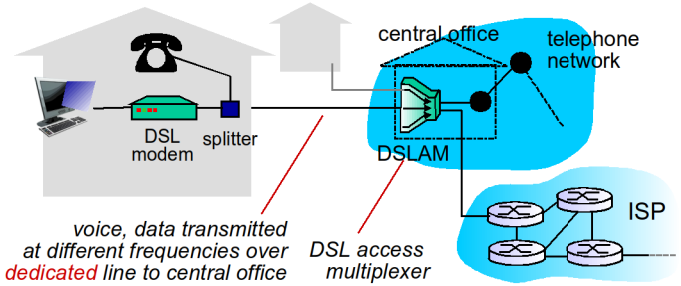
- residential access nets
- institutional access networks (school, company)
- mobile access networks

keep in mind:

- bandwidth (bits per second) of access network?
- shared or dedicated?

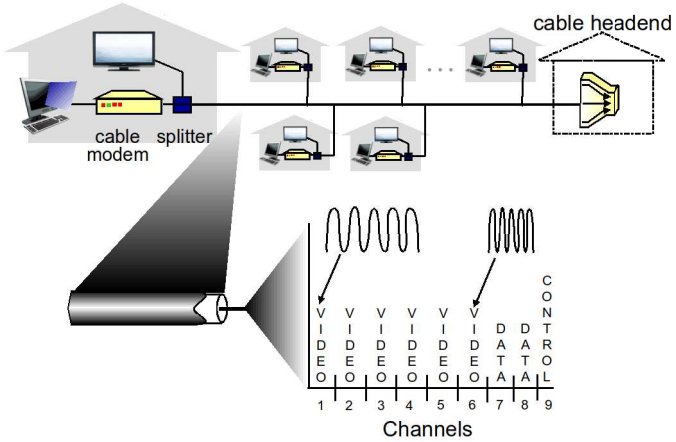


Access net: digital subscriber line (DSL)



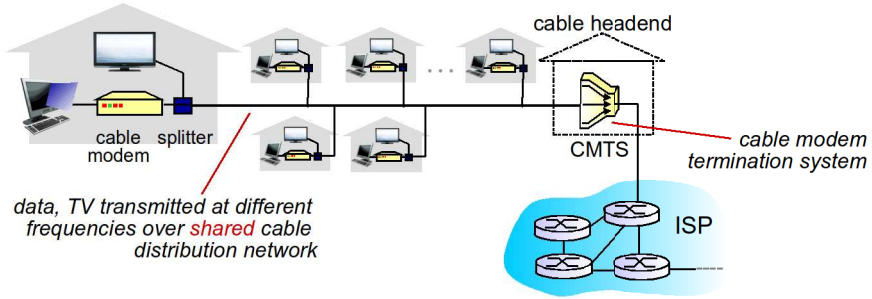
- use *existing* telephone line to central office DSLAM
  - data over DSL phone line goes to Internet
  - voice over DSL phone line goes to telephone net
- < 2.5 Mbps upstream transmission rate (typically < 1 Mbps)
- < 24 Mbps downstream transmission rate (typically < 10 Mbps)

Access net: cable network



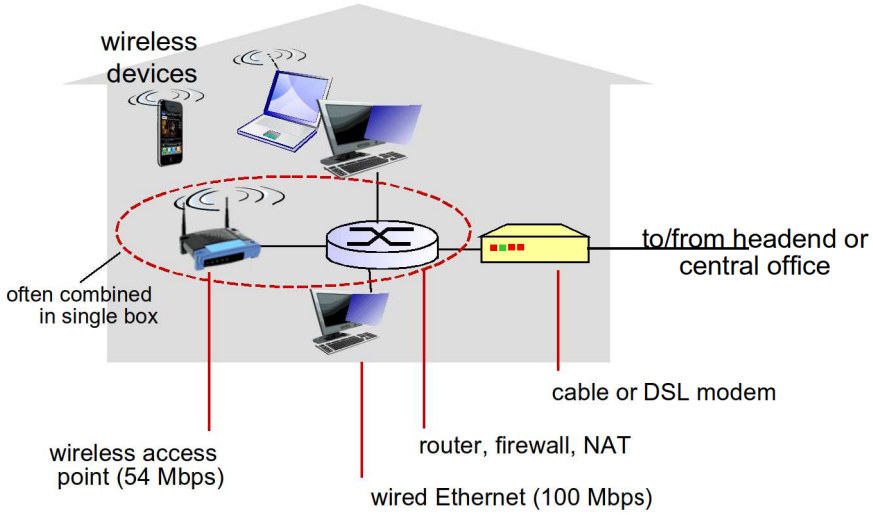
equency division multiplexing: different channels transmitte different frequency bands

Access net: cable network

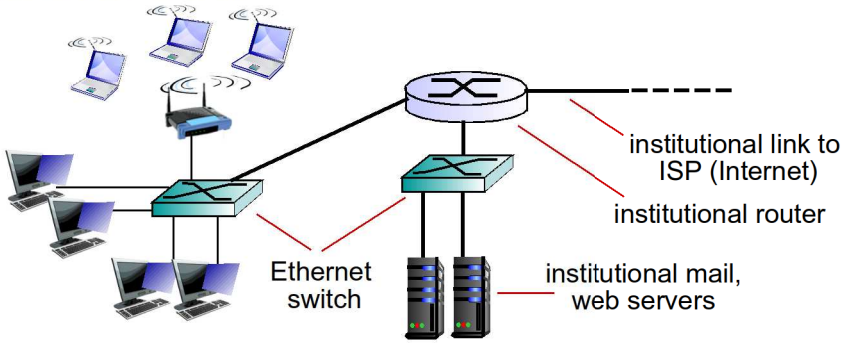


- HFC: hybrid fiber coax
  - asymmetric: up to 30Mbps downstream transmission rate, 2 Mbps upstream transmission rate
- network of cable, fiber attaches homes to ISP router
  - homes *share access network* to cable headend
  - unlike DSL, which has dedicated access to central office

Access net: home network



Enterprise access networks (Ethernet)



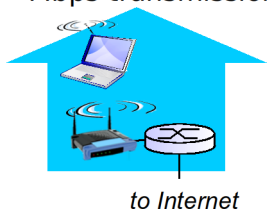
- typically used in companies, universities, etc
- 10 Mbps, 100Mbps, 1Gbps, 10Gbps transmission rates
- today, end systems typically connect into Ethernet switch

Wireless access networks

- shared *wireless* access network connects end system to router
  - via base station aka “access point”

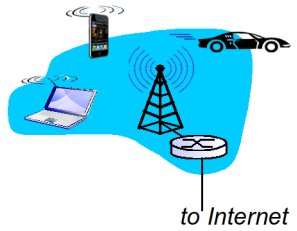
wireless LANs:

- within building (100 ft)
- 802.11b/g (WiFi): 11, 54 Mbps transmission rate



wide-area wireless access

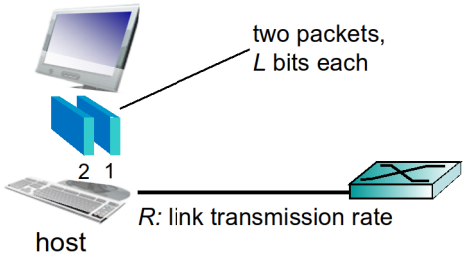
- provided by telco (cellular) operator, 10’s km
- between 1 and 10 Mbps
- 3G, 4G: LTE



Host: sends *packets* of data

host sending function:

- takes application message
- 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*



$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

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, e.g., radio

twisted pair (TP)

- two insulated copper wires
  - Category 5: 100 Mbps, 1 Gpbs Ethernet
  - Gbps

Physical media: coax, fiber

coaxial cable:

- two concentric copper conductors
- bidirectional
- broadband:
  - multiple channels on cable

fiber optic cable:

- glass fiber carrying light pulses, each pulse a bit
- high-speed operation:
  - high-speed point-to-point transmission (e.g., 10's-100's Gpbs transmission rate)
- low error rate:
  - repeaters spaced far apart
  - immune to electromagnetic noise

Physical media: radio

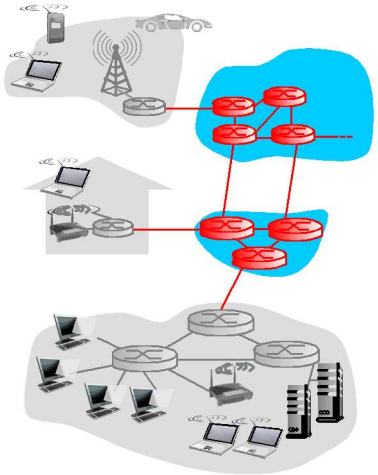
- signal carried in electromagnetic spectrum
- no physical "wire"
- bidirectional
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

radio link types:

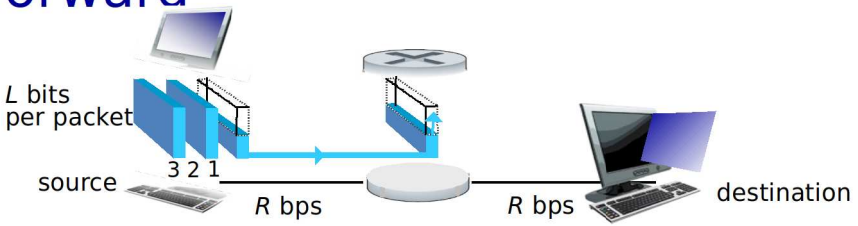
- terrestrial microwave
  - e.g. up to 45 Mbps channels
- LAN (e.g., WiFi)
  - 11Mbps, 54 Mbps
- wide-area (e.g., cellular)
  - 3G cellular: ~ few Mbps
- satellite
  - Kbps to 45Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low altitude

The network core

- mesh of interconnected routers
- packet-switching: hosts break application-layer messages into *packets*
  - forward packets from one router to the next, across links on path from source to destination
  - each packet transmitted at full link capacity



Packet-switching: store-and-forward

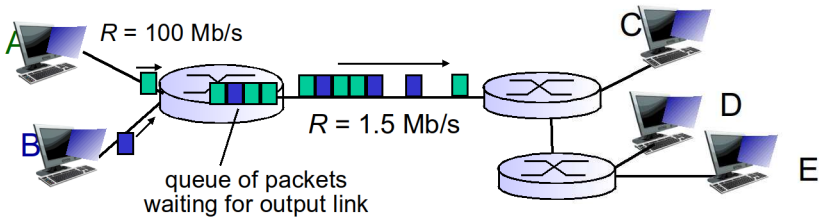


- takes L/R seconds to transmit (push out) L-bit packet into link at R bps
- *store and forward:* entire packet must arrive at router before it can be transmitted on next link
- end-end delay = $2L/R$ (assuming zero propagation delay)

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

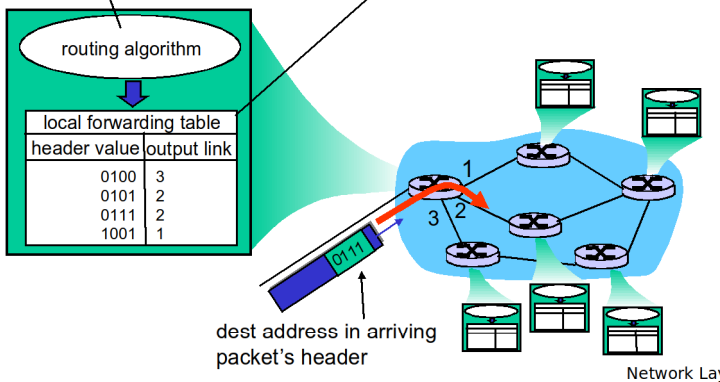


queuing and loss:
- If arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
  - packets will queue, wait to be transmitted on link
  - packets can be dropped (lost) if memory (buffer) fills up

Two key network-core functions

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

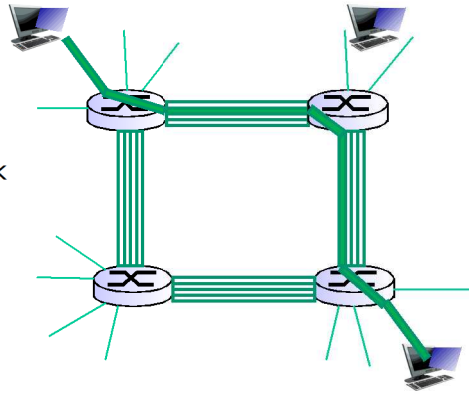
forwarding: move packets from router's input to appropriate router output



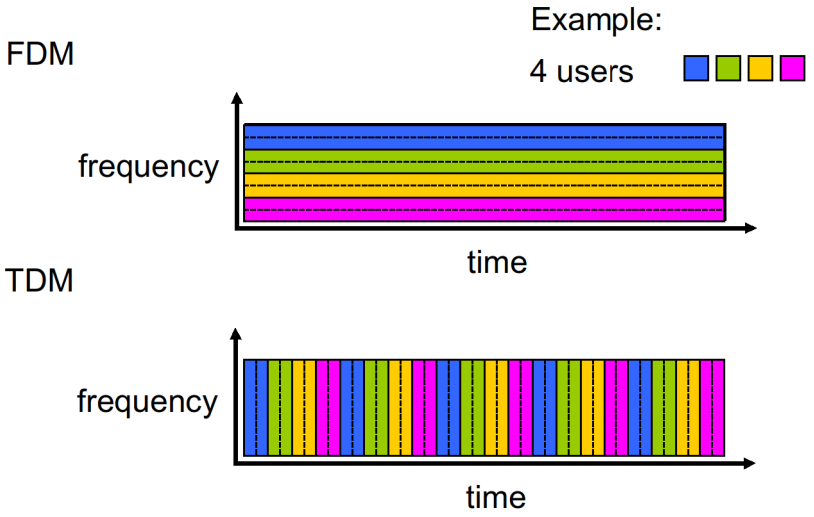
Alternative core: circuit switching

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

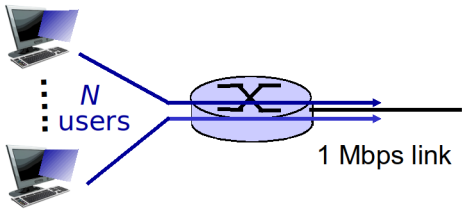


Packet switching versus circuit switching

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 ?

Packet switching versus circuit switching

is packet switching a "slam dunk winner?"

- great for bursty data
  - resource sharing
  - simpler, no call setup
- excessive congestion possible: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- *Q:* How to provide circuit-like behavior?
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem (chapter 7)

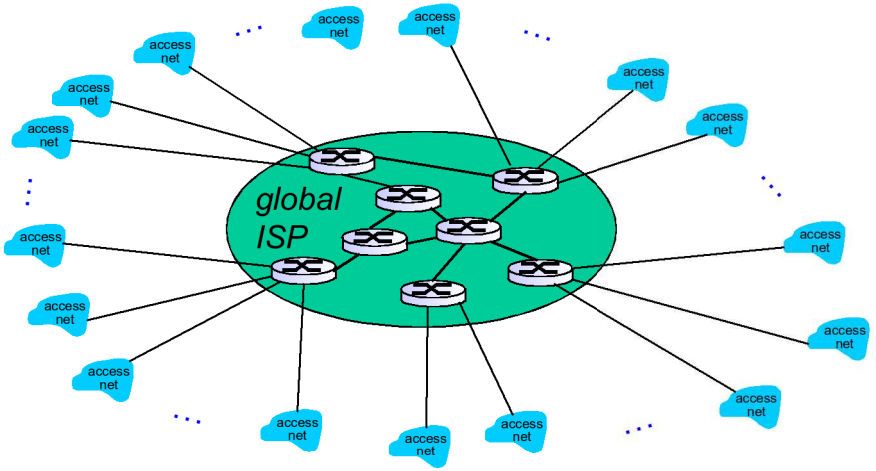
Q: human analogies of reserved resources (circuit switching) versus on-demand allocation (packet-switching)?

Internet structure: network of networks

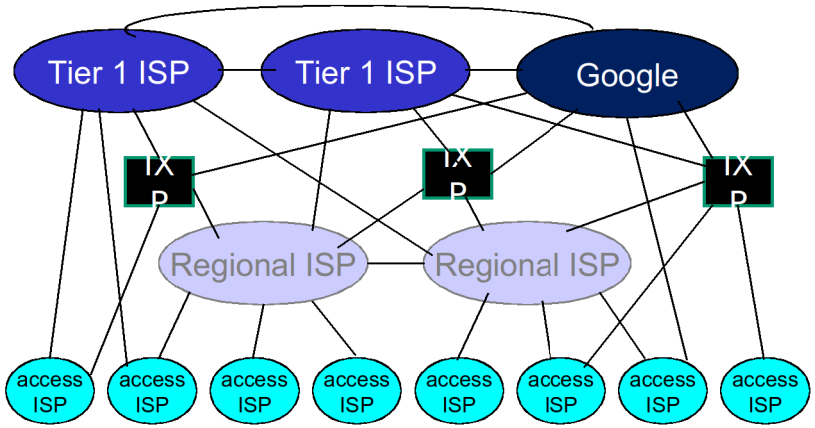
- End systems connect to Internet via access ISPs (Internet Service Providers)
  - Residential, company and university ISPs
- Access ISPs in turn must be interconnected.
  - So that any two hosts can send packets to each other
- Resulting network of networks is very complex
  - Evolution was driven by economics and national policies
- Let's take a stepwise approach to describe current Internet structure

Internet structure: network of networks

Option: connect each access ISP to a global transit ISP?
Customer and provider ISPs have economic agreement.



Internet structure: network of networks

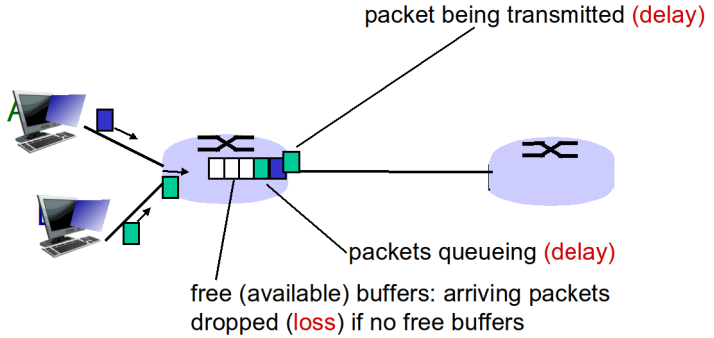


- at center: small # of well-connected large networks
  - "tier-1" commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
  - content provider network (e.g, Google): private network that connects it data centers to Internet, often bypassing tier-1, regional ISPs

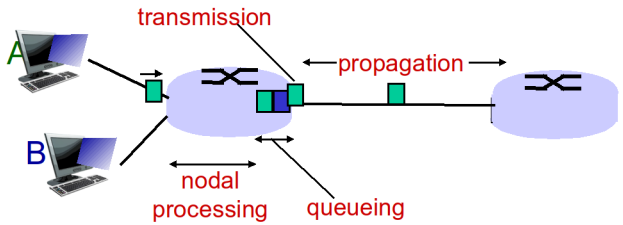
How do loss and delay occur?

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

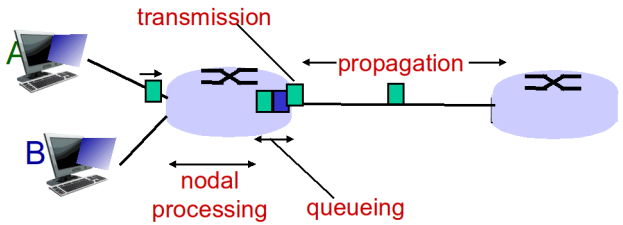


$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

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

Four sources of packet delay



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

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 (~2x10⁸ m/sec)
- $d_{prop} = d/s$

* Check out the Java applet for an interactive animation on trans vs. prop delay

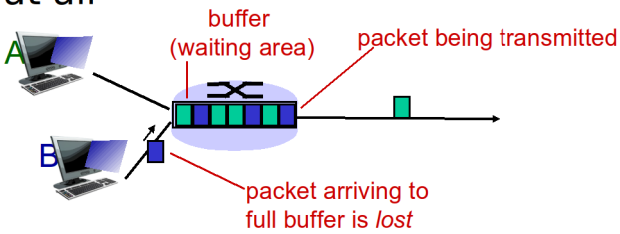
Queueing delay

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

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

Packet loss

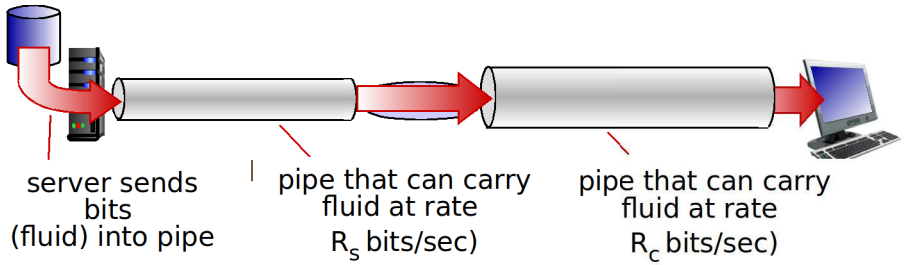
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



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

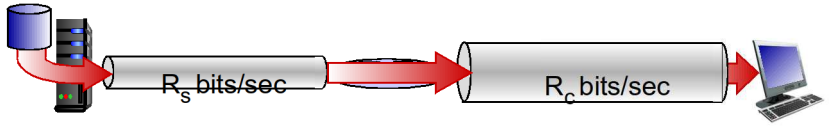
Throughput

- *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

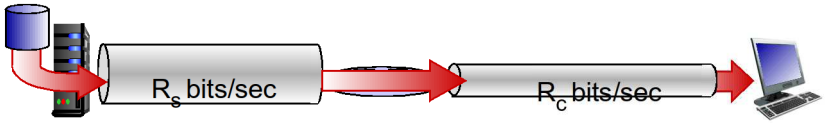


Throughput (more)

- $R_s < R_c$ What is average end-end throughput?



- $R_s > R_c$ What is average end-end throughput?

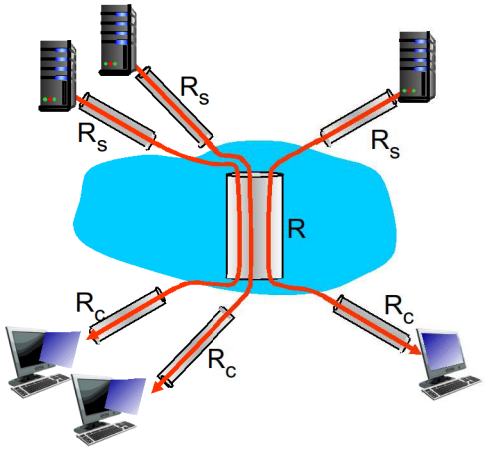


bottleneck link

lir d path that constrains end-end throughput

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

Protocol "layers"

Networks are complex, with many "pieces":

- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:

is there any hope of *organizing* structure of network?

.... or at least our discussion of networks?

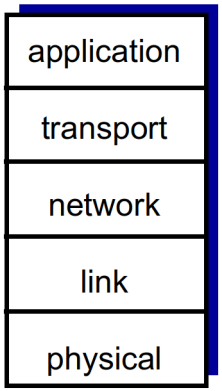
Why layering?

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
- layering considered harmful?

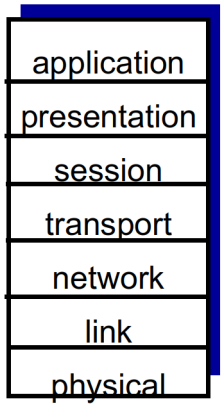
Internet protocol stack

- *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"

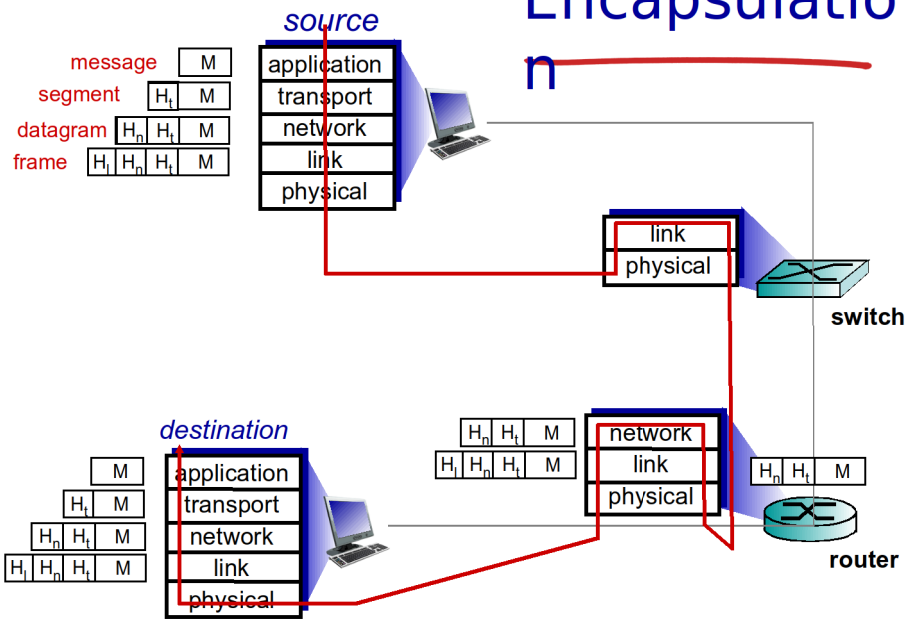


ISO/OSI reference model

- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application
  - needed?



Encapsulation



Network security

- field of network security:
  - how bad guys can attack computer networks
  - how we can defend networks against attacks
  - how to design architectures that are immune to attacks
- Internet not originally designed with (much) security in mind
  - *original vision:* "a group of mutually trusting users attached to a transparent network" ☺
  - Internet protocol designers playing "catch-up"
  - security considerations in all layers!

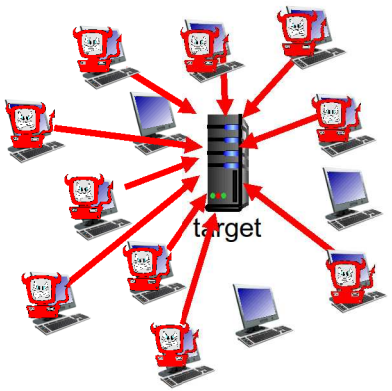
Bad guys: put malware into hosts via Internet

- malware can get in host from:
  - *virus:* self-replicating infection by receiving/executing object (e.g., e-mail attachment)
  - *worm:* self-replicating infection by passively receiving object that gets itself executed
- spyware malware can record keystrokes, web sites visited, upload info to collection site
- infected host can be enrolled in botnet, used for spam. DDoS attacks

Bad guys: attack server, network infrastructure

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

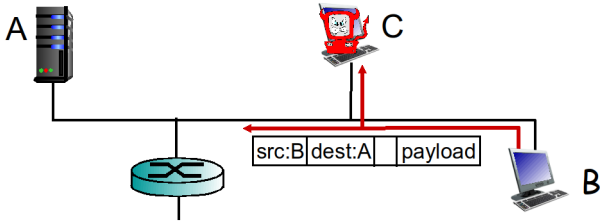
1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



Bad guys can sniff packets

packet "sniffing":

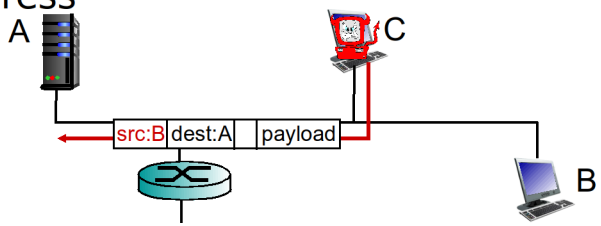
- broadcast media (shared ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by



- wireshark software used for end-of-chapter labs is a (free) packet-sniffer

Bad guys can use fake addresses

IP spoofing: send packet with false source address

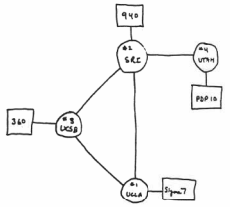


... lots more on security (throughout, Chapter 8)

Internet history

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- 1972:
  - ARPAnet public demo
  - NCP (Network Control Protocol) first host-host protocol
  - first e-mail program
  - ARPAnet has 15 nodes



THE ARPA NETWORK

Internet history

1972-1980: Internetworking, new and proprietary nets

- 1970: ALOHAnet satellite network in Hawaii
- 1974: Cerf and Kahn - architecture for interconnecting networks
- 1976: Ethernet at Xerox PARC
- late70's: proprietary architectures: DECnet, SNA, XNA
- late 70's: switching fixed length packets (ATM precursor)
- 1979: ARPAnet has 200 nodes

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

Internet history

1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: smtp e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: ftp protocol defined
- 1988: TCP congestion control
- new national networks: Csnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

Internet history

1990, 2000's: commercialization, the Web, new apps

- early 1990's: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
  - hypertext [Bush 1945, Nelson 1960's]
  - HTML, HTTP: Berners-Lee
  - 1994: Mosaic, later Netscape
  - late 1990's: commercialization of the Web

late 1990's – 2000's:

- more killer apps: instant messaging, P2P file sharing
- network security to forefront
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

Internet history

2005-present

- ~750 million hosts
  - Smartphones and tablets
- Aggressive deployment of broadband access
- Increasing ubiquity of high-speed wireless access
- Emergence of online social networks:
  - Facebook: soon one billion users
- Service providers (Google, Microsoft) create their own networks
  - Bypass Internet, providing "instantaneous" access to search, emai, etc.
- E-commerce, universities, enterprises running their services in "cloud" (eg, Amazon EC2)

Chapter 7: Network Security

confidentiality: only sender, intended receiver should "understand" message contents
- sender encrypts message
- receiver decrypts message

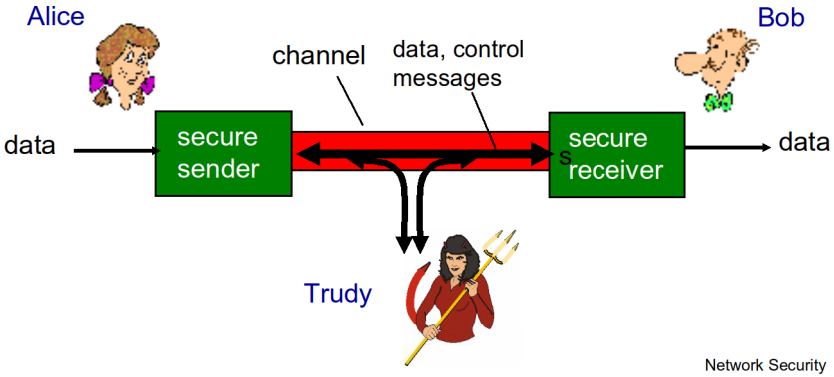
authentication: sender, receiver want to confirm identity of each other

message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

access and availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

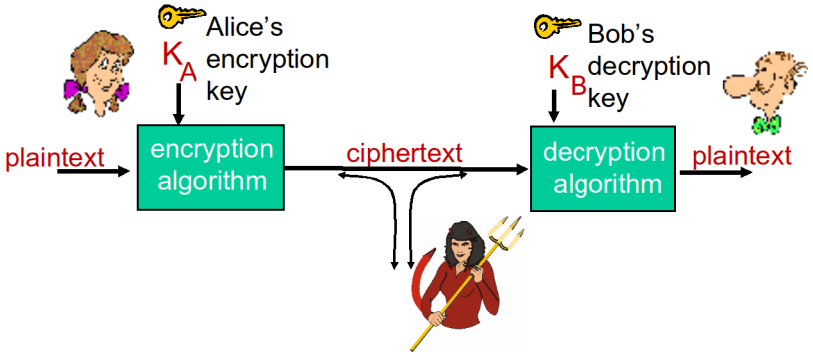
- … well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: A lot! See section 1.6
- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

The language of cryptography



m plaintext message

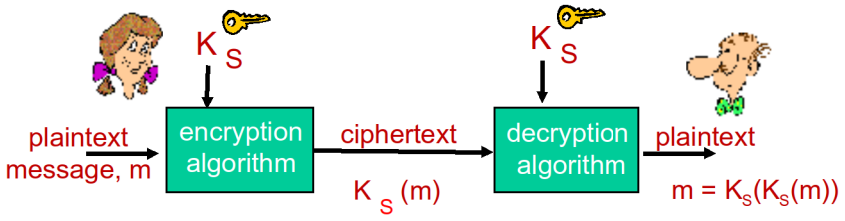
$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis
- known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

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

Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

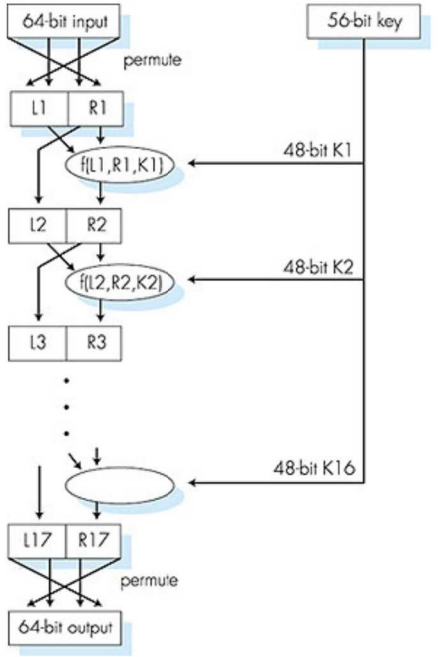
Symmetric key crypto: DES

DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation



AES: Advanced Encryption Standard

- symmetric-key NIST standard, replacied DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Public Key Cryptography

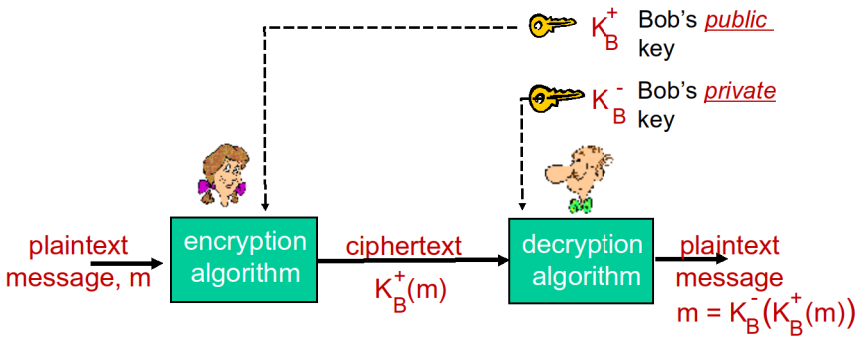
symmetric key crypto

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key crypto

- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public key cryptography



Public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that
$$K_B^-(K_B^+(m)) = m$$

2. given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:
  [(a mod n) + (b mod n)] mod n = (a+b) mod n
  [(a mod n) - (b mod n)] mod n = (a-b) mod n
  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus
  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number.

example:

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

1. choose two large prime numbers *p, q.* (e.g., 1024 bits each)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. choose e (with $e<n$) that has no common factors with z (e, z are "relatively prime").
4. choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5. *public* key is (n,e). *private* key is (n,d).

K_B^+ K_B^-

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above
1. to encrypt message m ($<n$), compute

$$c = m^e \bmod n$$

2. to decrypt received bit pattern, c, compute

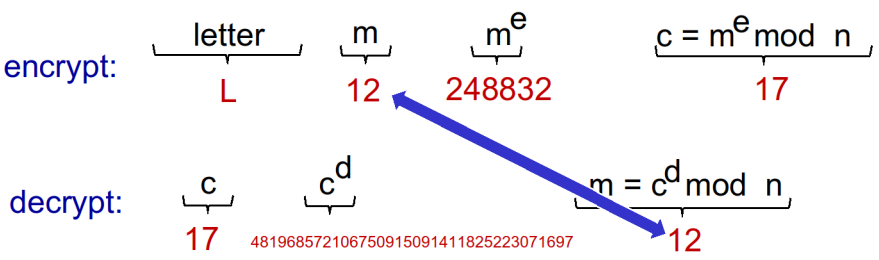
$$m = c^d \bmod n$$

magic happens! $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

RSA example:

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$.
$e=5$ (so e, z relatively prime).
$d=29$ (so $ed-1$ exactly divisible by z).

encrypting 8-bit messages.



Why does RSA work?

- must show that $c^d \bmod n = m$ where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where n= pq and z = (p-1)(q-1)
- thus,

$$c^d \bmod n = (m^e \bmod n)^d \bmod n$$
$$= m^{ed} \bmod n$$
$$= m^{(ed \bmod z)} \bmod n$$
$$= m^1 \bmod n$$
$$= m$$

RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

use public key first, followed by private key

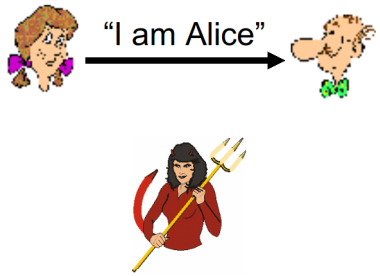
use private key first, followed by public key

result is the same!

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

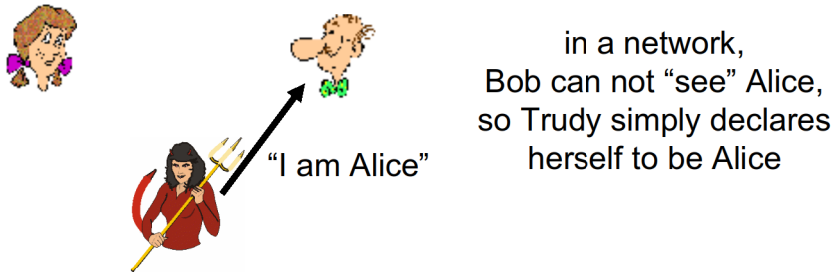


Failure scenario??

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"



in a network, Bob can not "see" Alice, so Trudy simply declares herself to be Alice

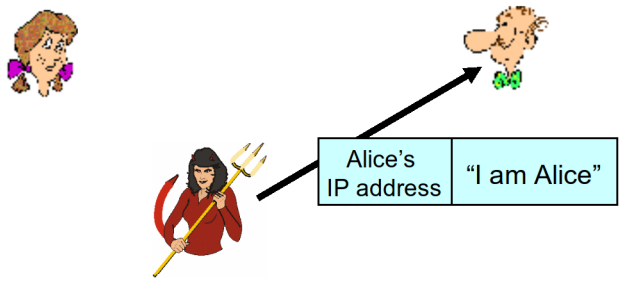
Authentication: another try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

Failure scenario??

Authentication: another try

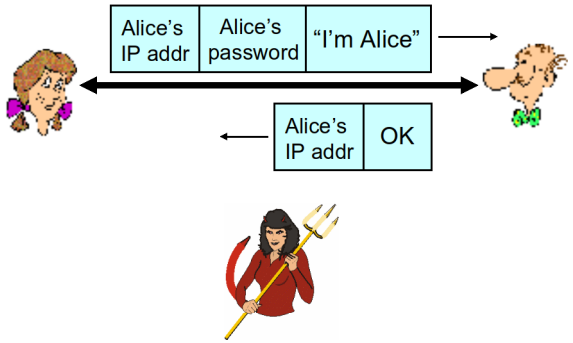
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Trudy can create a packet "spoofing" Alice's address

Authentication: another try

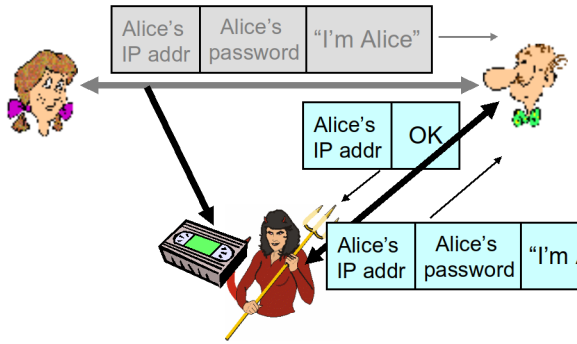
Protocol ap3.0: Alice says "I am Alice" and sends he secret password to "prove" i



Failure scenario??

Authentication: another try

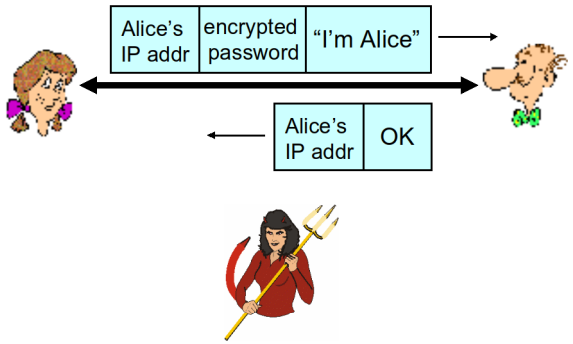
Protocol ap3.0: Alice says "I am Alice" and sends he secret password to "prove" i



playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: yet another try

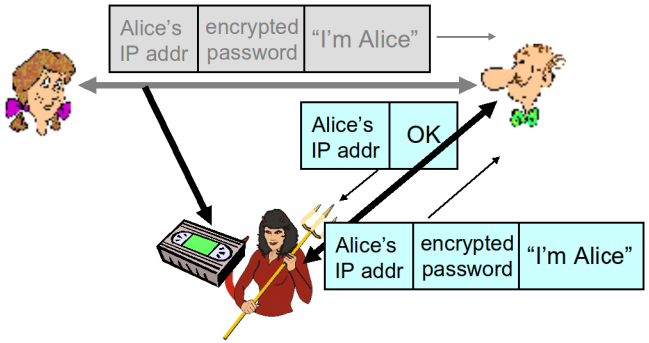
Protocol ap3.1: Alice says "I am Alice" and sends he *encrypted* secret password to "prove" i



Failure scenario??

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends he
encrypted secret password to "prove" i



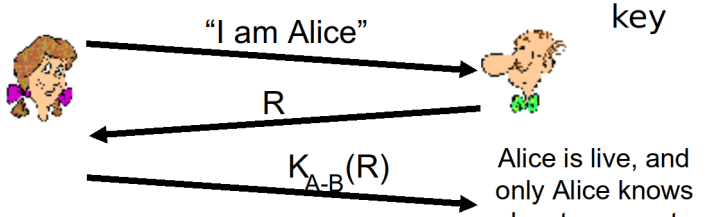
record and playback *still* works!

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

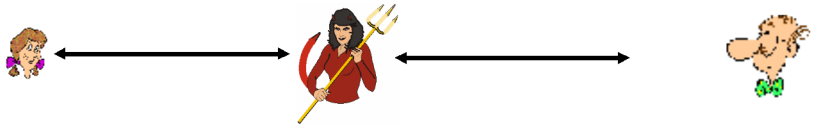


Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Failures, drawbacks?

security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Trudy receives all messages as well!

Digital signatures

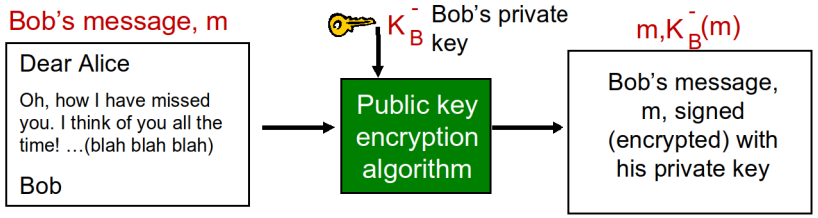
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- *verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital signatures

simple digital signature for message m:

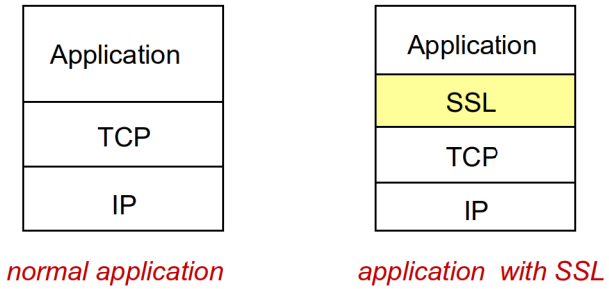
- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



SSL: Secure Sockets Layer

- widely deployed security protocol
  - supported by almost all browsers, web servers
  - https
  - billions $/year over SSL
- mechanisms: [Woo 1994], implementation: Netscape
- variation -TLS: transport layer security, RFC 2246
- provides
  - *confidentiality*
  - *integrity*
  - *authentication*
- original goals:
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant
- available to all TCP applications
  - secure socket interface

SSL and TCP/IP



- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

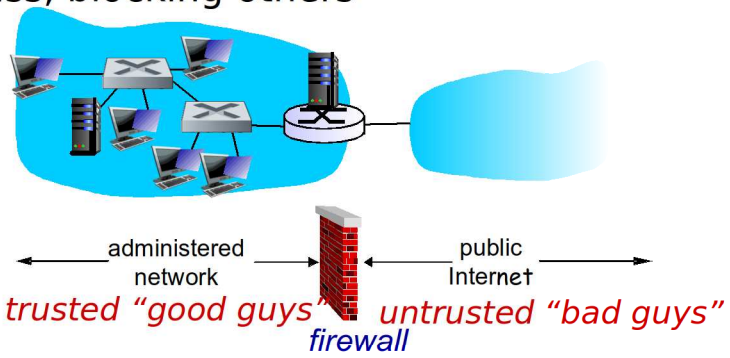
Toy SSL: a simple secure channel

- *handshake:* Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation:* Alice and Bob use shared secret to derive set of keys
- *data transfer:* data to be transferred is broken up into series of records
- *connection closure:* special messages to securely close connection

Firewalls

firewall

isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others



Firewalls: why

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 homepage with something else

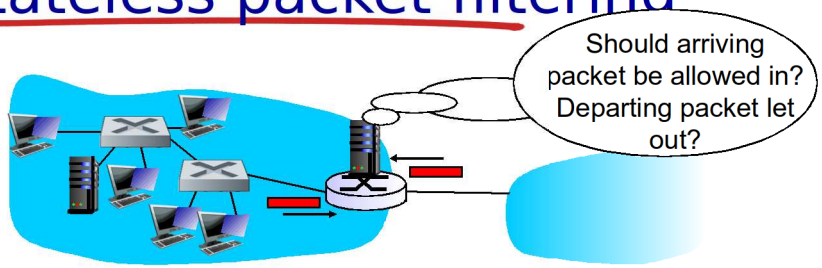
allow only authorized access to inside network

- set of authenticated users/hosts

three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

Stateless packet filtering



- internal network connected to Internet via *router firewall*
- router *filters packet-by-packet,* decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

Stateless packet filtering: example

- *example 1:* block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23
  - *result:* all incoming, outgoing UDP flows and telnet connections are blocked
- *example 2:* block inbound TCP segments with ACK=0.
  - *result:* prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

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

Stateful packet filtering

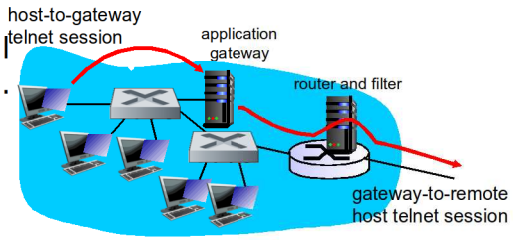
- *stateless packet filter:* heavy handed tool
  - admits packets that "make no sense," e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- *stateful packet filter:* track status of every TCP connection
  - track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets "makes sense"
  - timeout inactive connections at firewall: no longer admit packets

Application gateways

- filter packets on application data as well as on IP/TCP/UDP fields.
- *example:* allow select internal users to telnet outside



1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway.

Limitations of firewalls, gateways

- *IP spoofing:* router can't know if data "really" comes from claimed source
- if multiple app's. need special treatment, each has own app. gateway
- client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP
- *tradeoff:* degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks