

Part 0: Networking Review

Goals:

- ❑ review key topics from intro networks course
 - equalize backgrounds
 - identify remedial work
 - ease into course

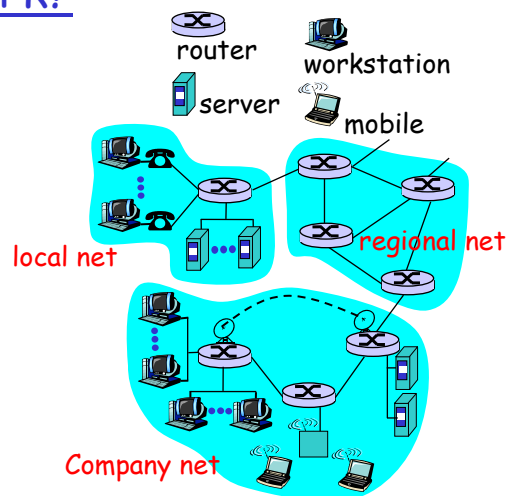
Overview:

- ❑ overview
- ❑ error control
- ❑ flow control
- ❑ congestion control
- ❑ routing
- ❑ LANs
- ❑ addressing
- ❑ synthesis:
 - "a day in the life"
 - control timescales

0-1

What is a network?

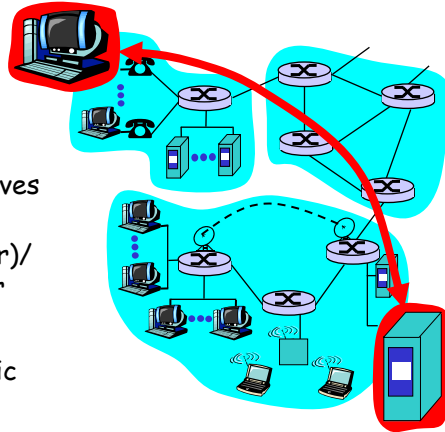
- ❑ **network edge:** applications and hosts
- ❑ **network core:**
 - routers
 - network of networks
- ❑ **access networks, physical media:** communication links
 - fiber, copper, radio, ...



0-2

The network edge:

- **end systems (hosts):**
 - run application programs
 - e.g., WWW, email
 - at "edge of network"
- **client/server model**
 - client host requests, receives service from server
 - e.g., WWW client (browser)/server; email client/server
- **peer-peer model:**
 - host interaction symmetric
 - e.g.: Gnutella, KaZaA



0-3

Network edge: connection-oriented service

Goal: data transfer between end sys.

- **handshaking:** setup (prepare for) data transfer ahead of time
 - Hello, hello back human protocol
 - **set up "state"** in two communicating hosts
- **TCP - Transmission Control Protocol**
 - Internet's connection-oriented service

TCP service [RFC 793]

- **reliable, in-order byte-stream data transfer**
 - loss: acknowledgements and retransmissions
- **flow control:**
 - sender won't overwhelm receiver
- **congestion control:**
 - senders "slow down sending rate" when network congested

0-4

Network edge: connectionless service

Goal: data transfer

between end systems

- same as before!

- **UDP** - User Datagram Protocol [RFC 768]: Internet's connectionless service

- unreliable data transfer
- no flow control
- no congestion control

App's using TCP:

- HTTP (WWW), FTP (file transfer), Telnet (remote login), SMTP (email)

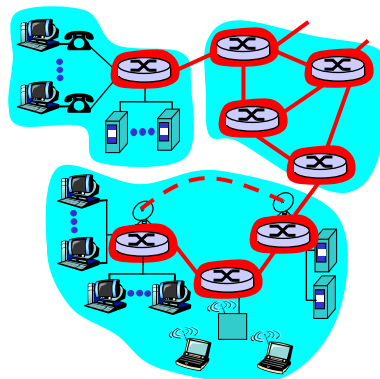
App's using UDP:

- streaming media, teleconferencing, Internet telephony

0-5

The Network Core

- mesh of interconnected routers
- the fundamental question: how is data transferred through net?
 - **circuit switching:** dedicated circuit per call: telephone net
 - **packet switching:** data sent thru net in discrete "chunks"

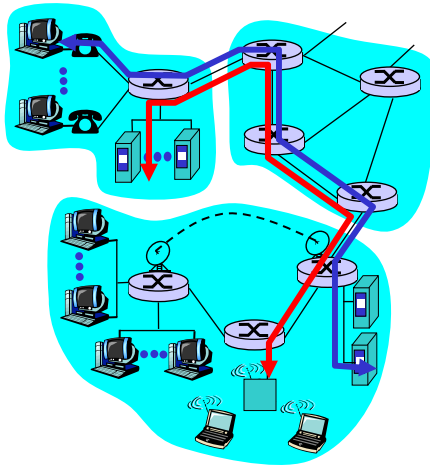


0-6

Network Core: Circuit Switching

End-end resources reserved for "call"

- ❑ link bandwidth, switch capacity
- ❑ dedicated resources: no sharing
- ❑ circuit-like (guaranteed) performance
- ❑ call setup required



0-7

Network Core: Packet Switching

each end-end data stream divided into *packets*

- ❑ user A, B packets *share* network resources
- ❑ each packet uses full link bandwidth
- ❑ resources used *as needed*

~~Bandwidth division into "pieces"
Dedicated allocation
Resource reservation~~

resource contention:

- ❑ aggregate resource demand can exceed amount available
- ❑ congestion: packets queue, wait for link use
- ❑ store and forward: packets move one hop at a time
 - transmit over link
 - wait turn at next link

0-8

Network core: routing

Goal: move data among routers from source to dest.

datagram packet network:

- *destination address* determines next hop
- routes may change during session
- analogy: driving, asking directions
- No notion of call state

virtual circuit network:

- packet carries tag, tag determines next hop
- fixed path (for call) determined at *call setup time*
- routers maintain little per-call state; resources not allocated

circuit-switched network:

- call allocated time slots of bandwidth at each link
- fixed path (for call) determined at call setup
- switches maintain lots of per call state (what?): resource allocation

0-9

Packet switching vs. circuit switching:

□ circuit-switching:

- "reliability" - no congestion
- in-order data delivery

□ packet switching:

- better bandwidth use
- smaller amount of state in resources
 - good: less control-plane processing resources along the way
 - bad: More data-plane (address lookup) processing

□ failure modes (routers/links down):

- packet switching routing reconfigures at sub-second timescale;
- circuit-switching: more complex recovery - need to involve all (downstream) switches on path

0-10

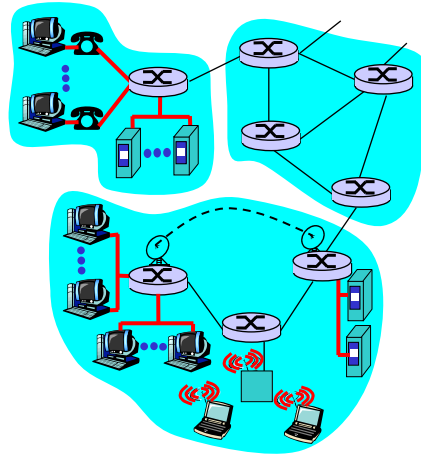
Access networks and physical media

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?

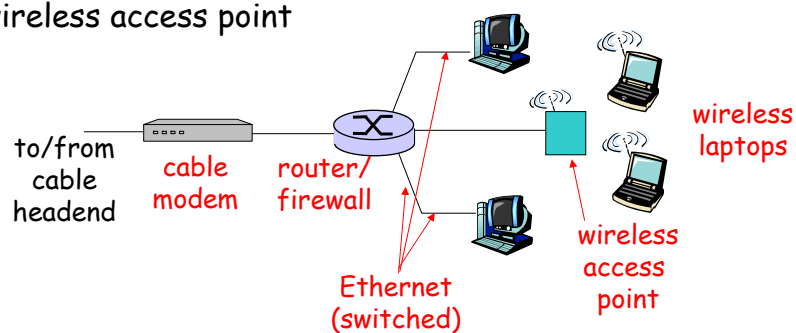


0-11

Example access net: home network

Typical home network components:

- ❑ ADSL or cable modem
- ❑ router/firewall
- ❑ Ethernet
- ❑ wireless access point



0-12

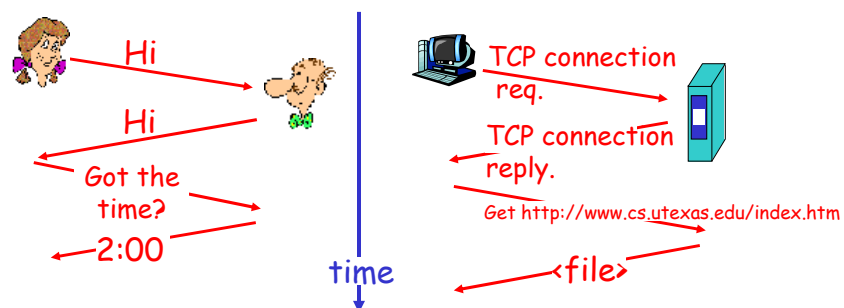
What's a network protocol?

- means (algorithm) for exchanging information
- "understanding" between two (or more) entities about how to exchange information
- shared (agreed upon, standardized) set of rules for communication
 - "how" you send/receive data
 - format of data exchanged

0-13

What's a protocol?

a human protocol and a computer network protocol:



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

0-14

So we have seen "pieces" of network

- ❑ edge, core, links
- ❑ protocols

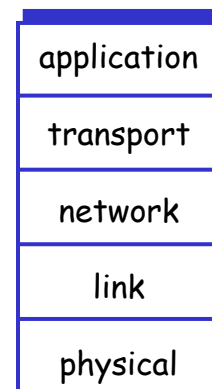
How do we talk about "structure" of network and its architecture?

- ❑ layered architecture
 - structure allows identification, relationship of complex system's pieces: layered **reference model** for discussion
 - layer N builds on services provided by layer N-1
 - Layer N provides service to layer N+1
- ❑ physical topology, interconnection

0-15

Internet protocol stack

- ❑ **application**: supporting network applications
 - ftp, smtp, http
- ❑ **transport**: host-host data transfer
 - tcp, udp
- ❑ **network**: routing of datagrams from source to destination
 - ip, routing protocols
- ❑ **link**: data transfer between neighboring network elements
 - ppp, ethernet
- ❑ **physical**: bits "on the wire"

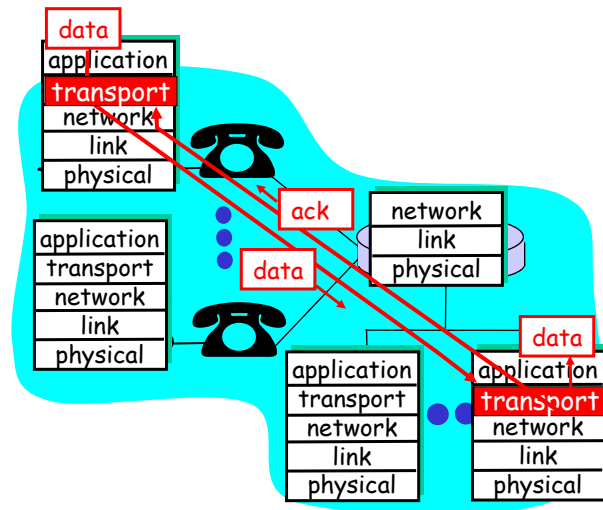


0-16

Layering: logical communication

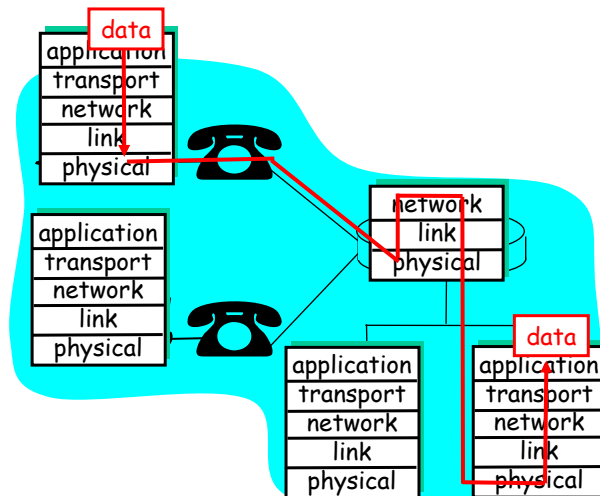
E.g.: transport

- ❑ take data from app
- ❑ add addressing, reliability check info to form "datagram"
- ❑ send datagram to peer
- ❑ wait for peer to ack receipt
- ❑ analogy: post office



0-17

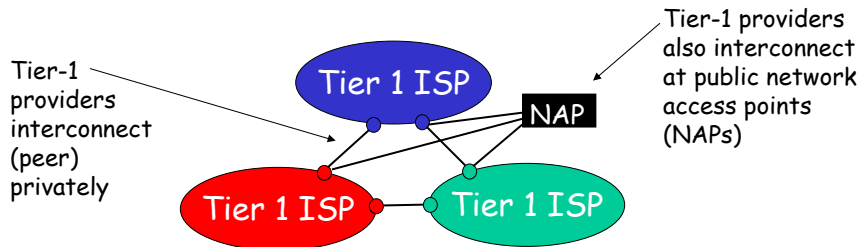
Layering: physical communication



0-18

Internet structure: network of networks

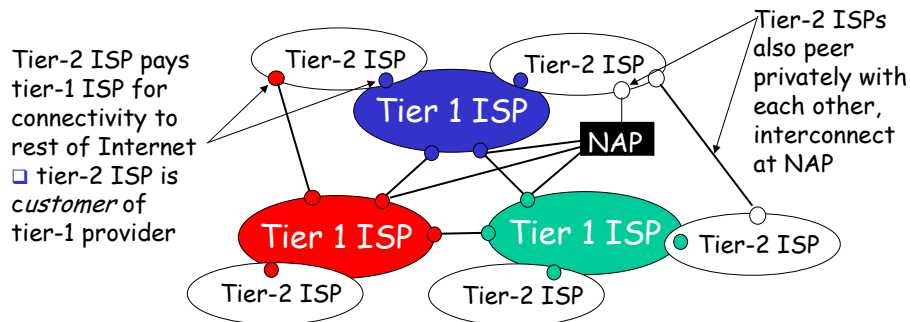
- roughly hierarchical
- **at center: "tier-1" ISPs** (e.g., UUNet, BBN/Genuity, Sprint, AT&T), national/international coverage
 - treat each other as equals



0-19

Internet structure: network of networks

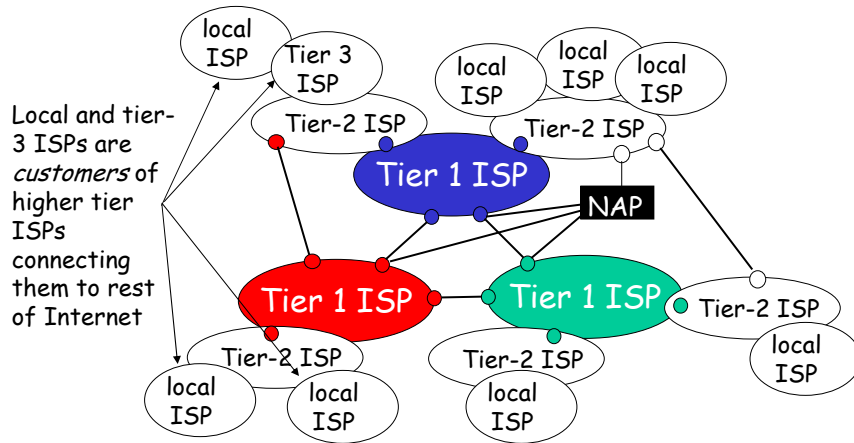
- **"Tier-2" ISPs: smaller (often regional) ISPs**
 - Connect to one or more tier-1 ISPs, possibly other tier-2 ISPs



0-20

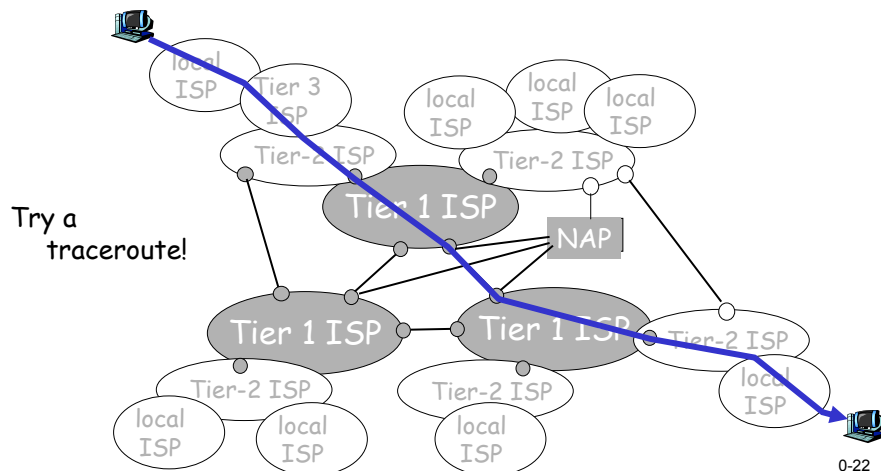
Internet structure: network of networks

- "Tier-3" ISPs and local ISPs
 - last hop ("access") network (closest to end systems)



Internet structure: network of networks

- a packet passes through many networks!



Part 0: Networking Review

Goals:

- review key topics from intro networks course
 - equalize backgrounds
 - identify remedial work
 - ease into course

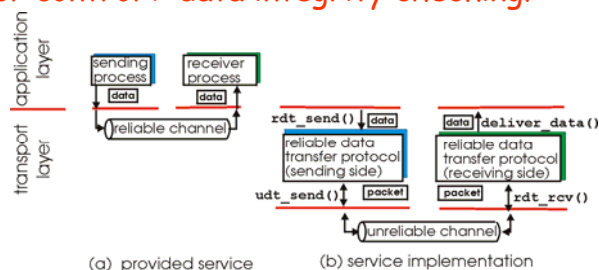
Overview:

- overview
- error control
- flow control
- congestion control
- routing
- LANs
- addressing
- synthesis:
 - "a day in the life"
 - control timescales

0-23

Error control

- reliable point-point communication
 - A generic problem: application-to-application, over path, over link
- what's the error model:
 - bits flipped in packet?
 - packets "lost"?
 - packets delayed or reordered?
- Error control ≠ data integrity checking!



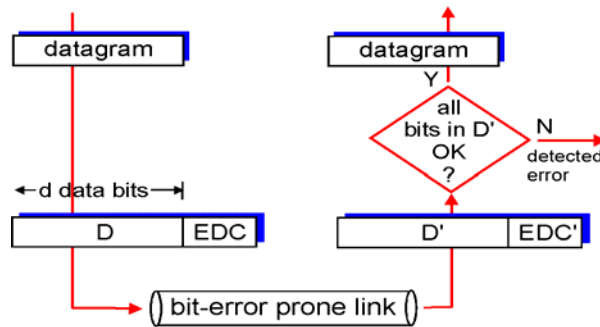
0-24

Bit level 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

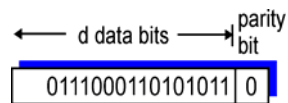


0-25

Parity Checking

Single Bit Parity:

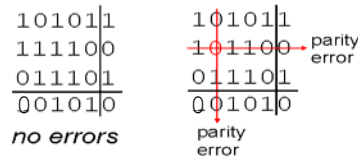
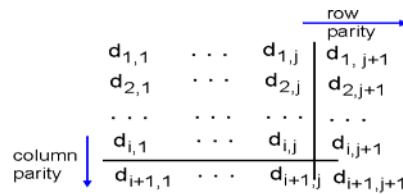
Detect single bit errors



Much more powerful error detection/correction schemes:
Cyclic Redundancy Check (CRC)
- polynomial division modulo 2

Two Dimensional Bit Parity:

Detect and correct single bit errors



Simple form of forward error correction (FEC)

0-26

Internet checksum

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

Sender:

- ❑ treat segment contents as sequence of 16-bit integers
- ❑ checksum: addition of segment contents
 - 1's complement of 1's complement sum
- ❑ 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?

Benefits: easy to compute; can do incremental update; endian-independent

0-27

Recovering from lost packets

- ❑ Why are packets lost: at end system, "within" network
 - Limited storage, discarded in congestion
 - outages: eventually reroute around failure (~sec recovery ties hopefully)
 - dropped at end system e.g., on NIC
- ❑ ARQ: automatic request repeat
 - sender puts sequence numbers on packets (why)
 - receiver positively or negatively acknowledges correct receipt of packet
 - sender starts (logical) timer for each packet, timeout and retransmits

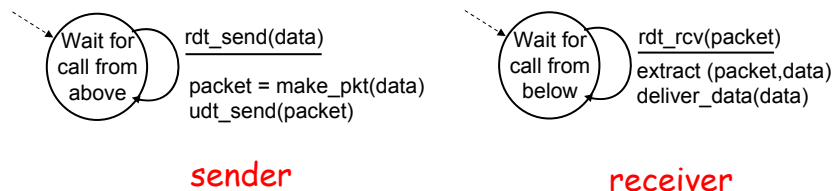
0-28

- Q: any other loss recovery paradigm besides ARQ? What's the tradeoff?
- A: coding!
 - Source coding
 - FEC: forward error correction
 - Hybrid: FEC + ARQ (e.g. FEC for retransmitted pkts)
 - Network coding
 - A hot research area!!
 - Coding granularity
 - Packet-level
 - Bit-level → partial recovery
 - Analog coding

0-29

Rdt1.0: reliable transfer over a reliable channel

- underlying channel perfectly reliable
 - no bit errors
 - no loss of packets
- separate FSMs for sender, receiver:
 - sender sends data into underlying channel
 - receiver read data from underlying channel



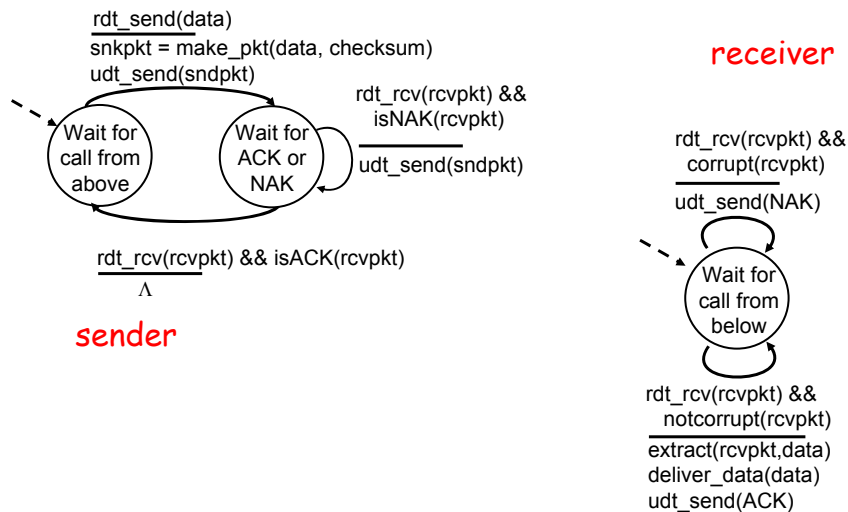
0-30

Rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
 - recall: UDP checksum to detect bit errors
- *the question: how to recover from errors:*
 - *acknowledgements (ACKs):* receiver explicitly tells sender that pkt received OK
 - *negative acknowledgements (NAKs):* receiver explicitly tells sender that pkt had errors
 - sender retransmits pkt on receipt of NAK
 - human scenarios using ACKs, NAKs?
- new mechanisms in rdt2.0 (beyond rdt1.0):
 - error detection
 - receiver feedback: control msgs (ACK,NAK) rcvr->sender

0-31

rdt2.0: FSM specification



0-32

- ❑ Q: Does rdt2.0 work?

0-33

rdt2.0 has a fatal flaw!

What happens if ACK/NAK corrupted?

- ❑ sender doesn't know what happened at receiver!
- ❑ can't just retransmit: possible duplicate

What to do?

- ❑ sender ACKs/NAKs receiver's ACK/NAK? What if sender ACK/NAK lost?
- ❑ retransmit, but this might cause retransmission of correctly received pkt!

Handling duplicates:

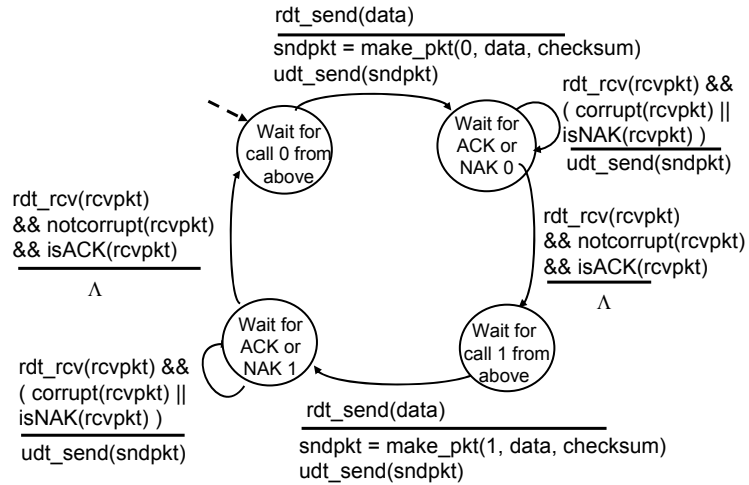
- ❑ sender adds *sequence number* to each pkt
- ❑ sender retransmits current pkt if ACK/NAK garbled
- ❑ receiver discards (doesn't deliver up) duplicate pkt

stop and wait

Sender sends one packet, then waits for receiver response

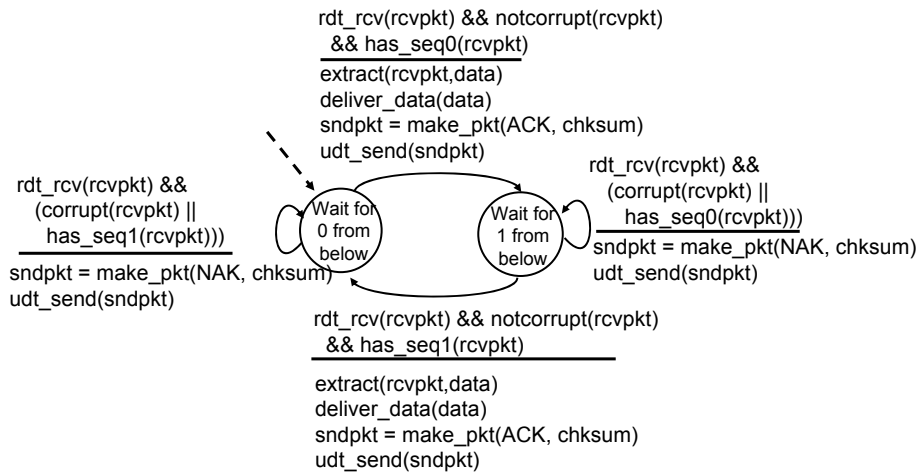
0-34

rdt2.1: sender, handles garbled ACK/NAKs



0-35

rdt2.1: receiver, handles garbled ACK/NAKs



0-36

rdt2.1: discussion

Sender:

- ❑ seq # added to pkt
- ❑ two seq. #'s (0,1) will suffice. Why?
- ❑ must check if received ACK/NAK corrupted
- ❑ FSM has twice as many states
 - state must "remember" whether "current" pkt has 0 or 1 seq. #

Receiver:

- ❑ must check if received packet is duplicate
 - state indicates whether 0 or 1 is expected pkt seq #

0-37

rdt2.2: a NAK-free protocol

- ❑ same functionality as rdt2.1, using ACKs only
- ❑ instead of NAK, receiver sends ACK for last pkt received OK
 - receiver must *explicitly* include seq # of pkt being ACKed
- ❑ duplicate ACK at sender results in same action as NAK: *retransmit current pkt*

0-38

rdt3.0: channels with errors AND loss

Assumption: underlying channel can also lose packets (data or ACKs)

- checksum, seq. #, ACKs, retransmissions will be of help, but not enough

□ Why seq #s

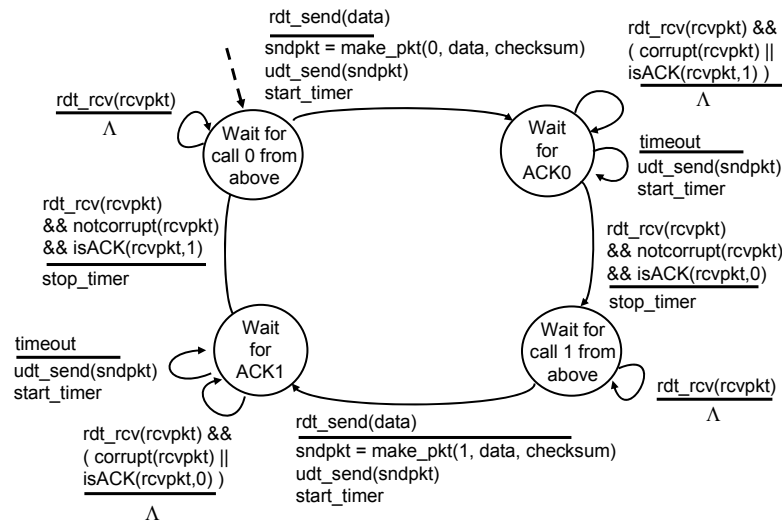
- detect reordering
- ACK, NAKing
- Detect missing packet
- Duplicate detection due to retransmissions

Approach: sender waits "reasonable" amount of time for ACK

- retransmits if no ACK received in this time
- if pkt (or ACK) just delayed (not lost):
 - retransmission will be duplicate, but use of 0,1 seq. #'s already handles this
 - receiver must specify seq # of pkt being ACKed
- requires countdown timer

0-39

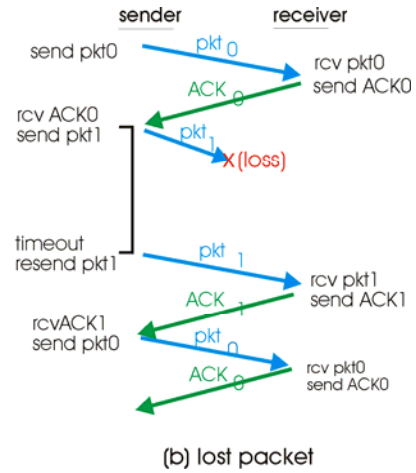
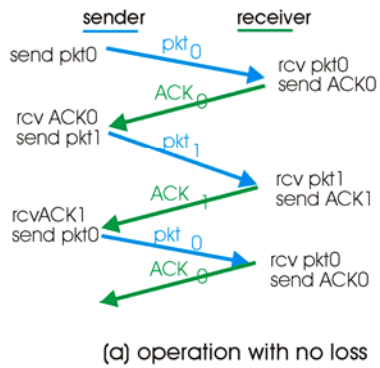
rdt3.0 sender



FSM specification of sender (details not important)

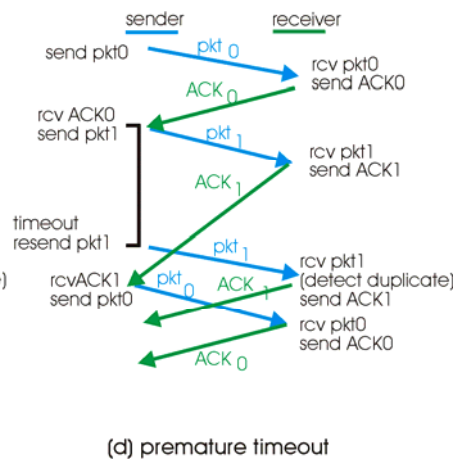
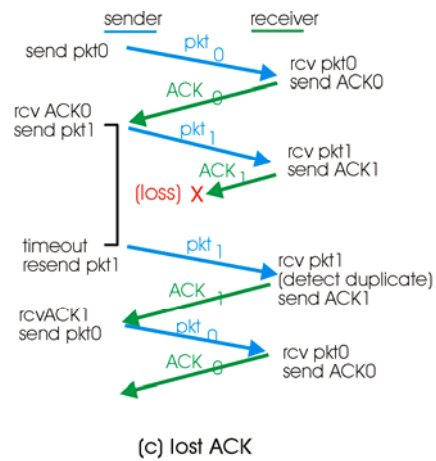
0-40

rdt3.0 in action



0-41

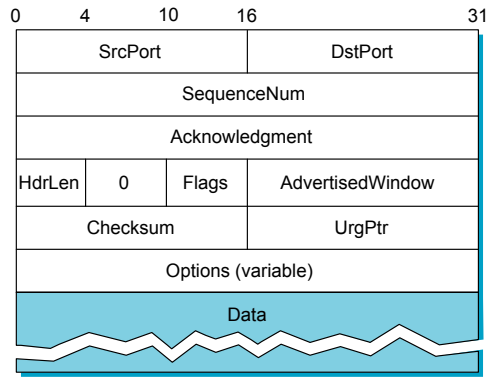
rdt3.0 in action



0-42

Case Study: TCP

TCP Segment Format:



Flags: SYN, FIN, RESET, PUSH, URG, ACK

0-43

Part 0: Networking Review

Goals:

- review key topics from intro networks course
 - equalize backgrounds
 - identify remedial work
 - ease into course

Overview:

- overview
- error control
- flow control**
- congestion control**
- routing
- LANs
- addressing
- synthesis:
 - "a day in the life"
 - control timescales

0-44

Flow Control (in TCP)

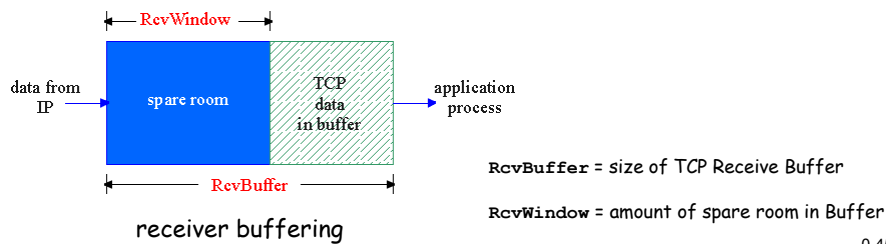
flow control

sender won't overrun receiver's buffers by transmitting too much, too fast

receiver: explicitly informs sender of (dynamically changing) amount of free buffer space

- **RcvWindow** field in TCP segment

sender: keeps the amount of transmitted, unACKed data less than most recently received **RcvWindow**



Principles of Congestion Control

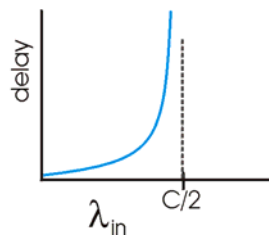
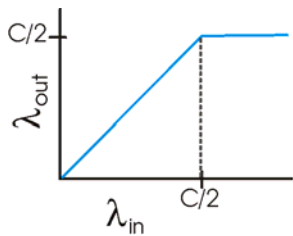
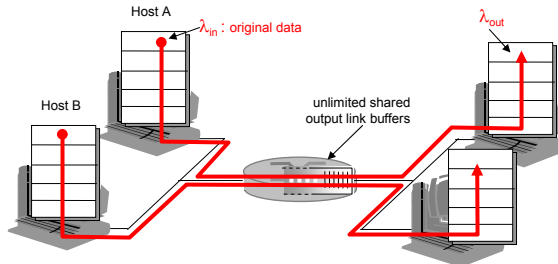
Congestion:

- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
 - lost packets (buffer overflow at routers)
 - long delays (queueing in router buffers)
- a top-10 problem!

0-46

Causes/costs of congestion: scenario 1

- two senders, two receivers
- one router, infinite buffers
- no retransmission

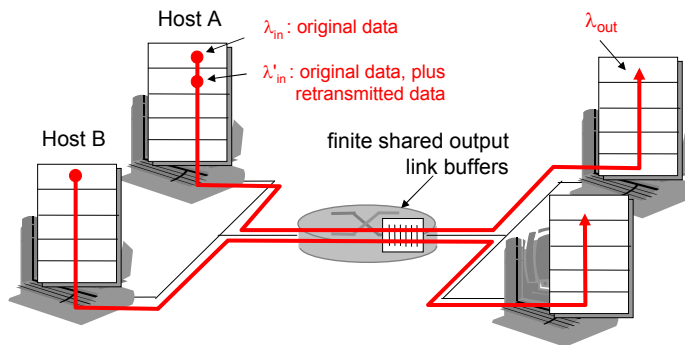


- large delays when congested
- maximum achievable throughput

0-47

Causes/costs of congestion: scenario 2

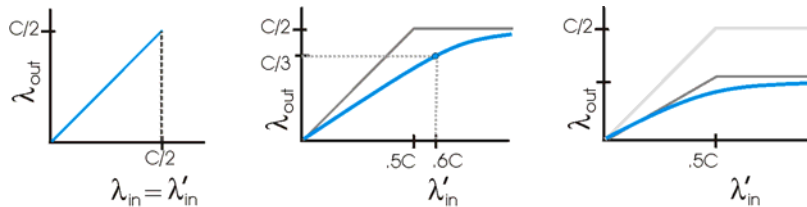
- one router, *finite* buffers
- sender retransmission of lost packet



0-48

Causes/costs of congestion: scenario 2

- always: $\lambda_{in} = \lambda_{out}$ (goodput)
- "perfect" retransmission only when loss: $\lambda'_{in} > \lambda_{out}$
- retransmission of delayed (not lost) packet makes λ'_{in} larger (than perfect case) for same λ_{out}



"costs" of congestion:

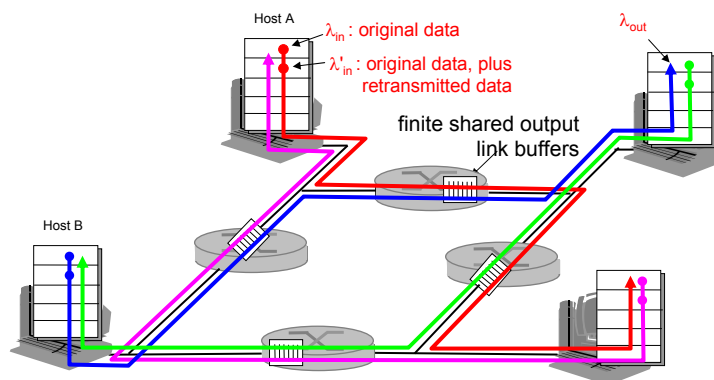
- more work (retrans) for given "goodput"
- unneeded retransmissions: link carries multiple copies of pkt

0-49

Causes/costs of congestion: scenario 3

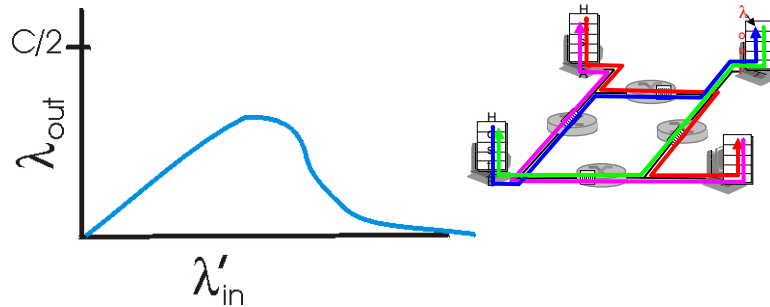
- four senders
- multihop paths
- timeout/retransmit

Q: what happens as λ_{in} and λ'_{in} increase?



0-50

Causes/costs of congestion: scenario 3



Another "cost" of congestion:

- when packet dropped, any "upstream transmission capacity used for that packet was wasted!

0-51

Approaches towards congestion control

Two broad approaches towards congestion control:

End-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

Network-assisted congestion control:

- routers provide feedback to end systems
 - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
 - explicit rate sender should send at
- Forms of feedback
 - sender: "choke" packet
 - receiver: packet hdr field

0-52

Case study: ATM ABR congestion control

ABR: available bit rate:

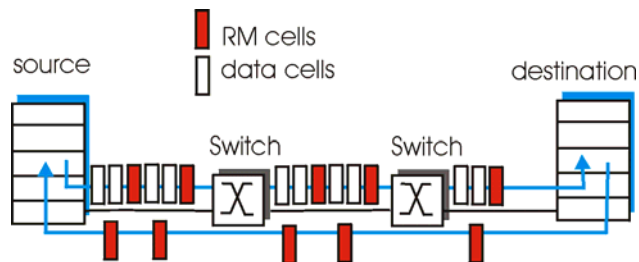
- "elastic service"
- if sender's path "underloaded":
 - sender should use available bandwidth
- if sender's path congested:
 - sender throttled to minimum guaranteed rate
- 3 signaling mechanisms:
 - NI/CI, ER, EFCI

RM (resource management) cells:

- sent by sender, interspersed with data cells
- bits in RM cell set by switches ("*network-assisted*")
 - NI bit: no increase in rate (mild congestion)
 - CI bit: congestion indication
- RM cells returned to sender by receiver, with bits intact
 - Switches can also directly generate RM cells to source

0-53

Case study: ATM ABR congestion control



- two-byte ER (explicit rate) field in RM cell
 - congested switch may lower ER value in cell
 - sender' send rate thus minimum supportable rate on path
- EFCI (explicit forward congestion indication) bit in data cells: set to 1 in congested switch
 - if data cell preceding RM cell has EFCI set, sender sets CI bit in returned RM cell

0-54

TCP Congestion Control

- end-end control (no network assistance)
- transmission rate limited by congestion window size, Congwin, over segments:



0-55

TCP congestion control:

- "probing" for usable bandwidth:
 - ideally: transmit as fast as possible (Congwin as large as possible) without loss
 - increase Congwin until loss (congestion)
 - loss: decrease Congwin, then begin probing (increasing) again
- Inherent assumption
 - Loss = congestion
- two "phases"
 - slow start
 - congestion avoidance
- important variables:
 - Congwin
 - threshold: defines threshold between the slow start and the congestion avoidance phase

0-56

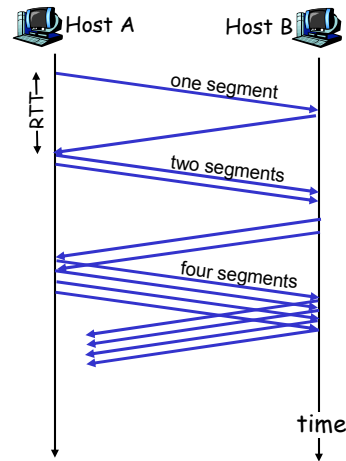
TCP Slowstart

Slowstart algorithm

```

initialize: Congwin = 1
for (each segment ACKed)
  Congwin++
until (loss event OR
      CongWin > threshold)
  
```

- exponential increase (per RTT) in window size (not so slow!)
- loss event: timeout (Tahoe TCP) and/or or three duplicate ACKs (Reno TCP)



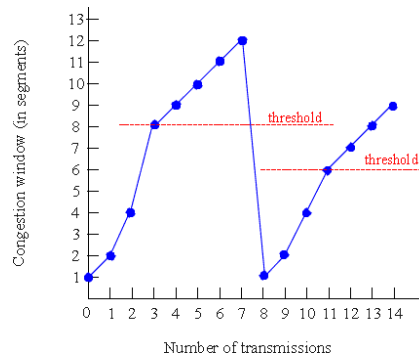
0-57

TCP Congestion Avoidance: Tahoe

TCP Tahoe Congestion avoidance

```

/* slowstart is over */
/* Congwin > threshold */
Until (loss event) {
  every w segments ACKed:
    Congwin++
}
threshold = Congwin/2
Congwin = 1
perform slowstart
  
```



Numerous improvements: TCP Reno, SACK

0-58

Current Status of TCP

- Lots of variants
 - Tahoe, Reno, NewReno, SACK, Vegas,
- Lots of research on improving TCP performance in specific scenarios
 - TCP for short transfers
 - TCP for wireless
 - TCP for networks with high $BW \cdot \text{delay}$ product
 - TCP for grid computing (e.g. TCP-NICE)
 - TCP for delay tolerant networks (?)

0-59

Part 0: Networking Review

Goals:

- review key topics from intro networks course
 - equalize backgrounds
 - identify remedial work
 - ease into course

Overview:

- overview
- error control
- flow control
- congestion control
- routing (and network layer services)
- LANs
- addressing
- synthesis:
 - "a day in the life"
 - control timescales

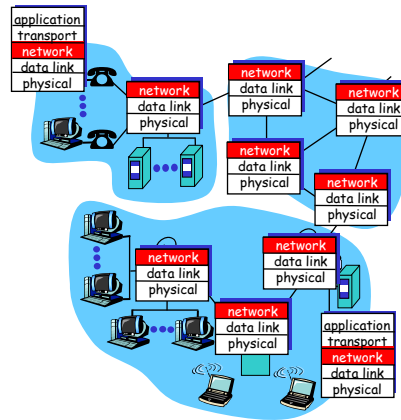
0-60

Network layer functions

- transport packet from sending to receiving hosts
- network layer protocols in *every* host, router

three important functions:

- *path determination*: route taken by packets from source to dest. *Routing algorithms*
- *switching*: move packets from router's input to appropriate router output
- *call setup*: some network architectures require router call setup along path before data flows



0-61

Network service model

Q: What *service model* for "channel" transporting packets from sender to receiver?

- service abstraction*
- guaranteed bandwidth?
 - preservation of inter-packet timing (no jitter)?
 - loss-free delivery?
 - in-order delivery?
 - congestion feedback to sender?

The most important abstraction provided by network layer:

virtual circuit
or
datagram?

CRUCIAL question!

0-62

Virtual circuits

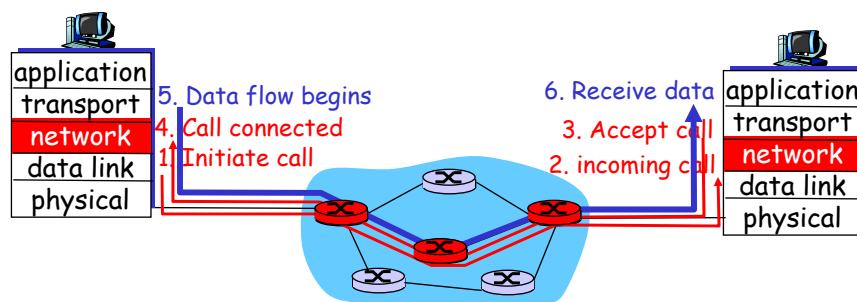
"source-to-dest path behaves much like telephone circuit"

- performance-wise
 - network actions along source-to-dest path
-
- call setup, teardown for each call *before* data can flow
 - each packet carries VC identifier (not destination host ID)
 - every router on source-dest path maintains "state" for each passing connection
 - transport-layer connection only involved two end systems
 - link, router resources (bandwidth, buffers) may be *allocated* to VC
 - to get circuit-like perf.

0-63

Virtual circuits: signaling protocols

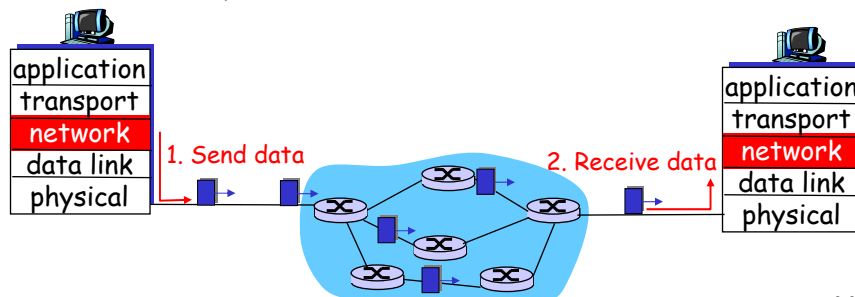
- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet



0-64

Datagram networks: the Internet model

- ❑ no call setup at network layer
- ❑ routers: no state about end-to-end connections
 - no network-level concept of "connection"
- ❑ packets typically routed using destination host ID
 - packets between same source-dest pair may take different paths



0-65

Datagram or VC network: why?

Internet

- ❑ data exchanged among computers
 - "elastic" service, no strict timing req.
- ❑ "smart" end systems (computers)
 - can adapt, perform cong. control, error recovery
 - simple inside network, complexity at "edge"
- ❑ many link types
 - different characteristics
 - uniform service difficult

ATM

- ❑ evolved from telephony
- ❑ human conversation:
 - strict timing, reliability requirements
 - need for guaranteed service
- ❑ "dumb" end systems
 - telephones
 - complexity inside network

0-66

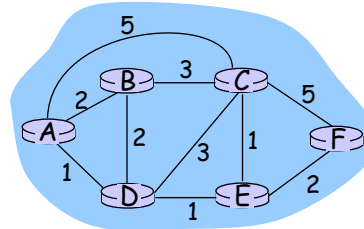
Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
 - link cost: delay, \$ cost, or congestion level
- "good" path:
 - typically means minimum cost path
 - other def's possible



0-67

Routing Algorithm classification

Global or decentralized information?

Global:

- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly
 - periodic update
 - in response to link cost changes

0-68

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
 - accomplished via "link state broadcast"
 - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
 - gives routing table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

- $c(i,j)$: link cost from node i to j . cost infinite if not direct neighbors
- $D(v)$: current value of cost of path from source to dest. V
- $p(v)$: predecessor node along path from source to v , that is next v
- N : set of nodes whose least cost path definitively known

0-69

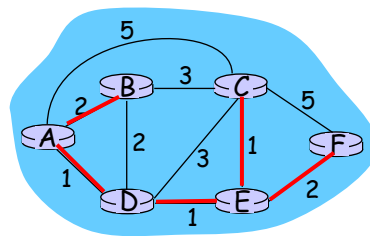
Dijkstra's Algorithm

- 1 **Initialization:**
- 2 $N = \{A\}$
- 3 for all nodes v
- 4 if v adjacent to A
- 5 then $D(v) = c(A,v)$
- 6 else $D(v) = \infty$
- 7
- 8 **Loop**
- 9 find w not in N such that $D(w)$ is a minimum
- 10 add w to N
- 11 update $D(v)$ for all v adjacent to w and not in N :
- 12 $D(v) = \min(D(v), D(w) + c(w,v))$
- 13 /* new cost to v is either old cost to v or known
- 14 shortest path cost to w plus cost from w to v */
- 15 **until all nodes in N**

0-70

Dijkstra's algorithm: example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→0	A	2,A	5,A	1,A	infinity	infinity
→1	AD	2,A	4,D		2,D	infinity
→2	ADE	2,A	3,E			4,E
→3	ADEB		3,E			4,E
→4	ADEBC					4,E
5	ADEBCF					



0-71

Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

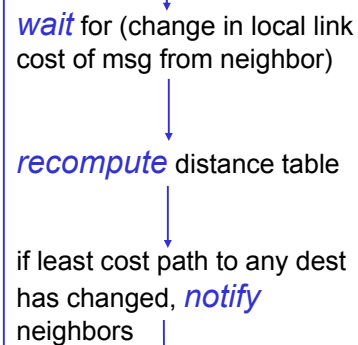
asynchronous:

- nodes need *not* exchange info/iterate in lock step!

distributed:

- each node communicates *only* with directly-attached neighbors

Each node:



0-72

Distance Vector Algorithm: Data Structures

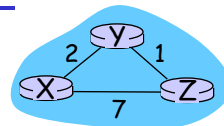
- Each node x maintains:
 - For each neighbor v , cost $c(x,v)$
 - Node x 's distance vector: $D_x = [D_x(y): y \in N]$ containing x 's estimate of cost to all destinations
 - Distance vectors for each neighbor v :
 $D_v = [D_v(y): y \in N]$

- Basic operation: Bellman-Ford algorithm
$$D_x(y) = \min_v \{c(x,v) + D_v(y)\} \quad \forall y \in N$$

0-73

Distance Vector Algorithm:

At all nodes, X :



- 1 **Initialization:**
- 2 For all destinations $y \in N$:
- 3 $D_x(y) = c(x,y)$ /* if y is not a neighbor, then $c(x,y) = \infty$ */:
- 4 For each neighbor w
- 5 $D_w(y) = \infty$ for all destinations $y \in N$
- 6 For each neighbor w
- 7 Send distance vector $D_x = [D_x(y): y \in N]$ to w

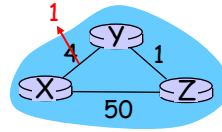
- 8 **Loop:**
- 9 Wait (until communication from neighbor w)
- 10 For each $y \in N$:
- 11 $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$
- 12 If $D_x(y)$ changes for any destination y
- 13 Send distance vector $D_x = [D_x(y): y \in N]$ to all neighbors

0-74

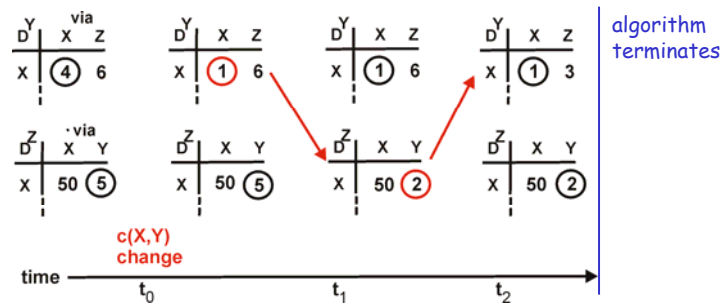
Distance Vector: link cost changes

Link cost changes:

- node detects local link cost change
- updates distance table
- if cost change in least cost path, notify neighbors



"good news travels fast"

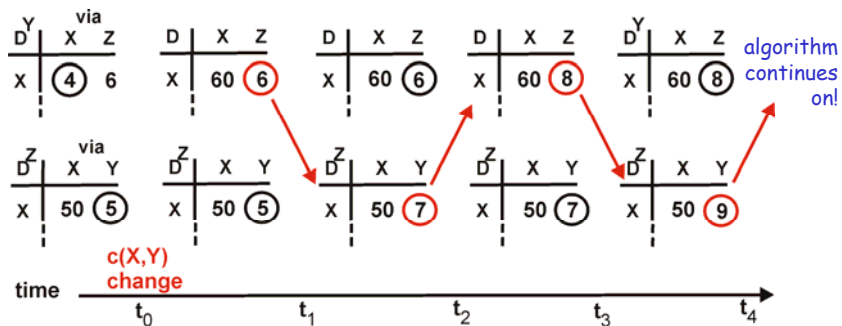
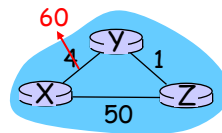


0-75

Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!

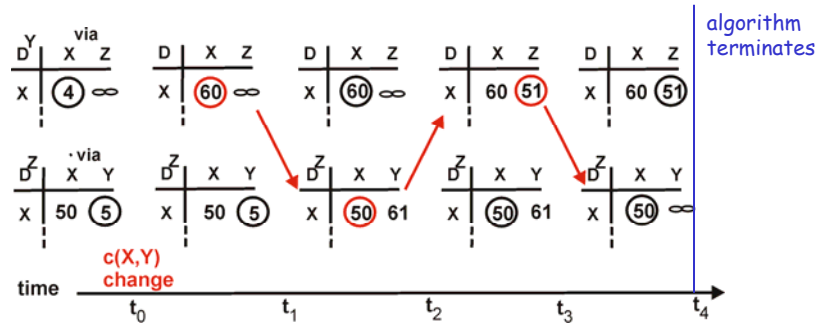
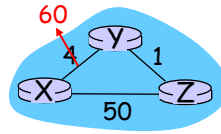


0-76

Distance Vector: poisoned reverse

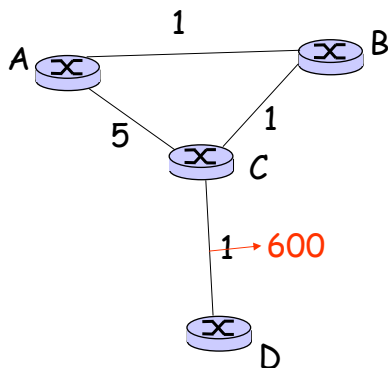
If Z routes through Y to get to X :

- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?



0-77

Is Poisoned Reverse Enough?



- $\text{dist}^{A \rightarrow D}(\text{via } B) = 3$, $\text{dist}^{B \rightarrow D}(\text{via } C) = 2$, $\text{dist}^{C \rightarrow D}(\text{via } D) = 1$
- $\text{dist}^{A \rightarrow D}(\text{via } B) = 3$, $\text{dist}^{B \rightarrow D}(\text{via } C) = 2$, $\text{dist}^{C \rightarrow D}(\text{via } A) = 8 \rightarrow$ a loop!
- $\text{dist}^{A \rightarrow D}(\text{via } B) = 3$, $\text{dist}^{B \rightarrow D}(\text{via } C) = 9$, $\text{dist}^{C \rightarrow D}(\text{via } A) = 8$
- $\text{dist}^{A \rightarrow D}(\text{via } B) = 10$, $\text{dist}^{B \rightarrow D}(\text{via } C) = 9$, $\text{dist}^{C \rightarrow D}(\text{via } A) = 8$
- $\text{dist}^{A \rightarrow D}(\text{via } B) = 10$, $\text{dist}^{B \rightarrow D}(\text{via } C) = 9$, $\text{dist}^{C \rightarrow D}(\text{via } A) = 15$
-

0-78

Comparison of LS and DV algorithms

Message Complexity

- **LS:** with n nodes, E links, $O(nE)$ msgs sent each
- **DV:** exchange between neighbors only
 - convergence time varies
 - if every distance vector is sent on every link once $\rightarrow O(nE)$ msgs

Speed of Convergence

- **LS:** $O(n^2)$ algorithm requires $O(nE)$ msgs
 - may have oscillations
- **DV:** convergence time varies
 - may be routing loops
 - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
 - error propagate thru network

0-79

Hierarchical Routing

Our routing review thus far - idealization

- all routers identical
 - network "flat"
- ... *not* true in practice

scale: with 200 million destinations:

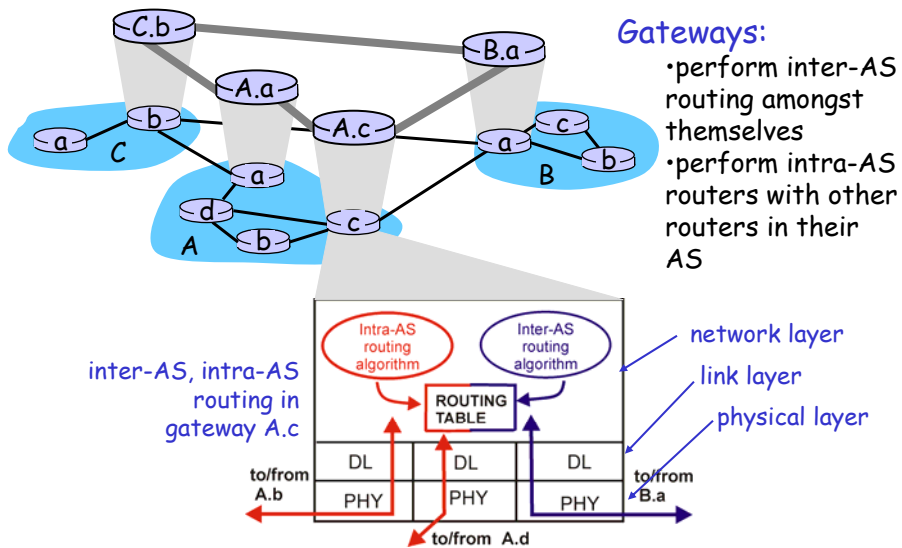
- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

0-80

IntraAS and InterAS routing



0-81

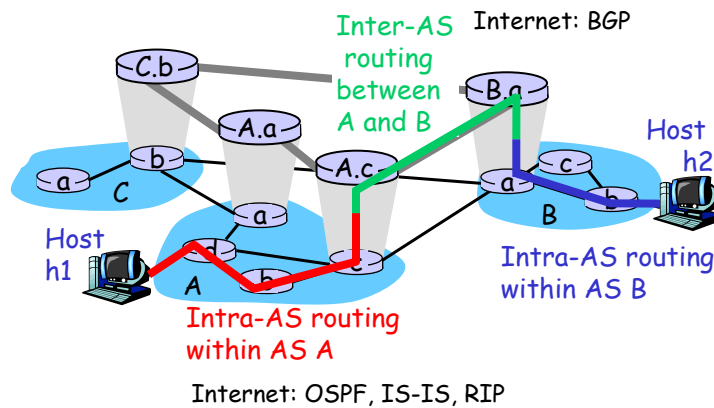
Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
 - routers in same AS run same routing protocol
 - "intra-AS" routing protocol
 - routers in different AS can run different intra-AS routing protocol
- gateway routers

 - special routers in AS
 - run intra-AS routing protocol with all other routers in AS
 - also responsible for routing to destinations outside AS
 - run *inter-AS routing* protocol with other gateway routers

0-82

IntraAS and InterAS routing



0-83

Acronyms

- **IGP**: Interior Gateway Protocol
 - **OSPF**: Open Shortest Path First
 - **RIP**: Routing Information Protocol
 - **IGRP**: Interior Gateway Routing Protocol
- **EGP**: Exterior Gateway Protocol
 - **BGP**: Border Gateway Protocol
 - **iBGP**: internal BGP
 - **eBGP**: external BGP

0-84

Intra-AS Routing

- ❑ Also known as **Interior Gateway Protocols (IGP)**
- ❑ Most common Intra-AS routing protocols:
 - RIP: Routing Information Protocol
 - OSPF: Open Shortest Path First
 - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

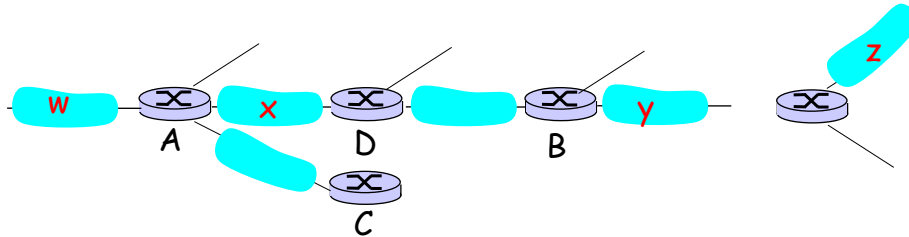
0-85

RIP (Routing Information Protocol)

- ❑ Distance vector algorithm
- ❑ Included in BSD-UNIX Distribution in 1982
- ❑ Distance metric: # of hops (max = 15 hops)
- ❑ Distance vectors: exchanged every 30 sec via Response Message (also called **advertisement**)
- ❑ Each advertisement: route to up to 25 destination nets

0-86

RIP (Routing Information Protocol)



Destination Network	Next Router	Num. of hops to dest.
W	A	2
Y	B	2
Z	B	7
X	--	1
....

Routing table in D

0-87

RIP: Link Failure and Recovery

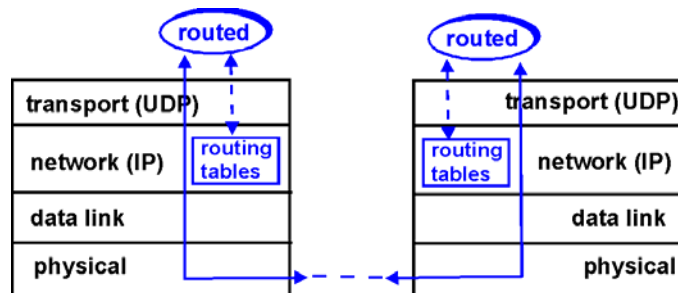
If no advertisement heard after 180 sec → neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

0-88

RIP Table processing

- ❑ RIP routing tables managed by **application-level** process called route-d (daemon)
 - Q: Is this a violation of layering?
- ❑ advertisements sent in UDP packets, periodically repeated



0-89

OSPF (Open Shortest Path First)

- ❑ "open": publicly available
- ❑ Uses Link State algorithm
 - LS packet dissemination
 - Topology map at each node
 - Route computation using Dijkstra's algorithm
- ❑ OSPF advertisement carries one entry per neighbor router
- ❑ Advertisements disseminated to **entire** AS (via flooding)
 - Carried in OSPF messages directly over IP (rather than TCP or UDP)

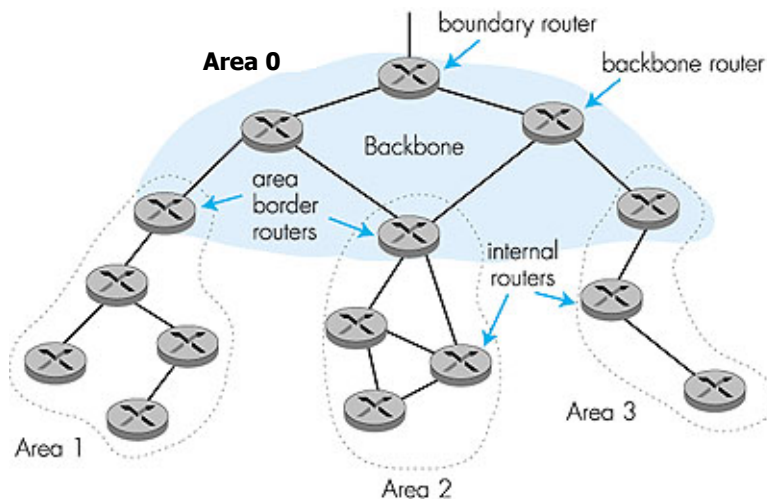
0-90

OSPF "advanced" features (not in RIP)

- ❑ **Security:** all OSPF messages authenticated (to prevent malicious intrusion)
- ❑ **Multiple same-cost paths** allowed (only one path in RIP)
- ❑ For each link, multiple cost metrics for different **TOS** (e.g., satellite link cost set "low" for best effort; high for real time)
- ❑ Integrated uni- and **multicast** support:
 - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- ❑ **Hierarchical** OSPF in large domains.

0-91

Hierarchical OSPF



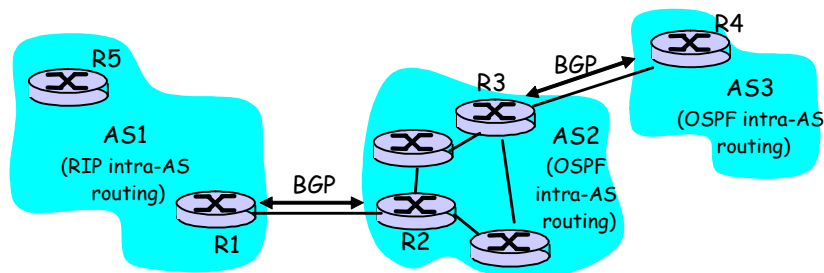
0-92

Hierarchical OSPF

- ❑ **Two-level hierarchy:** local area, backbone.
 - Link-state advertisements only in area
 - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- ❑ **Area border routers:** "summarize" distances to nets in own area, advertise to other Area Border routers.
- ❑ **Backbone routers:** run OSPF routing limited to backbone.
- ❑ **Boundary routers:** connect to other AS's.

0-93

Inter-AS routing in the Internet: BGP



0-94

Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol):** *the de facto standard*
- **Path Vector** protocol:
 - similar to Distance Vector protocol
 - each Border Gateway broadcast to neighbors (peers) *entire path* (i.e., sequence of AS's) to destination
 - BGP routes to networks (ASs), not individual hosts
 - E.g., Gateway X may send its path to dest. Z:

$$\text{Path (X,Z)} = X, Y_1, Y_2, Y_3, \dots, Z$$

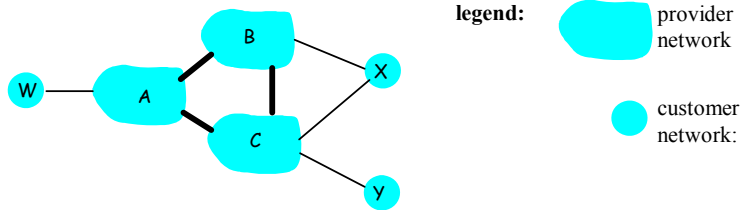
0-95

Internet inter-AS routing: BGP

- Suppose:* gateway X send its path to peer gateway W
- W may or may not select path offered by X
 - cost, policy (don't route via competitors AS), loop prevention reasons.
 - If W selects path advertised by X, then:
$$\text{Path (W,Z)} = w, \text{Path (X,Z)}$$
 - Note: X can control incoming traffic by controlling its route advertisements to peers:
 - e.g., don't want to route traffic to Z → don't advertise any routes to Z

0-96

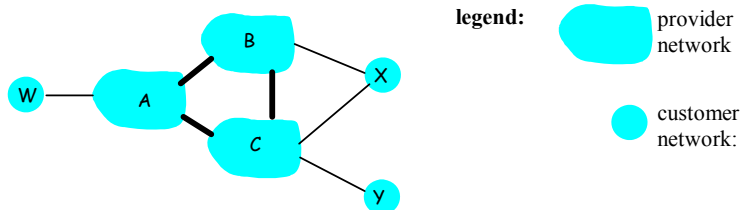
BGP: controlling who routes to you



- A,B,C are **provider networks**
- X,W,Y are customer (of provider networks)
- X is **dual-homed**: attached to two networks
 - X does not want to route from B via X to C
 - .. so X will not advertise to B a route to C

0-97

BGP: controlling who routes to you



- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW?
 - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
 - B wants to force C to route to w via A
 - B wants to route **only** to/from its customers!

0-98

BGP operation

Q: What does a BGP router do?

- Receiving and filtering route advertisements from directly attached neighbor(s).
- Route selection.
 - To route to destination X, which path (of several advertised) will be taken?
- Sending route advertisements to neighbors.

0-99

BGP messages

- BGP messages exchanged using TCP.
 - Q1: Is this a violation of layering?
 - Q2: Any undesirable effects of running BGP on top of TCP?
- BGP messages:
 - **OPEN**: opens TCP connection to peer and authenticates sender
 - **UPDATE**: advertises new path (or withdraws old)
 - **KEEPALIVE** keeps connection alive in absence of UPDATES; also ACKs OPEN request
 - **NOTIFICATION**: reports errors in previous msg; also used to close connection

0-100

BGP Details

- ❑ iBGP and eBGP use the same BGP protocol !
- ❑ eBGP sessions: import/export routes from/to other ASes.
 - Can apply import / export policies
- ❑ iBGP sessions: distribute routes to all routers
 - In principle, one can directly inject routes from BGP to IGP → not every router in an AS needs to speak BGP
 - This happens in some enterprise environment
 - In practice, large ISPs don't inject routes from BGP into IGP → every router speaks BGP
- ❑ BGP route decision process
 - Based on local-pref, AS hop-count, IGP weight, etc.
- ❑ RIB vs. FIB
 - Routing information base: routes learned from all routing protocols
 - Forwarding information base: (prefix, output interface, next-hop)

0-101

How BGP & IGP work together?

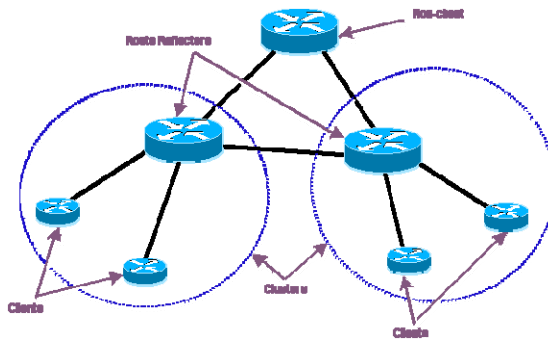
- ❑ Every router run both IGP and BGP
- ❑ BGP: learn route from peers, distribute them to interior gateways, select routes
 - Outcome: which next-hop AS to use for a destination prefix
- ❑ When multiple egress points are equally good, each router direct traffic to the closest egress point
 - IGP path weight is the tie breaker
- ❑ IGP gives the next hop interface to that egress point
 - Outcome: forwarding table (FIB)

- ❑ **Q: Any undesirable implications of using IGP path weight as tie breaker?**

0-102

Improve BGP Scalability

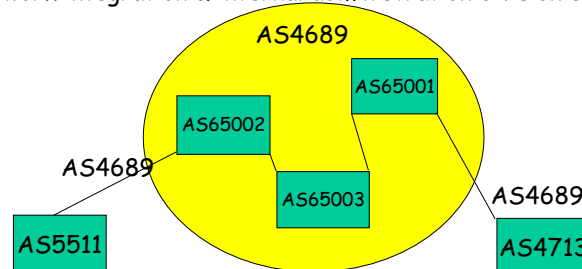
- ❑ iBGP Mesh not scalable
 - $N(N-1)/2$ BGP sessions
- ❑ Solution: Route Reflector (RR)
 - Divide BGP speakers into clusters
 - For routes from a client, send to other clients & all non-clients
 - For routes from a non-client, send to all its clients



0-103

BGP Confederation

- ❑ A BGP speaker with multiple AS numbers can be made to look like a single AS number from outside
- ❑ **Q:** When is BGP confederation useful?
 - Countermeasure for OSPF process enlargement
 - Compress OSPF process
 - Too large → divide
 - Possible to control policy per region
 - Localization of failure
 - Absolute avoid it becoming network-wide
 - Network integration & internal administration division etc.



0-104

Why different Intra- and Inter-AS routing ?

Policy:

- ❑ Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- ❑ Intra-AS: single admin, so no policy decisions needed

Scale:

- ❑ hierarchical routing saves table size, reduced update traffic

Performance:

- ❑ Intra-AS: can focus on performance
- ❑ Inter-AS: policy may dominate over performance

0-105

Addressing

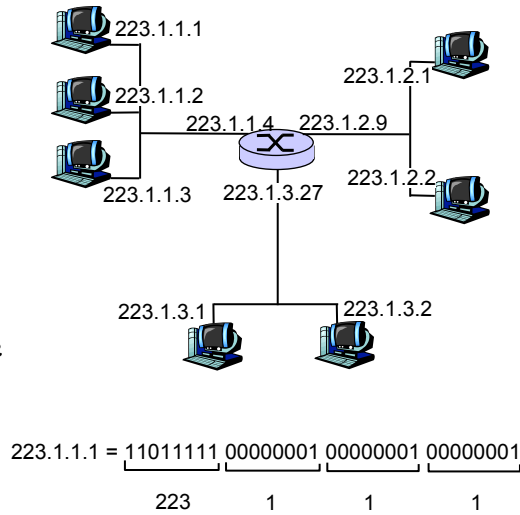
- ❑ What's an address?
 - identifier that differentiates between me and someone else, and also helps route data to/from me
- ❑ Real world examples of addressing?
 - mailing address
 - office #, floor, etc
 - Phone#

 - different "levels of addressing"

0-106

Addressing: network layer

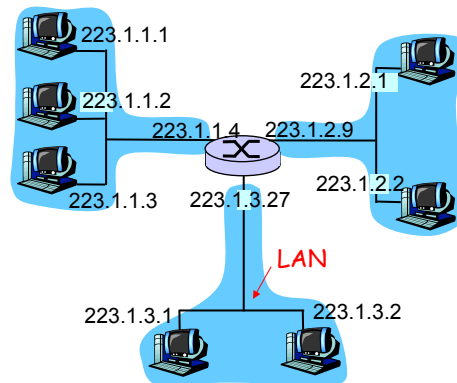
- IP address: 32-bit identifier for host, router *interface*
- *interface*: connection between host, router and physical link
 - router's typically have multiple interfaces
 - host may have multiple interfaces
 - IP addresses associated with interface, not host, router



0-107

IP Addressing

- IP address:
 - network part (high order bits)
 - host part (low order bits)
- *What's a network?* (from IP address perspective)
 - device interfaces with same network part of IP address
 - can physically reach each other without intervening router



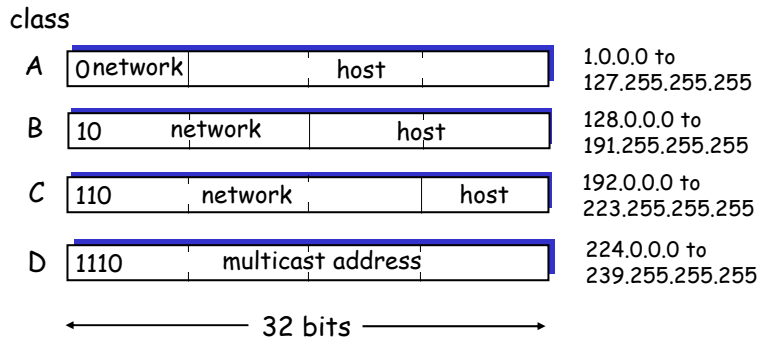
network consisting of 3 IP networks
(for IP addresses starting with 223,
first 24 bits are network address)

0-108

IP Addresses

given notion of "network", let's re-examine IP addresses:

"class-full" addressing:



0-109

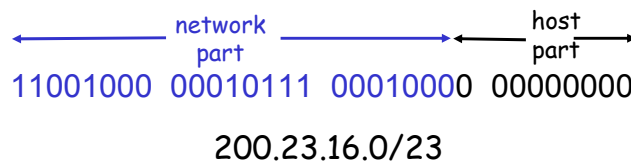
IP addressing: CIDR

□ Classful addressing:

- inefficient use of address space, address space exhaustion
- e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network

□ CIDR: Classless InterDomain Routing

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



0-110

IP addresses: how to get one?

Q: How does host get IP address?

- ❑ hard-coded by system admin in a file
 - Wintel: control-panel->network->configuration->tcp/ip->properties
 - UNIX: /etc/rc.config
- ❑ **DHCP: Dynamic Host Configuration Protocol:** dynamically get address: "plug-and-play"
 - host broadcasts "DHCP discover" msg
 - DHCP server responds with "DHCP offer" msg
 - host requests IP address: "DHCP request" msg
 - DHCP server sends address: "DHCP ack" msg

0-111

Getting a datagram from source to dest.

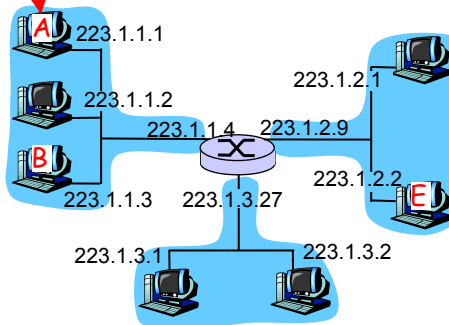
IP datagram:

misc	source	dest	data
fields	IP addr	IP addr	

- ❑ datagram remains **unchanged**, as it travels source to destination
- ❑ addr fields of interest here

routing table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



0-112

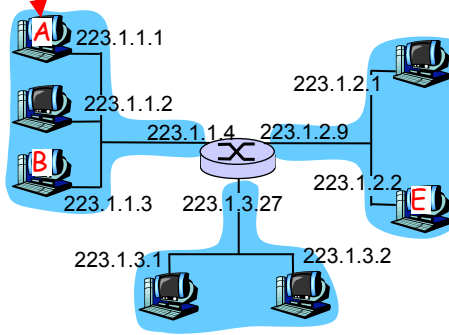
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.1.3	data
-------------	-----------	-----------	------

Starting at A, given IP datagram addressed to B:

- ❑ look up net. address of B
- ❑ find B is on same net. as A
- ❑ link layer will send datagram directly to B inside link-layer frame
 - B and A are directly connected

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



0-113

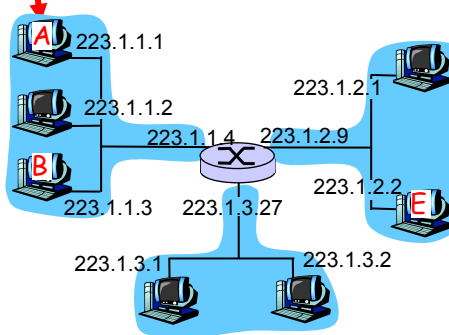
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.3	data
-------------	-----------	-----------	------

Starting at A, dest. E:

- ❑ look up network address of E
- ❑ E on *different* network
 - A, E not directly attached
- ❑ routing table: next hop router to E is 223.1.1.4
- ❑ link layer sends datagram to router 223.1.1.4 inside link-layer frame
- ❑ datagram arrives at 223.1.1.4
- ❑ continued.....

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



0-114

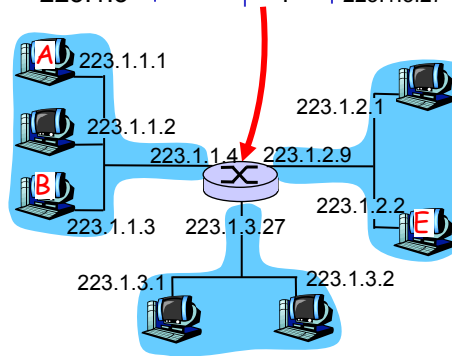
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.3	data
-------------	-----------	-----------	------

Arriving at 223.1.4,
destined for 223.1.2.2

- ❑ look up network address of E
- ❑ E on *same* network as router's interface 223.1.2.9
 - router, E directly attached
- ❑ link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- ❑ datagram arrives at 223.1.2.2!!! (hooray!)

Dest. network	next router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27



0-115

Part 0: Networking Review

Goals:

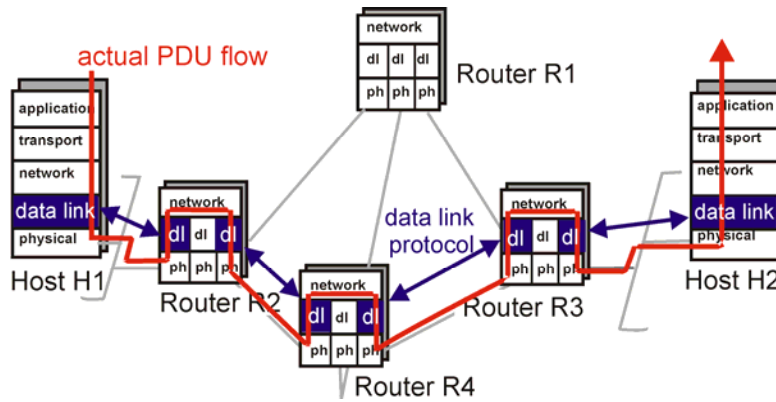
- ❑ review key topics from intro networks course
 - equalize backgrounds
 - identify remedial work
 - ease into course

Overview:

- ❑ overview
- ❑ error control
- ❑ flow control
- ❑ congestion control
- ❑ routing
- ❑ LANs
- ❑ addressing (cont.)
- ❑ synthesis:
 - "a day in the life"
 - control timescales

0-116

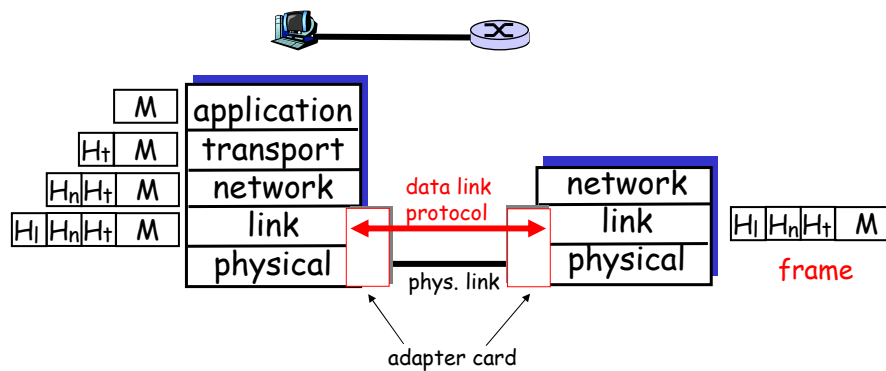
Link Layer: setting the context



0-117

Link Layer: setting the context

- two *physically connected* devices:
 - host-router, router-router, host-host
- unit of data: *frame*



0-118

Link Layer Services

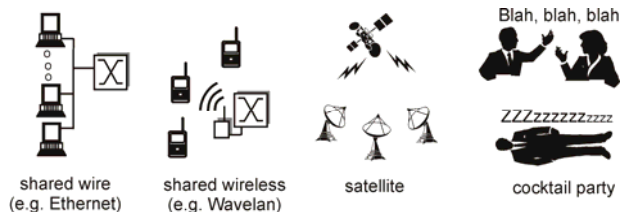
- **Framing, link access:**
 - encapsulate datagram into frame, adding header, trailer
 - implement channel access if shared medium (e.g., Ethernet)
 - 'physical addresses' used in frame headers to identify source, dest
 - different from IP address!
- **reliable delivery between two physically connected devices**
- **flow control**
- **error detection/correction**

0-119

Multiple Access Links and Protocols

Three types of "links":

- point-to-point (single wire, e.g. PPP, SLIP)
- **broadcast** (shared wire or medium; e.g, Ethernet, Wavelan, etc.)



- switched (e.g., switched Ethernet, ATM etc)

0-120

Multiple Access protocols

- single shared communication channel
- two or more simultaneous transmissions by nodes: interference
 - only one node can send **successfully** at a time
- **multiple access protocol:**
 - distributed algorithm that determines how stations share channel, i.e., determine when station can transmit
 - communication about channel sharing must use channel itself!
 - what to look for in multiple access protocols:
 - synchronous or asynchronous
 - information needed about other stations
 - robustness (e.g., to channel errors)
 - performance

0-121

MAC Protocols: a taxonomy

Three broad classes:

- **Channel Partitioning**
 - divide channel into smaller "pieces" (time slots, frequency, code)
 - allocate piece to node for exclusive use
- **Random Access**
 - allow collisions
 - "recover" from collisions
- **"Taking turns"**
 - tightly coordinate shared access to avoid collisions

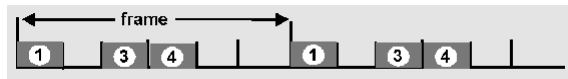
Goal: efficient, fair, simple, decentralized

0-122

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

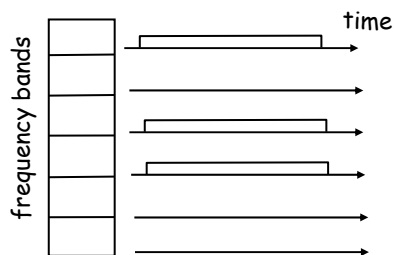


0-123

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



0-124

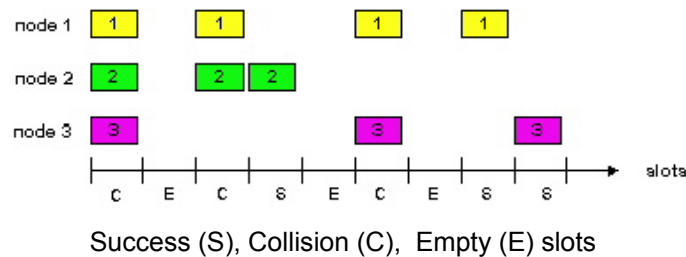
Random Access protocols

- 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
 - CSMA and CSMA/CD

0-125

Slotted Aloha

- time is divided into equal size slots (= pkt trans. time)
- node with new arriving pkt: transmit at beginning of next slot
- if collision: retransmit pkt in future slots with probability p , until successful.



0-126

Slotted Aloha efficiency

Q: what is max fraction slots successful?

A: Suppose N stations have packets to send

- each transmits in slot with probability p
- prob. successful transmission S is:

by single node: $S = p (1-p)^{N-1}$

by any of N nodes

$S = \text{Prob (only one transmits)}$

$= N p (1-p)^{N-1}$

... choosing optimum p as $n \rightarrow \infty$...

$= 1/e = .37$ as $N \rightarrow \infty$

At best: channel
use for useful
transmissions 37%
of time!

0-127

CSMA: Carrier Sense Multiple Access

CSMA: listen before transmit:

- If channel sensed idle: transmit entire pkt
- If channel sensed busy, defer transmission
 - **Persistent CSMA:** retry immediately with probability p when channel becomes idle (may cause instability)
 - **Non-persistent CSMA:** retry after random interval

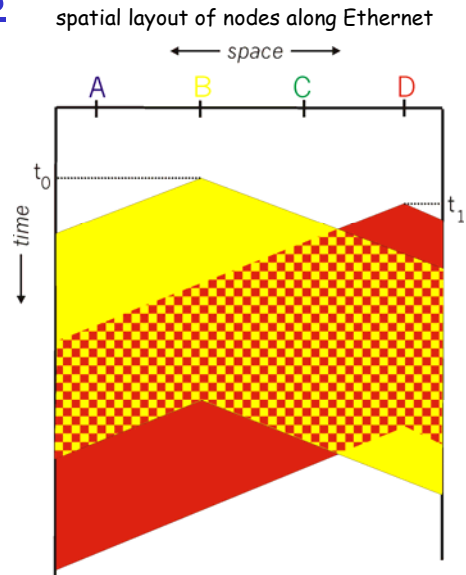
0-128

CSMA collisions

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

collision:
entire packet transmission
time wasted

note:
role of distance and
propagation delay in
determining collision prob.



CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- persistent or non-persistent retransmission

□ collision detection:

- easy in wired LANs: measure signal strengths, compare transmitted, received signals
- difficult in wireless LANs: receiver shut off while transmitting

0-130

"Taking Turns" MAC protocols

channel partitioning MAC protocols:

- share channel efficiently at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

Random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

"taking turns" protocols

look for best of both worlds!

0-131

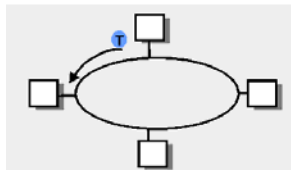
"Taking Turns" MAC protocols

Polling:

- master node "invites" slave nodes to transmit in turn
- Request to Send, Clear to Send msgs
- concerns:
 - polling overhead
 - latency
 - single point of failure (master)

Token passing:

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



0-132

Summary of MAC protocols

- What do you do with a shared media?
 - Channel Partitioning, by time, frequency or code
 - Time Division, Frequency Division, Code Division
 - Random partitioning (dynamic),
 - ALOHA, S-ALOHA, CSMA, CSMA/CD
 - carrier sensing: easy in some technologies (wire), hard in others (wireless)
 - CSMA/CD used in Ethernet
 - Taking Turns
 - polling from a central site, token passing

0-133

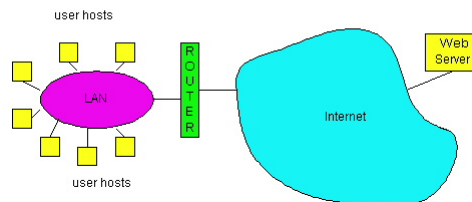
LAN technologies

Data link layer so far:

- services, error detection/correction, multiple access

Next: LAN technologies

- addressing
- Ethernet
- hubs, switches



0-134

LAN Addresses and ARP

32-bit IP address:

- network-layer address
- used to get datagram to destination network (recall IP network definition)

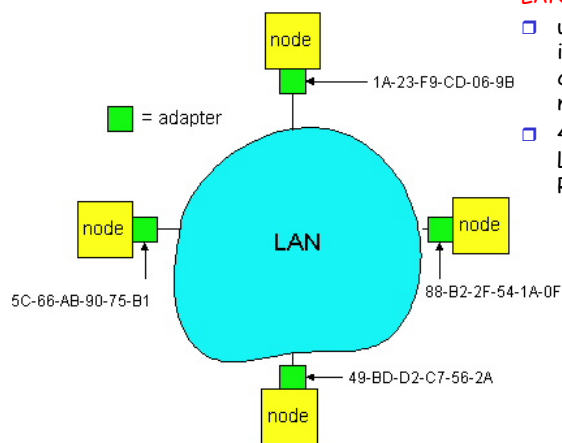
LAN (or MAC or physical) address:

- used to get frame from one interface to another physically-connected interface (same network)
- 48 bit MAC address (for most LANs) burned in the adapter ROM
- WHY MAC and Internet addresses separate?
 - IP addresses depend on network that you're on
 - MAC address in hardware makes it faster
 - "Permanent" unique identifier worldwide, forever
 - ** What about networks without IP addresses? E.g. IPX, DECnet ...

0-135

LAN Addresses

Each adapter on LAN has unique LAN address



LAN (or MAC or physical) address:

- used to get datagram from one interface to another physically-connected interface (same network)
- 48 bit MAC address (for most LANs) burned in the adapter ROM

0-136

LAN Address (more)

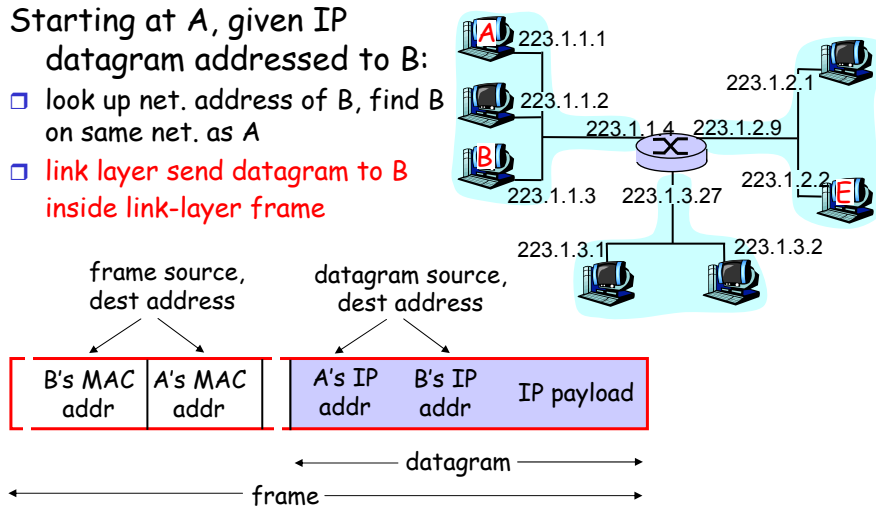
- ❑ MAC address allocation administered by IEEE
- ❑ manufacturer buys portion of MAC address space (to assure uniqueness)
- ❑ Analogy:
 - (a) MAC address: like Social Security Number
 - (b) IP address: like postal address
- ❑ MAC flat address → portability
 - can move LAN card from one LAN to another
- ❑ IP hierarchical address NOT portable
 - depends on network to which one attaches

0-137

From IP to MAC addresses

Starting at A, given IP datagram addressed to B:

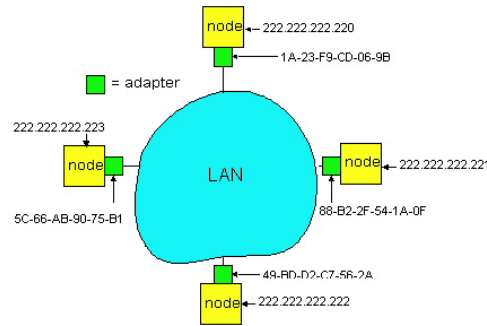
- ❑ look up net. address of B, find B on same net. as A
- ❑ link layer send datagram to B inside link-layer frame



0-138

ARP: Address Resolution Protocol

Question: how to determine MAC address of B given B's IP address?



□ Each IP node (Host, Router) on LAN has **ARP** module, table

□ ARP 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)

0-139

ARP protocol

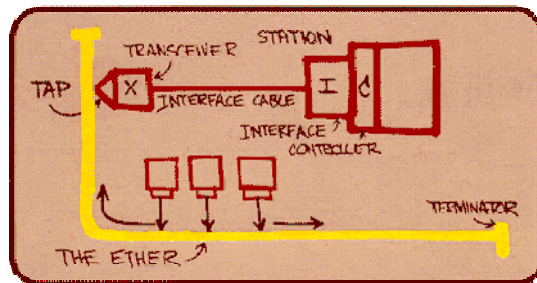
- A knows B's IP address, wants to learn physical address of B
- A **broadcasts** ARP query pkt, containing B's IP address
 - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) physical layer address
- A caches (saves) IP-to-physical address pairs until information becomes old (times out)
 - soft state: information that times out (goes away) unless refreshed

0-140

Ethernet: Case Study

"dominant" LAN technology:

- ❑ cheap \$20 for 100Mbps!
- ❑ first widely used LAN technology
- ❑ Simpler, cheaper than token LANs and ATM
- ❑ Kept up with speed race: 10, 100, 1000 Mbps

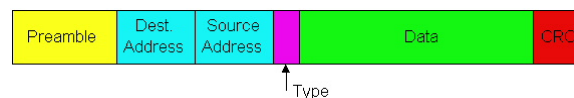


Metcalfe's Ethernet sketch

0-141

Ethernet Frame Structure

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



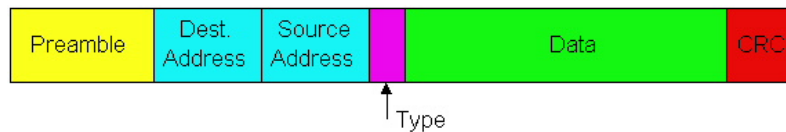
Preamble:

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

0-142

Ethernet Frame Structure (more)

- **Addresses:** 6 bytes, frame is received by all adapters on a LAN and dropped if address does not match
- **Type:** indicates the higher layer protocol, mostly IP but others may be supported such as Novell IPX and AppleTalk)
- **CRC:** checked at receiver, if error is detected, the frame is simply dropped



0-143

Ethernet: uses CSMA/CD

A: sense channel, if idle

```
then {
    transmit and monitor the channel;
    If detect another transmission
    then {
        abort and send jam signal;
        update # collisions;
        delay as required by exponential backoff algorithm;
        goto A
    }
    else {done with the frame; set collisions to zero}
}
else {wait until ongoing transmission is over and goto A}
```

0-144

Ethernet's CSMA/CD (more)

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

Exponential Backoff:

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

0-145

Interconnecting LANs

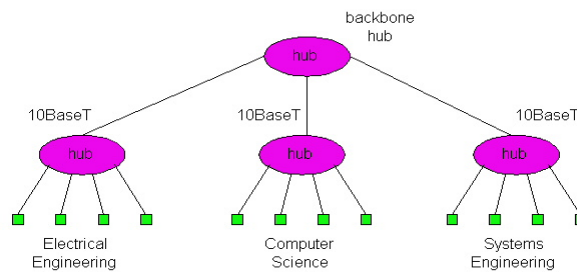
Q: Why not just one big LAN?

- ❑ Limited amount of supportable traffic: on single LAN, all stations must share bandwidth
- ❑ limited length: 802.3 specifies maximum cable length
- ❑ large "collision domain" (can collide with many stations)
- ❑ limited number of stations: 802.5 have token passing delays at each station

0-146

Hubs

- **Physical Layer devices**: essentially repeaters operating at bit levels: repeat received bits on one interface to all other interfaces
- Hubs can be arranged in a **hierarchy** (or multi-tier design), with **backbone** hub at its top



0-147

Hubs (more)

- Each connected LAN referred to as LAN **segment**
- Hubs **do not isolate** collision domains: node may collide with any node residing at any segment in LAN
- Hub Advantages:
 - simple, inexpensive device
 - Multi-tier provides graceful degradation: portions of the LAN continue to operate if one hub malfunctions
 - extends maximum distance between node pairs (100m per Hub)

0-148

Hub limitations

- ❑ single collision domain results in no increase in max throughput
 - multi-tier throughput same as single segment throughput
- ❑ individual LAN restrictions pose limits on number of nodes in same collision domain and on total allowed geographical coverage
- ❑ cannot connect different Ethernet types (e.g., 10BaseT and 100baseT)

0-149

Switches

- ❑ **Link Layer devices**: operate on Ethernet frames, examining frame header and **selectively** forwarding frame based on its destination
- ❑ Switch **isolates collision** domains since it buffers frames
- ❑ When frame is to be forwarded on segment, switch uses CSMA/CD to access segment and transmit

0-150

Switches (more)

- ❑ Switch advantages:
 - Isolates collision domains resulting in higher total max throughput, and does not limit the number of nodes nor geographical coverage
 - Can connect different type Ethernet since it is a store and forward device
 - Transparent: no need for any change to hosts LAN adapters

0-151

Switch: frame filtering, forwarding

- ❑ switches filter packets
 - same-LAN -segment frames not forwarded onto other LAN segments
- ❑ forwarding:
 - how to know which LAN segment on which to forward frame?
 - looks like a routing problem

0-152

Self-learning: Filtering and Forwarding

- switches *learn* which hosts can be reached through which interfaces: maintain filtering tables
 - when frame received, switch "learns" location of sender: incoming LAN segment
 - records sender location in filtering table
- filtering table entry:
 - (Node MAC Address, Switch Interface, Time Stamp)
 - stale entries in Filtering Table dropped (TTL can be 60 minutes)

0-153

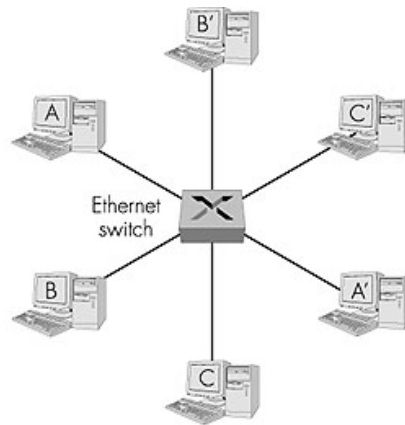
Filtering and Forwarding

- filtering procedure:
 - if** destination is on LAN on which frame was received
 - then** drop the frame
 - else {** lookup filtering table
 - if** entry found for destination
 - then** forward the frame on interface indicated;
 - else** flood; */* forward on all but the interface on which the frame arrived*/*
 - }**

0-154

Ethernet "Duplex" Switches

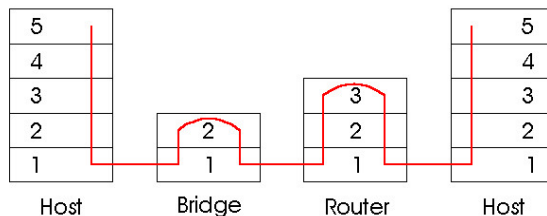
- ❑ layer 2 (frame) forwarding, filtering using LAN addresses
- ❑ **Switching:** A-to-B and A'-to-B' simultaneously, no collisions
- ❑ large number of interfaces
- ❑ often: individual hosts, star-connected into switch
 - Ethernet, but no collisions!



0-155

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 filtering tables, implement filtering, learning and spanning tree algorithms



0-156

Routers vs. Switches

Switches + and -

- + Switch operation is simpler requiring less processing bandwidth
- Topologies are restricted with switches: a spanning tree must be built to avoid cycles
- Switches do not offer protection from broadcast storms (endless broadcasting by a host will be forwarded by a switch)

0-157

Routers vs. Switches

Routers + and -

- + arbitrary topologies can be supported, cycling is limited by TTL counters (and good routing protocols)
 - + provide firewall protection against broadcast storms
 - require IP address configuration (not plug and play)
 - require higher processing bandwidth
- switches do well in small (few hundred hosts) while routers used in large networks (thousands of hosts)

0-158

Part 0: Networking Review

Goals:

- review key topics from intro networks course
 - equalize backgrounds
 - identify remedial work
 - ease into course

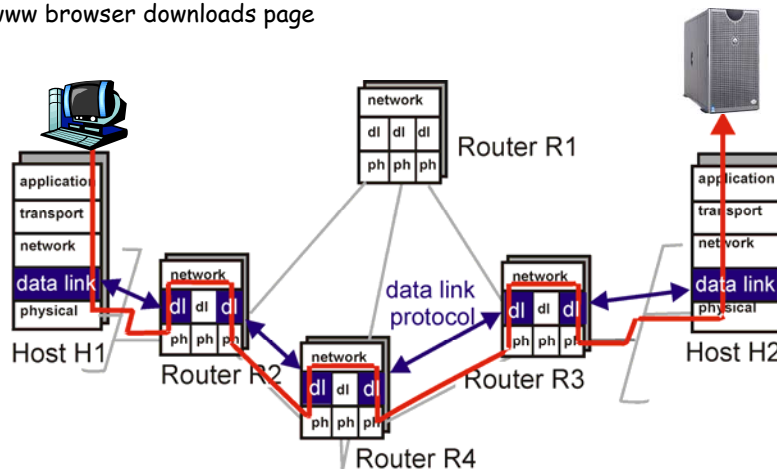
Overview:

- overview
- error control
- flow control
- congestion control
- routing
- LANs
- addressing (cont.)
- **synthesis:**
 - "a day in the life"

0-159

Synthesis: which protocols involved?

www browser downloads page



0-160

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., orchid.cs.utexas.edu - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers communicate to *resolve* names (address/name translation)
 - note: core Internet function, implemented as application-layer protocol
 - complexity at network's "edge" (e2e principle)

0-161

DNS name servers

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

- no server has all name-to-IP address mappings

local name servers:

- each ISP, company has *local (default) name server*
- host DNS query first goes to local name server

authoritative name server:

- for a host: stores that host's IP address, name
- can perform name/address translation for that host's name

0-162

DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
 - contacts authoritative name server if name mapping not known
 - gets mapping
 - returns mapping to local name server

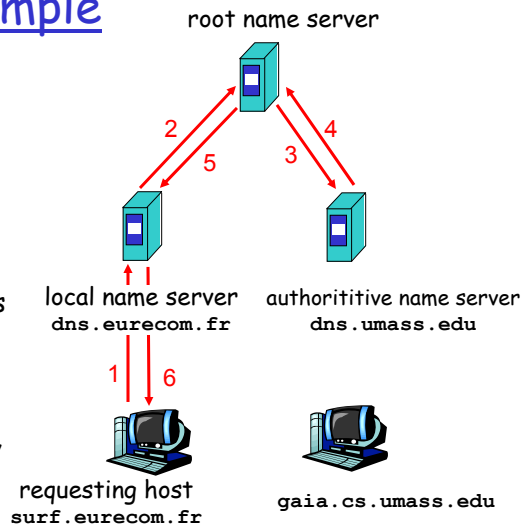


0-163

Simple DNS example

host **surf.eurecom.fr**
wants IP address of
gaia.cs.umass.edu

1. contacts its local DNS server, **dns.eurecom.fr**
2. **dns.eurecom.fr** contacts root name server, if necessary
3. root name server contacts authoritative name server, **dns.umass.edu**, if necessary

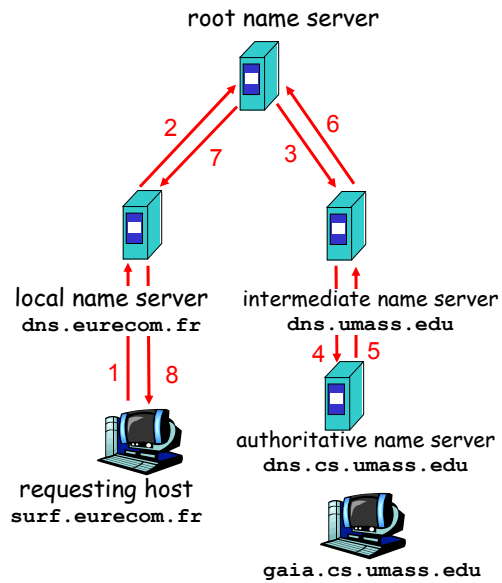


0-164

DNS example

Root name server:

- ❑ may not know authoritative name server
- ❑ may know *intermediate name server*: who to contact to find authoritative name server
- ❑ may need to first contact the TLD server



0-165

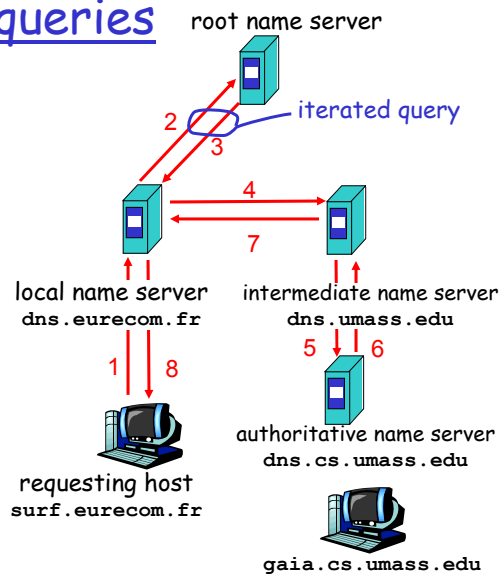
DNS: iterated queries

recursive query:

- ❑ puts burden of name resolution on contacted name server
- ❑ heavy load?

iterated query:

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



0-166

DNS records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

- Type=A
 - name is hostname
 - value is IP address
- Type=CNAME
 - name is alias name for some "canonical" (the real) name
www.ibm.com is really
servereast.backup2.ibm.com
 - value is canonical name
- Type=NS
 - name is domain (e.g. foo.com)
 - value is IP address of authoritative name server for this domain
- Type=MX
 - value is name of mailserver associated with name

Once a name server learns mapping, it *caches* mapping

0-167

Reverse DNS lookup

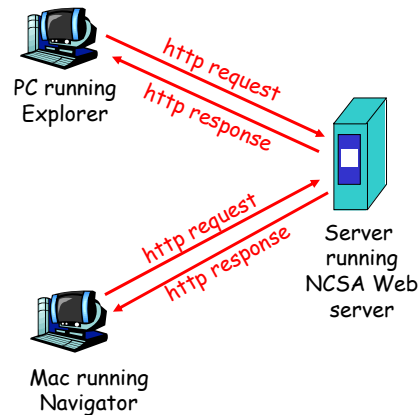
- Q: what's the host name for 192.0.2.25?
- DNS resolver's actions
 - Query local DNS server for PTR record for 25.2.0.192.in-addr.arpa
 - Query root DNS server for 25.2.0.192.in-addr.arpa
 - Query DNS server for 192.in-addr.arpa
 - Query DNS server of Regional Internet Registry (RIR)
 - ARIN: North America, APNIC: Asian-Pacific, RIPE: Europe
 - Query DNS server of the organization that was originally given the IP range
 - Often the DNS server of your ISP
 - Query the DNS server of the real owner organization
 - Get the answer

0-168

The Web: the http protocol

http: hypertext transfer protocol

- ❑ Web's application layer protocol
- ❑ client/server model
 - *client*: browser that requests, receives, "displays" Web objects
 - *server*: Web server sends objects in response to requests
- ❑ http1.0: RFC 1945
- ❑ http1.1: RFC 2068



0-169

The http protocol: more

http: TCP transport service:

- ❑ client initiates TCP connection (creates socket) to server, port 80
- ❑ server accepts TCP connection from client
- ❑ http messages (application-layer protocol messages) exchanged between browser (http client) and Web server (http server)
- ❑ TCP connection closed

http is "stateless"

- ❑ server maintains no information about past client requests

Protocols that maintain "state" are complex! aside

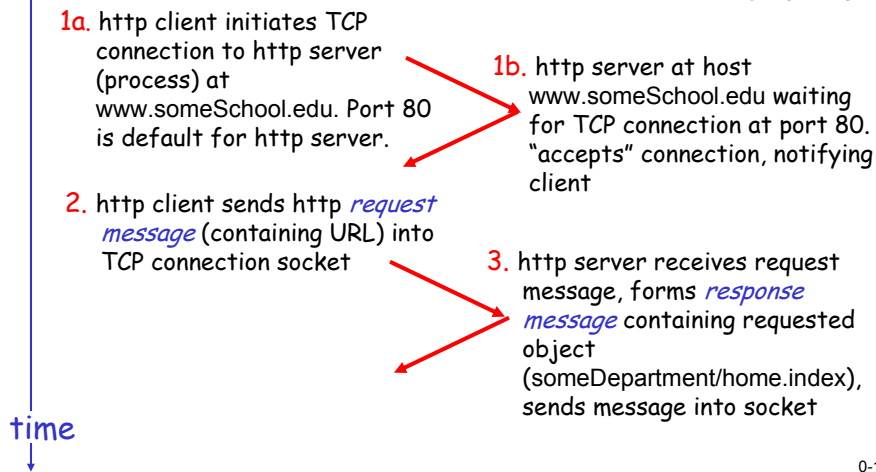
- ❑ past history (state) must be maintained
- ❑ if server/client crashes, their views of "state" may be inconsistent, must be reconciled

0-170

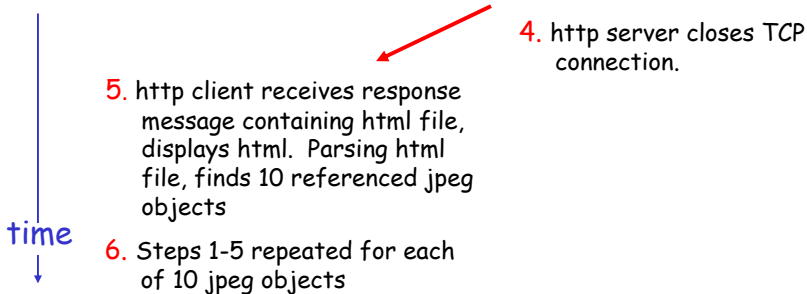
http example

Suppose user enters URL

www.someSchool.edu/someDepartment/home.index (contains text, references to 10 jpeg images)



http example (cont.)



Non-persistent, persistent connections

Non-persistent

- ❑ http/1.0: server parses request, responds, closes TCP connection
- ❑ 2 RTTs to fetch object
 - TCP connection
 - object request/transfer
- ❑ each transfer suffers from TCP's initially slow sending rate
- ❑ many browsers open multiple parallel connections

Persistent

- ❑ default for http/1.1
- ❑ on same TCP connection: server, parses request, responds, parses new request,..
- ❑ client sends requests for all referenced objects as soon as it receives base HTML.
- ❑ fewer RTTs, less slow start.
- ❑ Pipelining can further reduce RTTs

0-173

http message format: request

- ❑ two types of http messages: *request, response*
- ❑ **http request message:**
 - ASCII (human-readable format)

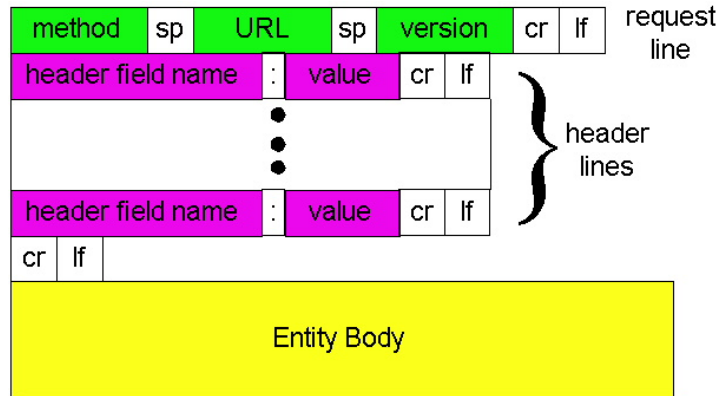
request line
(GET, POST,
HEAD commands) → GET /somedir/page.html HTTP/1.0

header
lines → Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: fr

Carriage return,
line feed
indicates end
of message → (extra carriage return, line feed)

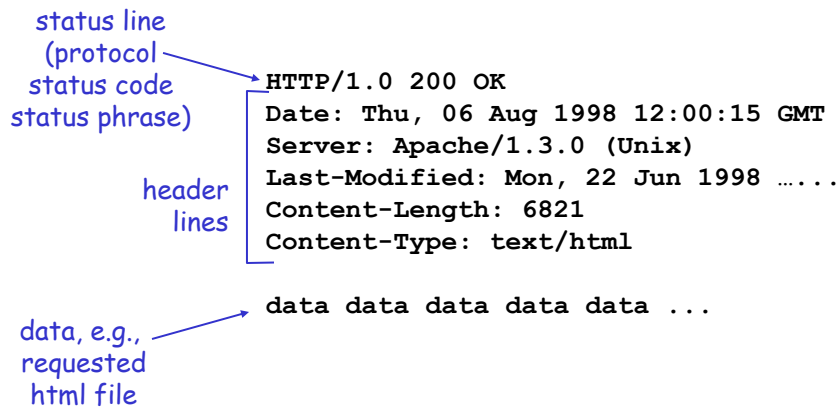
0-174

http request message: general format



0-175

http message format: response



0-176

http response status codes

In first line in server→client response message.

A few sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

- requested object moved, new location specified later in this message (Location:)

400 Bad Request

- request message not understood by server

404 Not Found

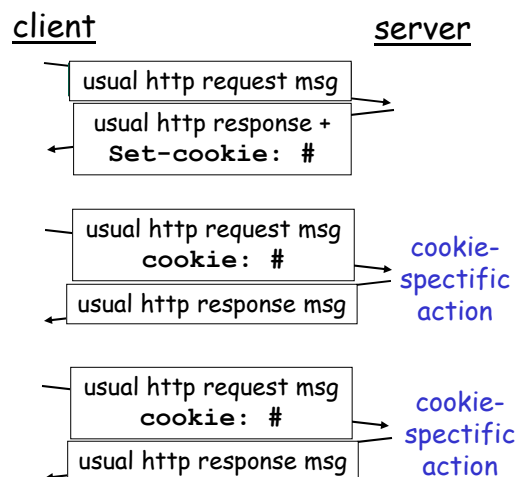
- requested document not found on this server

505 HTTP Version Not Supported

0-177

Cookies: keeping "state"

- server-generated # , server-remembered # , later used for:
 - authentication
 - remembering user preferences, previous choices
- server sends "cookie" to client in response msg
Set-cookie: 1678453
- client presents cookie in later requests
cookie: 1678453

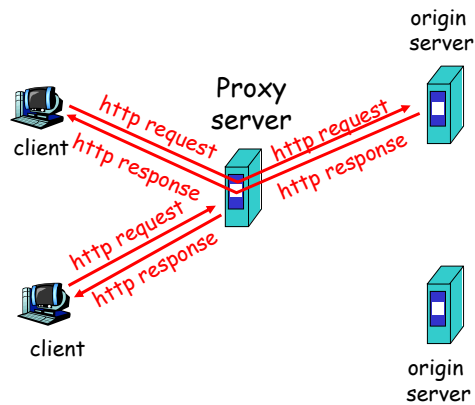


0-178

Web Caches (proxy server)

Goal: satisfy client request without involving origin server

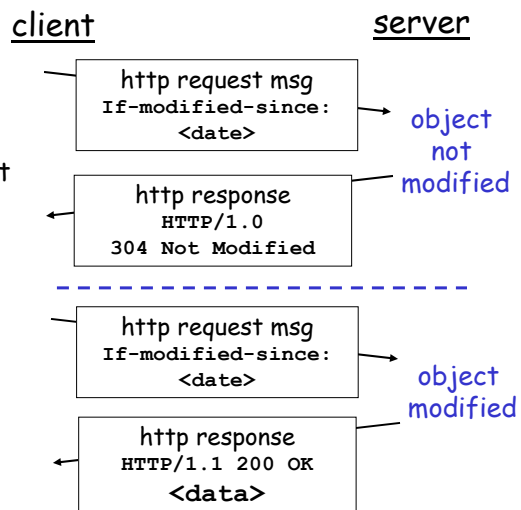
- user sets browser: Web accesses via web cache
- client sends all http requests to web cache
 - object in web cache: web cache returns object
 - else web cache requests object from origin server, then returns object to client



0-179

Conditional GET: client-side caching

- **Goal:** don't send object if client has up-to-date cached version
- client: specify date of cached copy in http request
If-modified-since: <date>
- server: response contains no object if cached copy is up-to-date:
HTTP/1.0 304 Not Modified



0-180

Protocols involved in http GET

- User types in a URL, what happens?
- DNS: translate hostname to IP address
 - Via DHCP, source has IP address of DNS server (suppose DNS server is on same network segment)
 - Create DNS query, pass to UDP, create UDP segment containing DNS query, pass to IP on host
 - Look in routing table (DHCP gave me default router), recognize that DNS server is on same network.
 - Use ARP to determine MAC address of DNS server
 - Ethernet used to send frame to DNS server on physically connected "wire" (network segment, Ethernet "cable")
 - On DNS machine Ethernet->IP->UDP. UDP looks at dest port #, sees it is DNS, passes DNS query to DNS application. (assume dns knows IP addresses of hostname in original URL - address found!)
 - DNS server sends UDP reply back to originating machine

0-181

Protocols involved in http GET

- So: browser now has IP address of GET destination server
- Need to establish TCP connection to server, TCP connection establishment sends SYN packet (will get an SYNACK back, eventually....)
 - SYN packet down to network layer, with IP address of server. Since server destined "off my network", SYN packet will need to go through router.
 - Look in routing table, see that destined off network, need to send to "default gateway" (to get off my net)
 - Use ARP to get MAC address of default gateway, create Ethernet frame with gateway MAC address, containing IP packet containing TCP segment, containing SYN
 - It is IMPORTANT to realize that while the Ethernet frame containing the IP datagram that contains the TCP SYN has as its destination address the MAC address of the router, the IP datagram (still) has as its destination address the IP address of the remote www server

0-182

Protocols involved in http GET

- ❑ Router receives Ethernet frame (frame is addressed to router), looks at IP datagram and sees that IP datagram is not addressed to itself (IP datagram is addressed to server). So router knows that it must forward the IP datagram to the next hop router along the path to the eventual destination.
- ❑ Router checks routing tables (table values populated using intra, possibly inter-, domain routing protocols like OSPF, RIP, IS-IS, BGP (inter). Get IP address of next hop router.
- ❑ Router puts IP packets in Ethernet frame, Ethernet frame addressed to next hop router. MAC address of next hop router determined by ARP. Frame sent to next hop router.
- ❑ This forwarding continues until IP datagram contain TCP SYN eventually arrives at destination, XYZ.cs.utexas.edu
- ❑ Up to IP, demultiplex from Ethernet to IP using Ethernet TYPE field to identify IP as upper layer protocol
- ❑ From IP to TCP using protocol field of IP datagram,
- ❑ SYN packet arrives at XYZ TCP (FINALLY)

0-183

Protocols involved in http GET

- ❑ So SYN has arrived at XYZ. XYZ sends back SYNACK to initial sender
- ❑ Sender gets synack, ready to send data.
- ❑ HTTP GET message now sent to XYZ.cs.utexas.edu in a TCP segment, in IP datagram, Ethernet frame, along hops to XYZ.cs.utexas.edu
- ❑ GET arrives! REPLY formulated by http server ... and sent

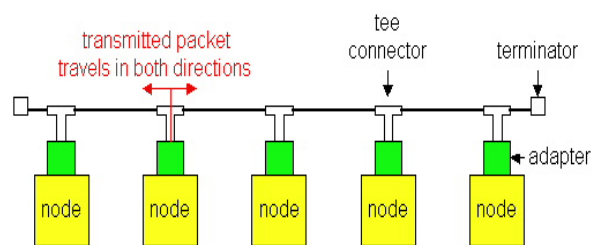
0-184

Additional Slides

0-185

Ethernet Technologies: 10Base2

- 10: 10Mbps; 2: under 200 meters max cable length
- thin coaxial cable in a bus topology

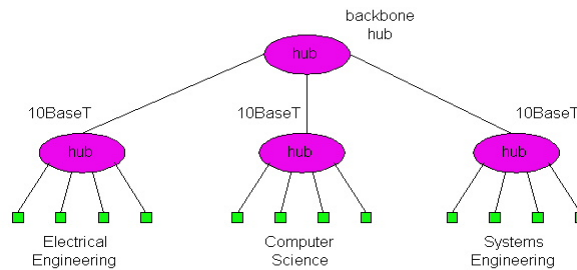


- repeaters used to connect up to multiple segments
- repeater repeats bits it hears on one interface to its other interfaces: physical layer device only!

0-186

10BaseT and 100BaseT

- ❑ 10/100 Mbps rate; latter called "fast ethernet"
- ❑ T stands for Twisted Pair
- ❑ Hub to which nodes are connected by twisted pair, thus "star topology"
- ❑ CSMA/CD implemented at hub



0-187

Gbit Ethernet

- ❑ use standard Ethernet frame format
- ❑ allows for point-to-point links and shared broadcast channels
- ❑ in shared mode, CSMA/CD is used; short distances between nodes to be efficient
- ❑ uses hubs, called here "Buffered Distributors"
- ❑ Full-Duplex at 1 Gbps for point-to-point links

0-188