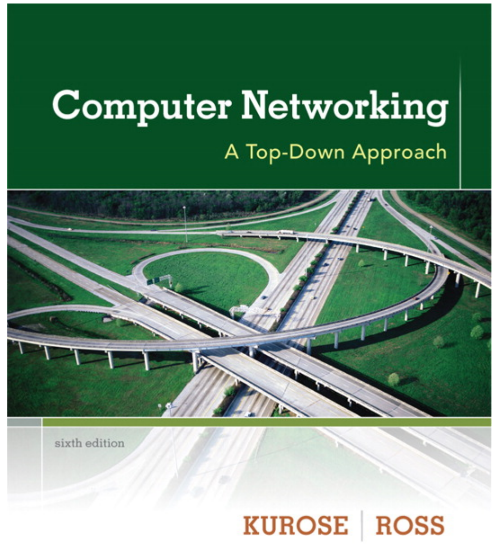
Chapter 4
Network Layer



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Chapter 4: network layer

chapter goals:

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast
- instantiation, implementation in the Internet

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP
- IPv6

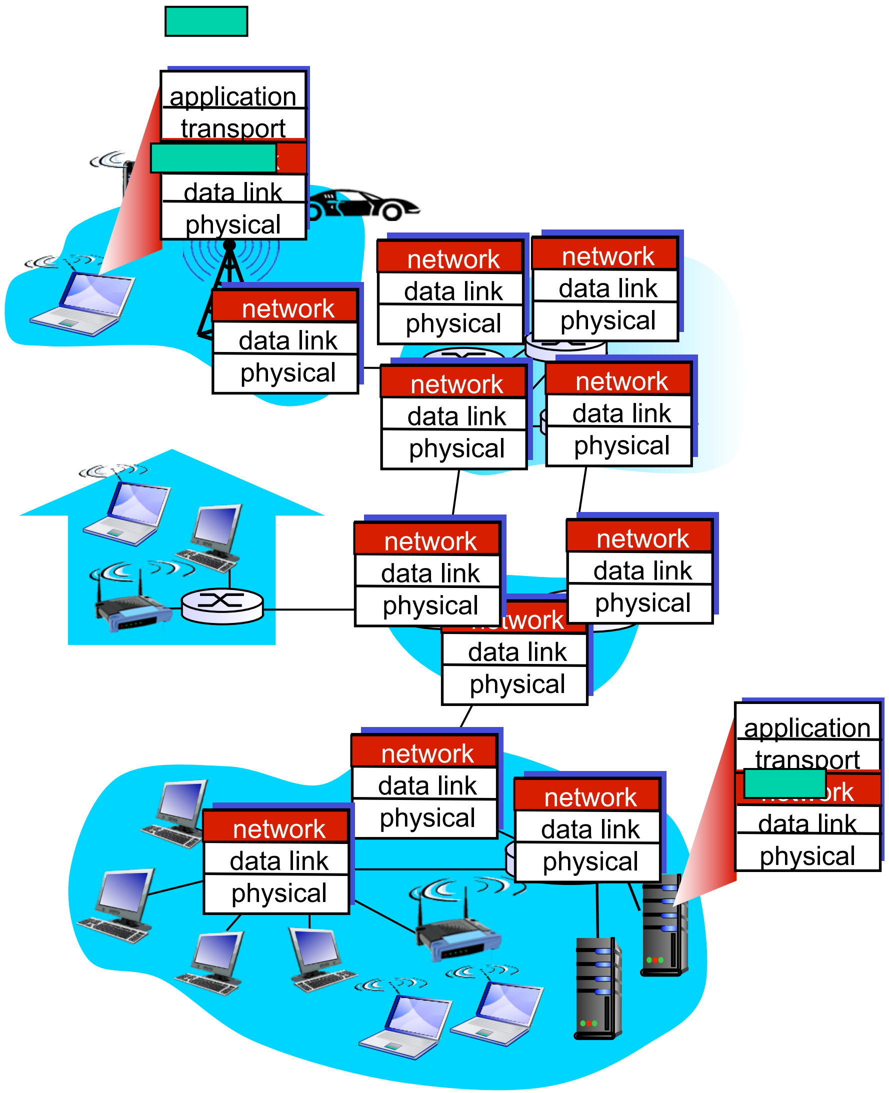
4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet (intra-AS routing)

4.7 broadcast and multicast routing

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer



- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it

```
⊞ Frame 90: 760 bytes on wire (6080 bits), 760 bytes captured (6080 bits) on interface 0
⊞ Ethernet II, Src: QuantaCo_02:eb:19 (00:1e:68:02:eb:19), Dst: Cisco_d5:79:ff (00:14:6a:d5:79:ff)
⊞ Internet Protocol Version 4, Src: 10.60.80.213 (10.60.80.213), Dst: 161.139.21.50 (161.139.21.50)
⊞ Transmission Control Protocol, Src Port: nms-dpnss (2503), Dst Port: http (80), Seq: 1, Ack: 1, Len: 706
⊞ Hypertext Transfer Protocol
```

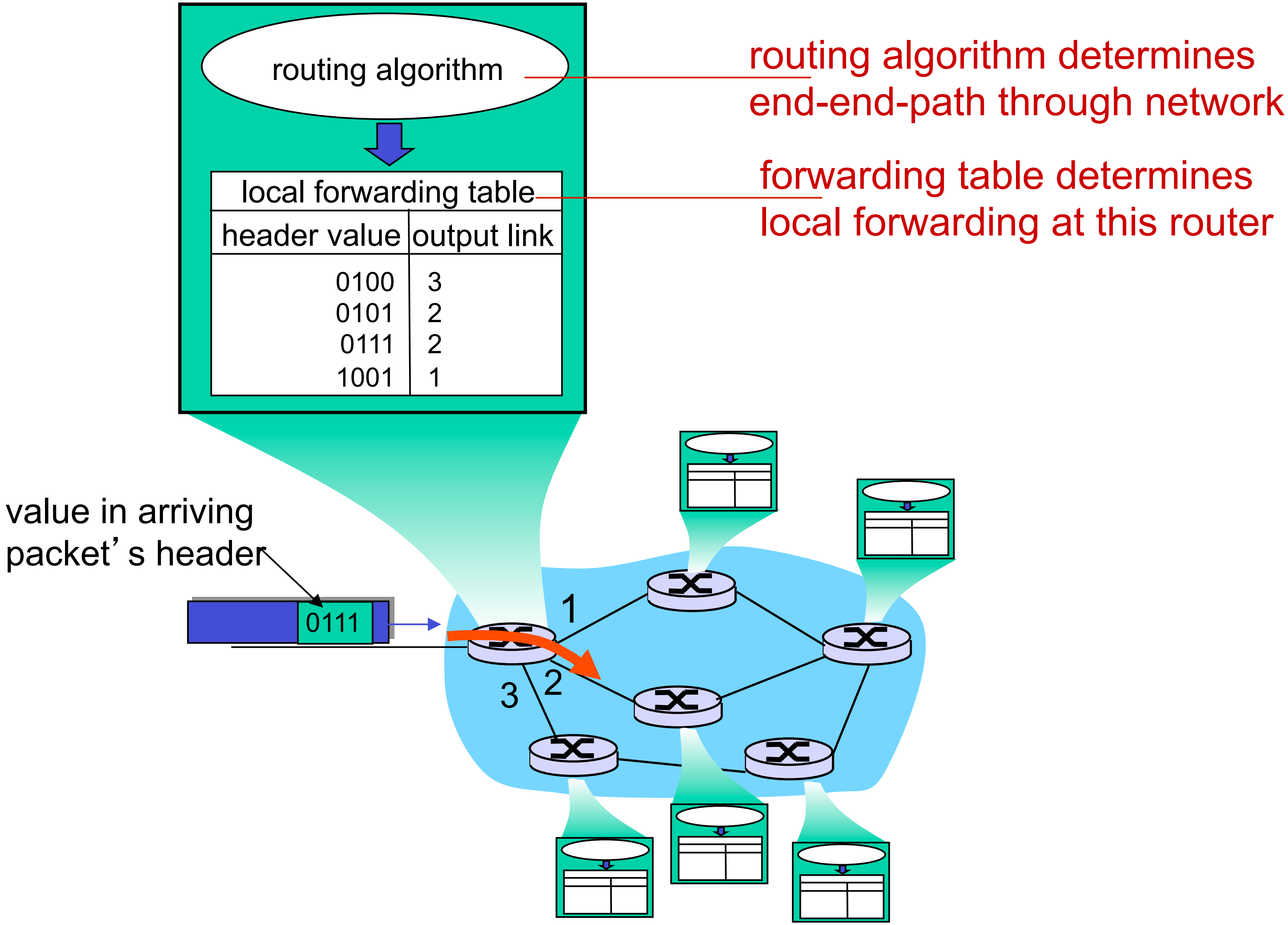
Two key network-layer functions

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

analogy:

- *routing:* process of planning trip from source to dest
- *forwarding:* process of getting through single interchange

Interplay between routing and forwarding



3rd function: Connection setup

- 3rd important function in *some* network architectures:
  - Applied in ATM, frame relay, X.25
  - Not applied in Internet
- before datagrams flow, two end hosts *and* intervening routers establish virtual connection (VC)
  - routers get involved
- network vs transport layer connection service:
  - *network:* between two hosts (may also involve intervening routers in case of VCs)
  - *transport:* between two processes

Network service model

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

example services for individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network layer service models:

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Asynchronous Transfer Mode (ATM) - Network Architecture
CBR: Constant Bit Rate, VBR=Variable Bit Rate, ABR=Available Bit Rate, UBR= Unspecified Bit Rate

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet (intra-AS routing)

4.7 broadcast and multicast routing

Connection, connection-less service

- *datagram* network provides network-layer *connectionless* service (e..g apply in Internet architecture)
- *virtual-circuit (VC)* network provides network-layer *connection* service (e.g. apply in ATM architecture)
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service:* host-to-host
  - *no choice:* network provides one or the other
  - *implementation:* in network core

Virtual circuits (VC connection)

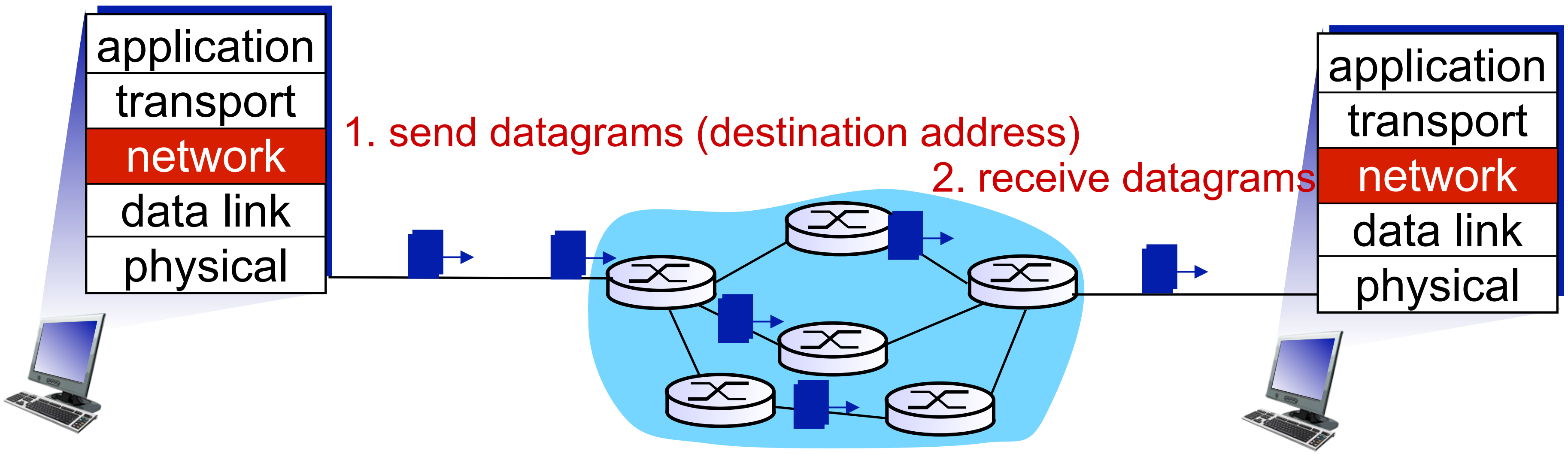
"source-to-dest path behaves much like telephone circuit"

- performance-wise
- network actions along source-to-dest path

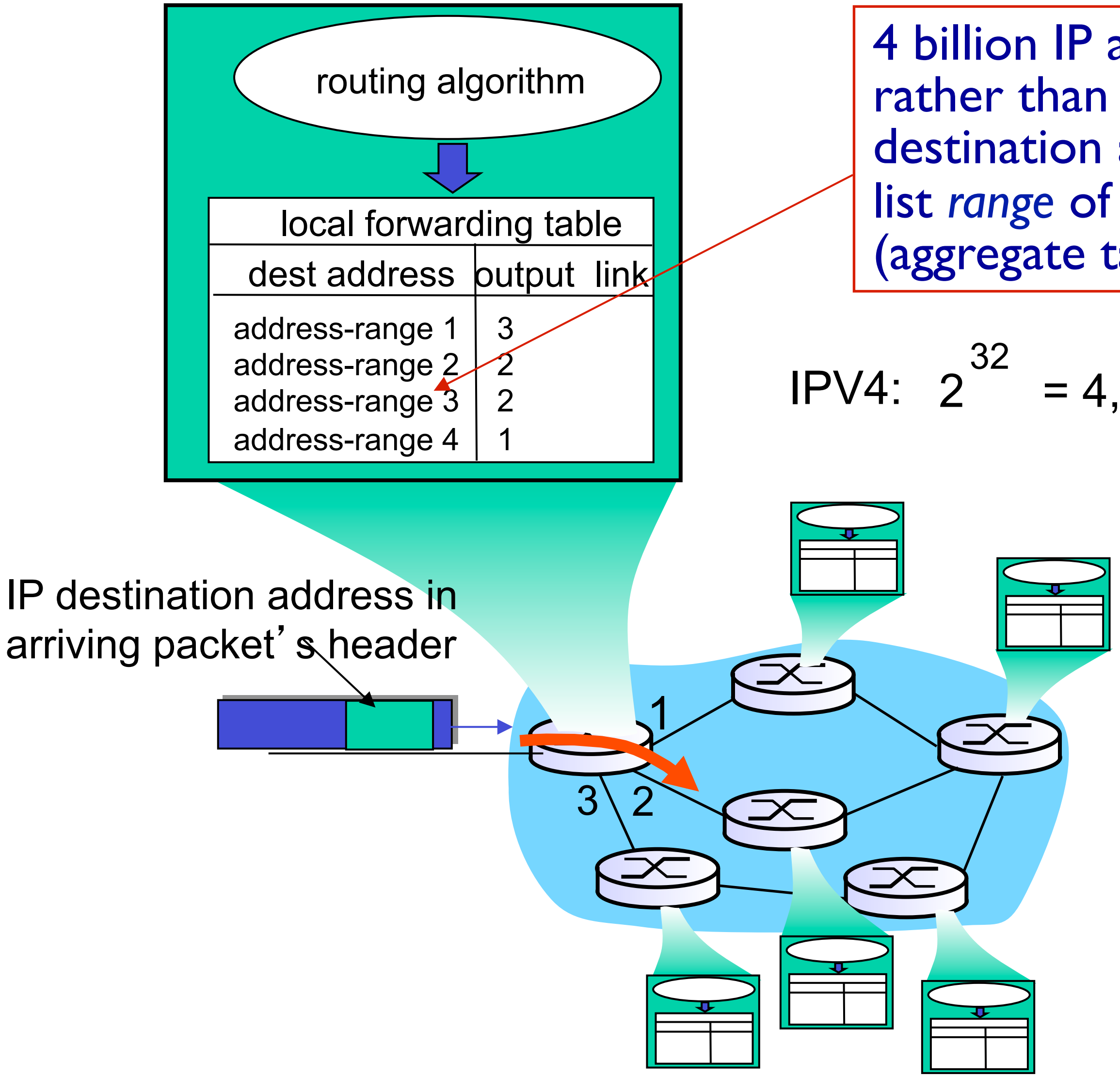
- call setup, teardown for each call *before* data can flow
  - Have 3 connection phase:
    - *VC setup(i), Data transfer(ii), VC teardown (iii)*
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

Datagram networks (connectionless)

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



Datagram forwarding table



4 billion IP addresses, so rather than list individual destination address list *range* of addresses (aggregate table entries)

IPV4: $2^{32} = 4{,}294{,}967{,}296$

Datagram forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Datagram or VC network: why?

Internet (datagram)

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

ATM (VC)

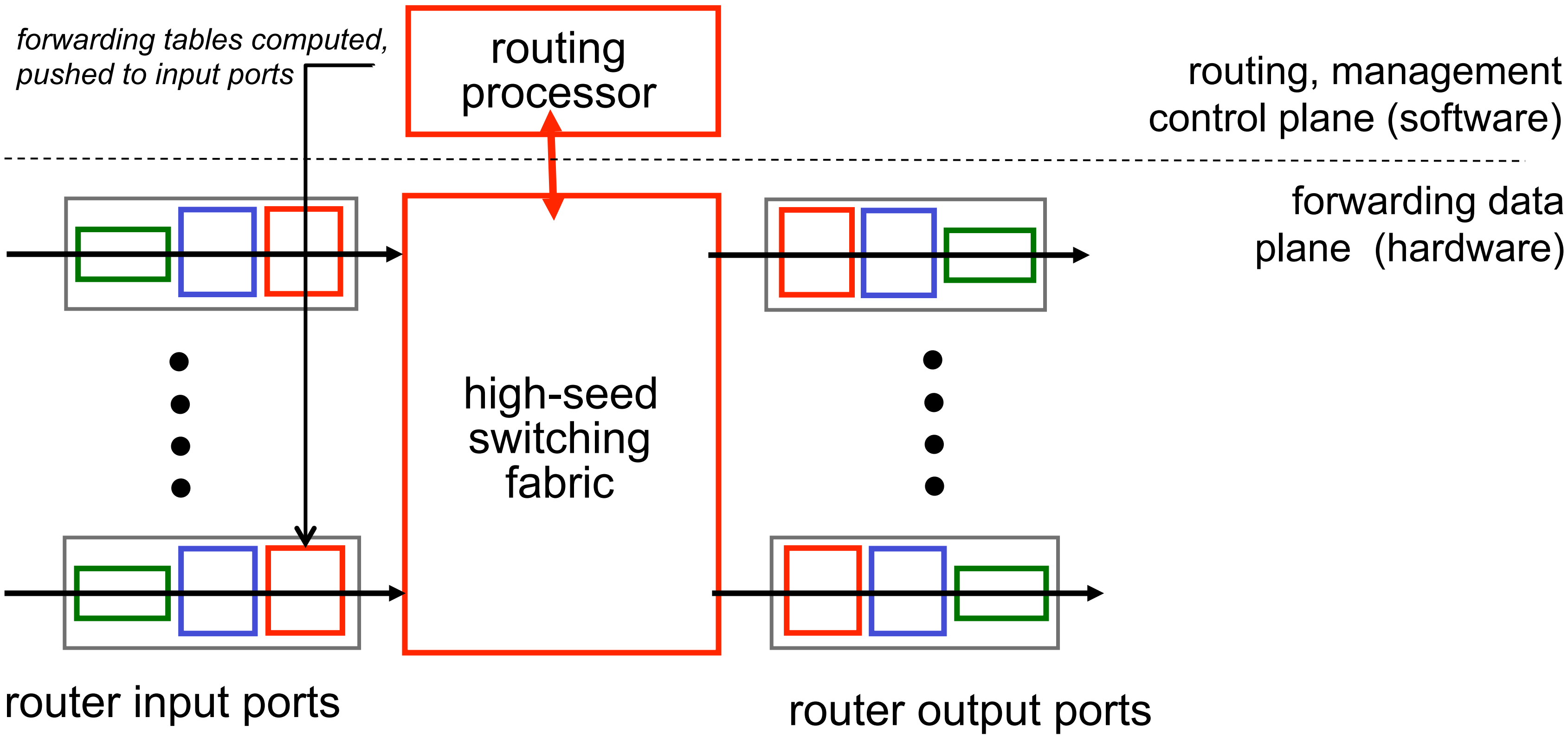
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - ***complexity inside network***

Chapter 4: outline

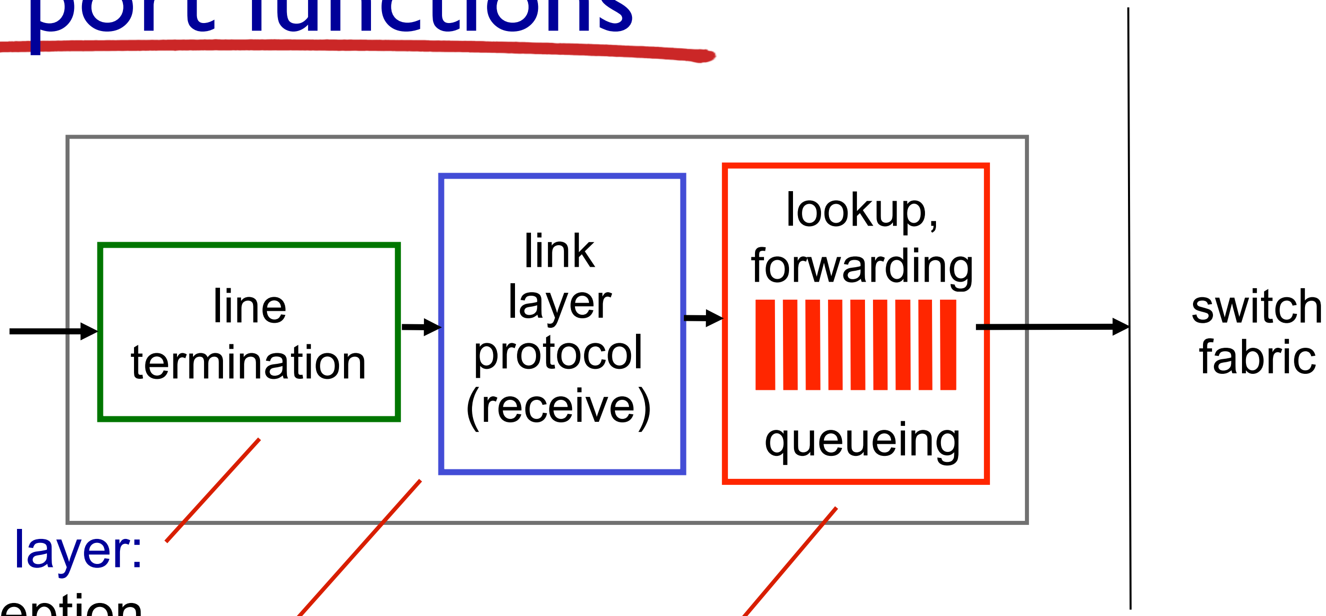
Router architecture overview

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



Input port functions



physical layer:
bit-level reception

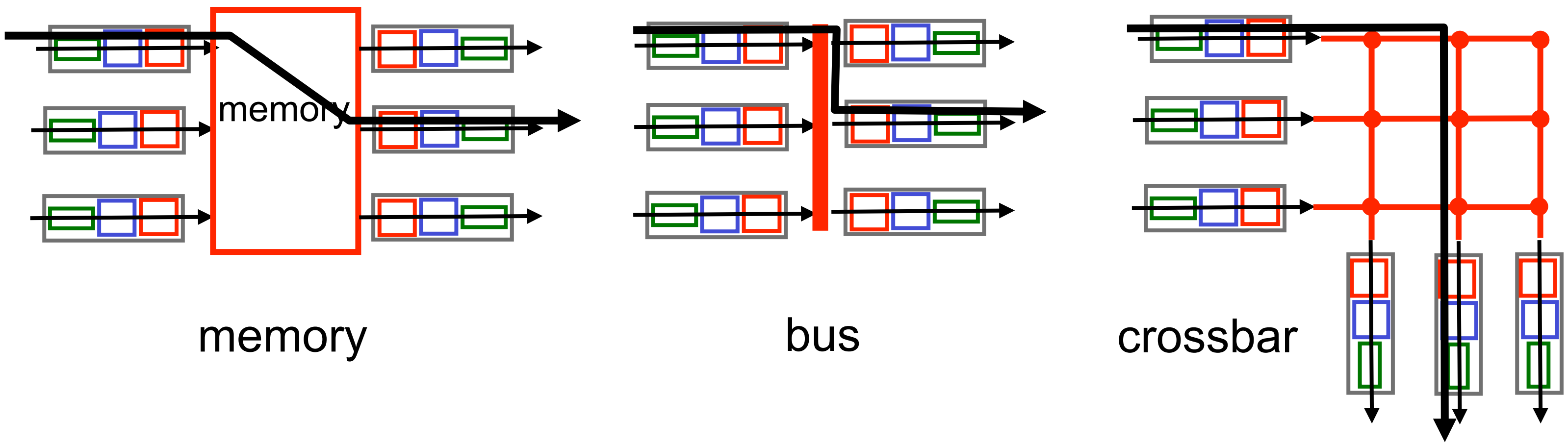
data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

- given datagram dest., lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Switching fabrics

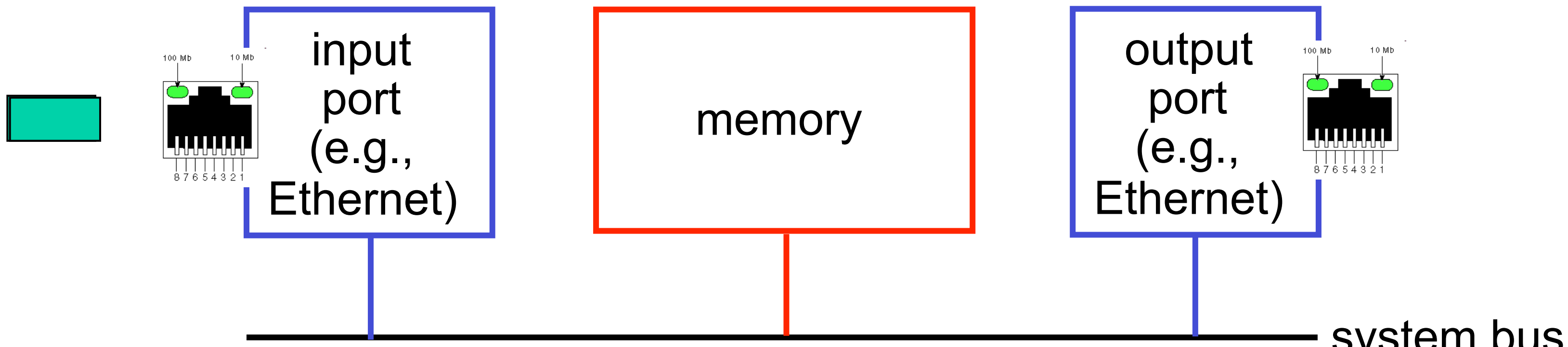
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate = N times line rate desirable
- three types of switching fabrics



Switching via memory

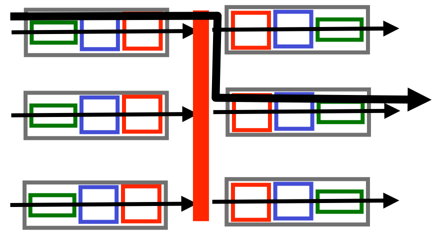
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via a bus

- datagram from input port memory to output port memory via a single shared bus
- *bus contention (conflict):* switching speed limited by bus bandwidth
  - e.g. 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

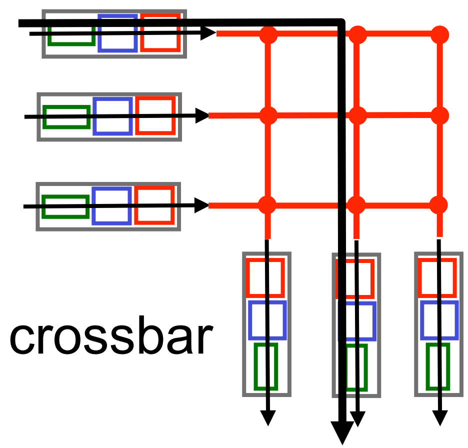


bus

Switching via interconnection network

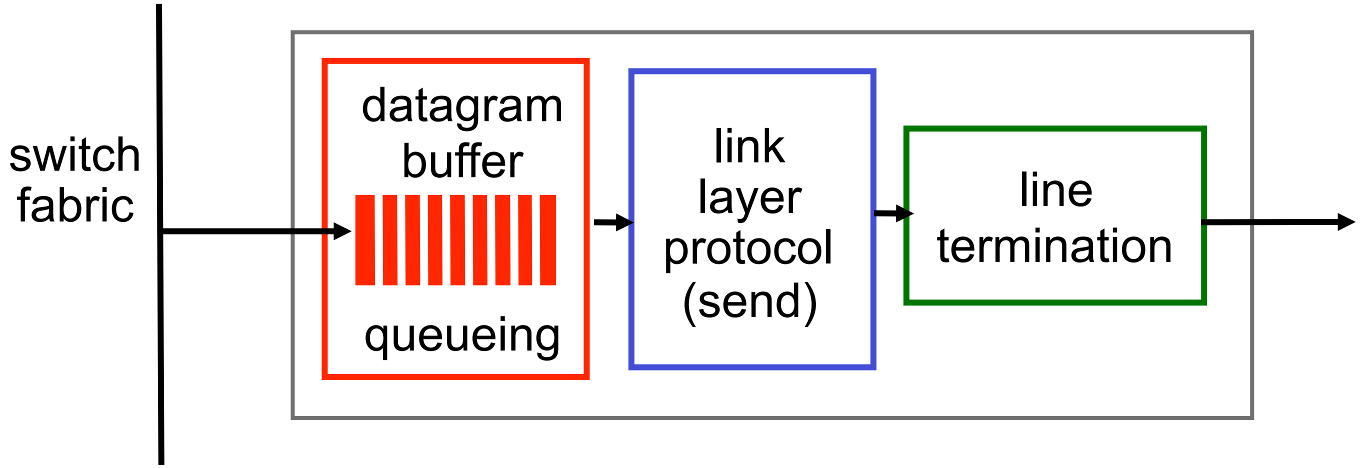
- overcome bus bandwidth limitations (single shared bus)
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design (to speedup switching)
  - ✓ fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

Banyan networks:
N input x N output



Re-assemble at output port

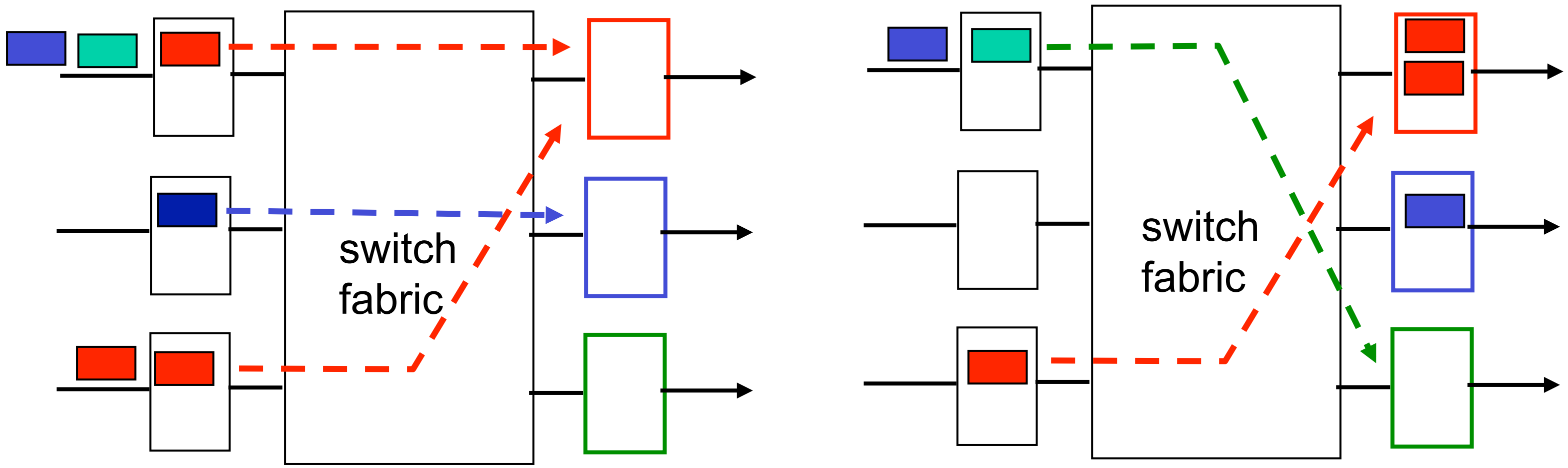
Output ports



- *buffering* required when datagrams arrive from fabric faster than the transmission rate
- *scheduling discipline* chooses among queued datagrams for transmission

Output port queueing

Packet scheduler:
e.g. FCFS or WFQ



at *t*, packets more from input to output

one packet time later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss (or drop tail) due to output port buffer overflow!*
  - *QoS will be affected*

How much buffering (buffer size) is needed?

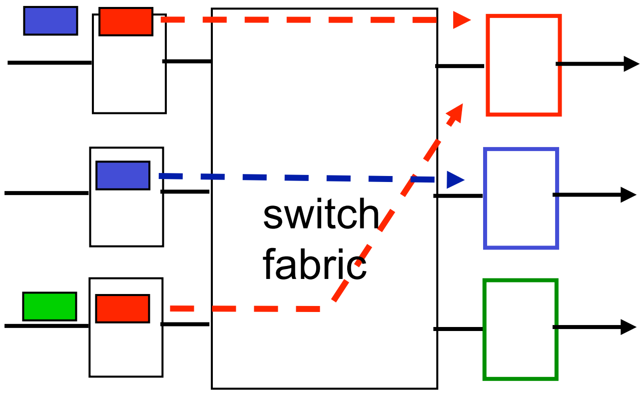
- RFC 3439 rule of thumb:
  - average buffering size = RTT * link capacity C
  - e.g., RTT=250 msec ("typical"), C = 10 Gpbs link
    - average buffering size =250 msec * 10 Gpbs =2.5 Gbit buffer

- recent recommendation: with *N* flows, buffering equal to

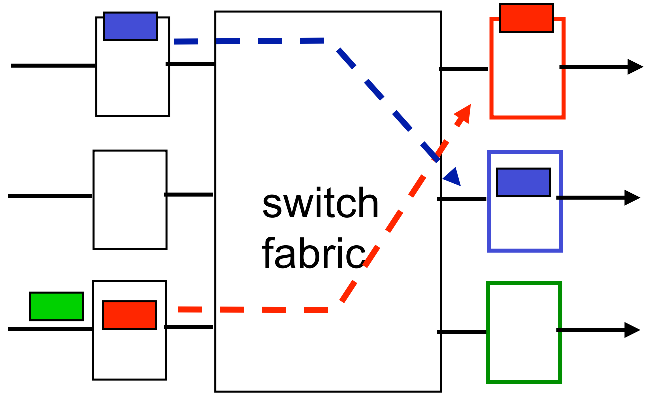
$$\frac{RTT \cdot C}{\sqrt{N}}$$

Input port queuing

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



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

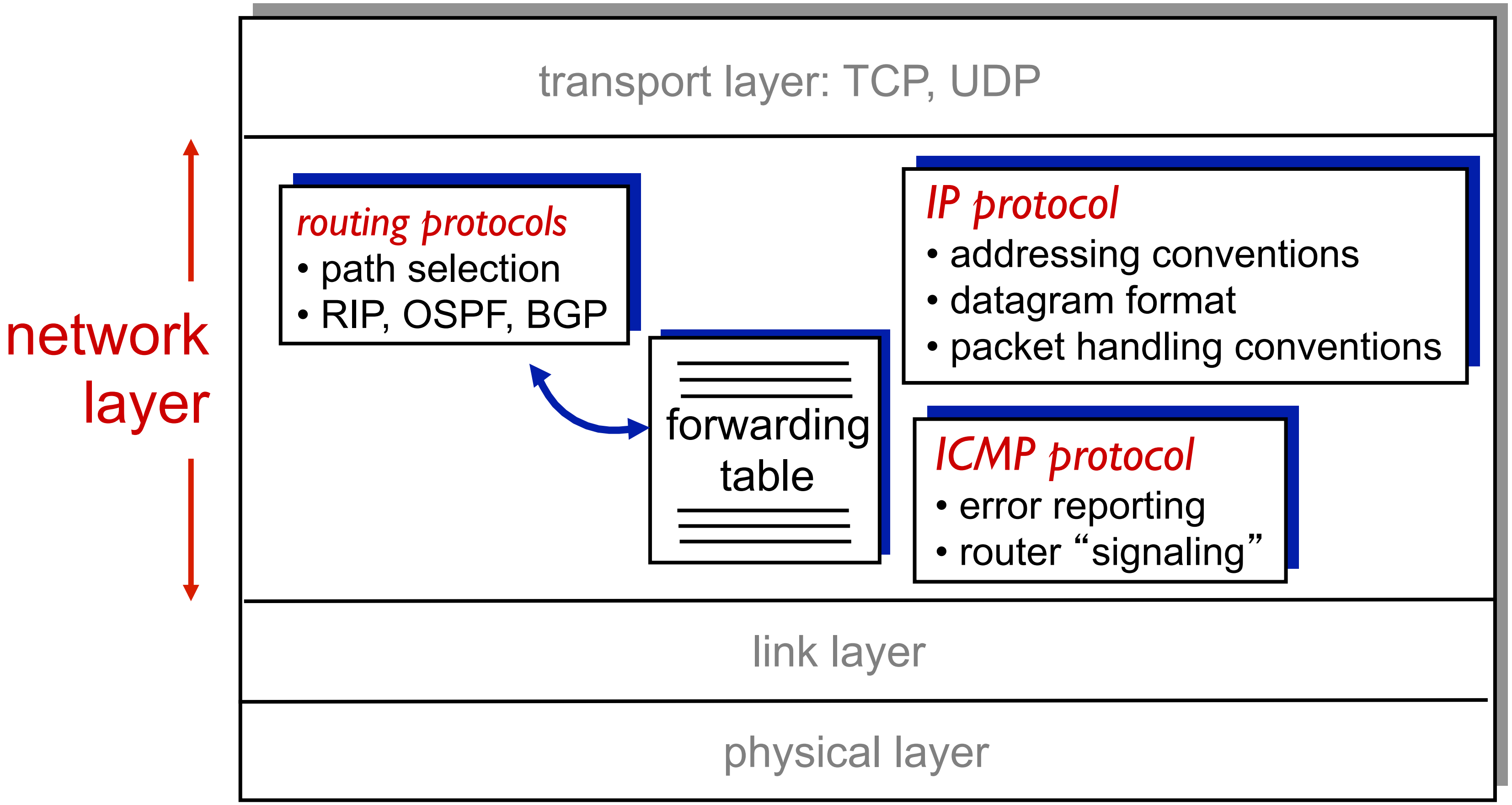


one packet time later: green packet experiences HOL blocking

Chapter 4: outline

The Internet network layer

host, router network layer functions:



IP datagram format: IPv4

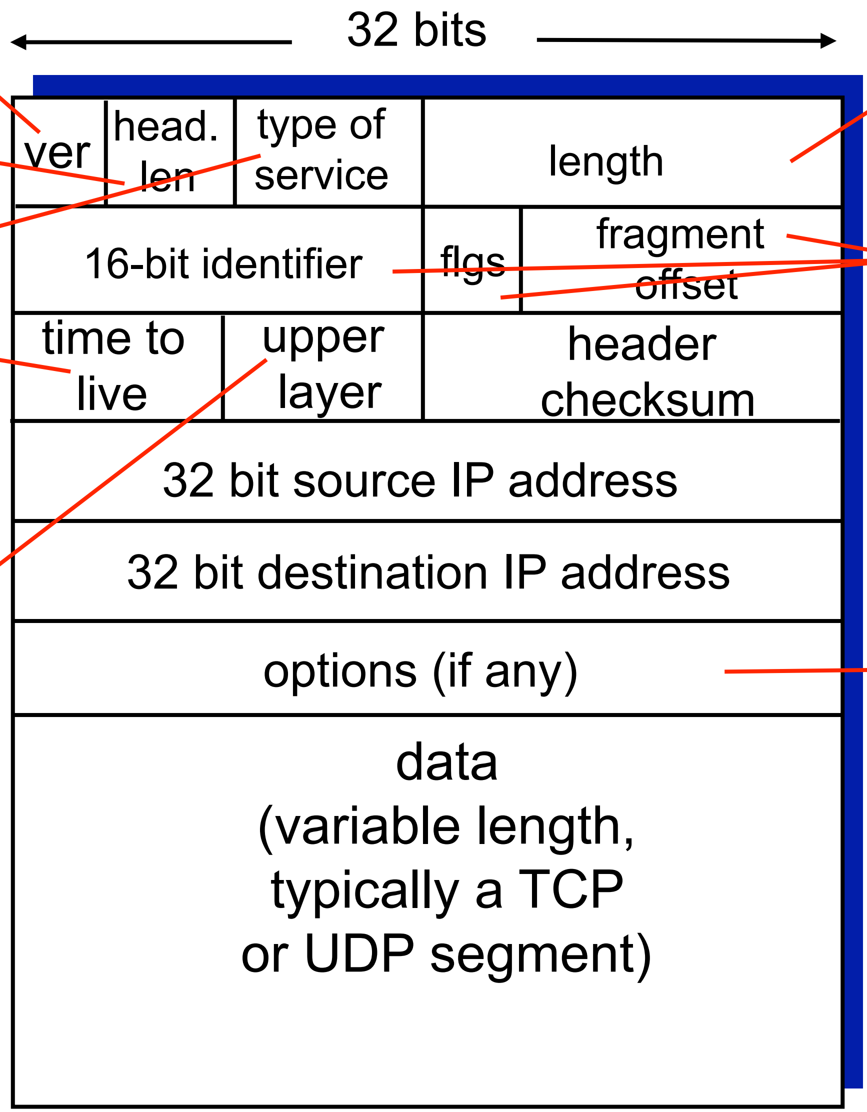
IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to



total datagram length (bytes)

for fragmentation/ reassembly

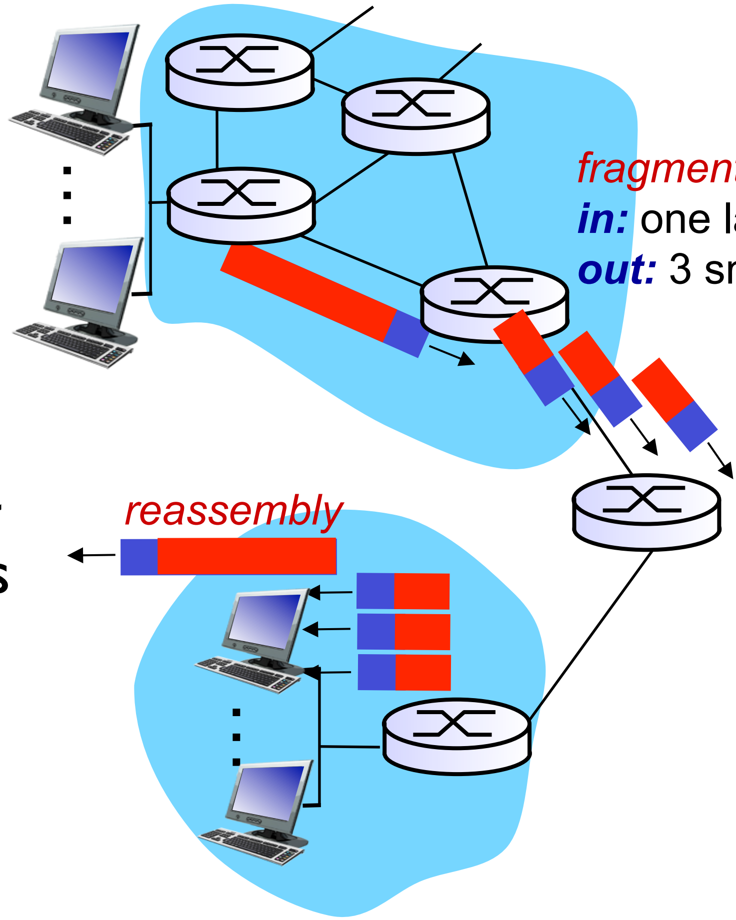
e.g. timestamp, record route taken, specify list of routers to visit.

how much overhead?

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

IP fragmentation, reassembly

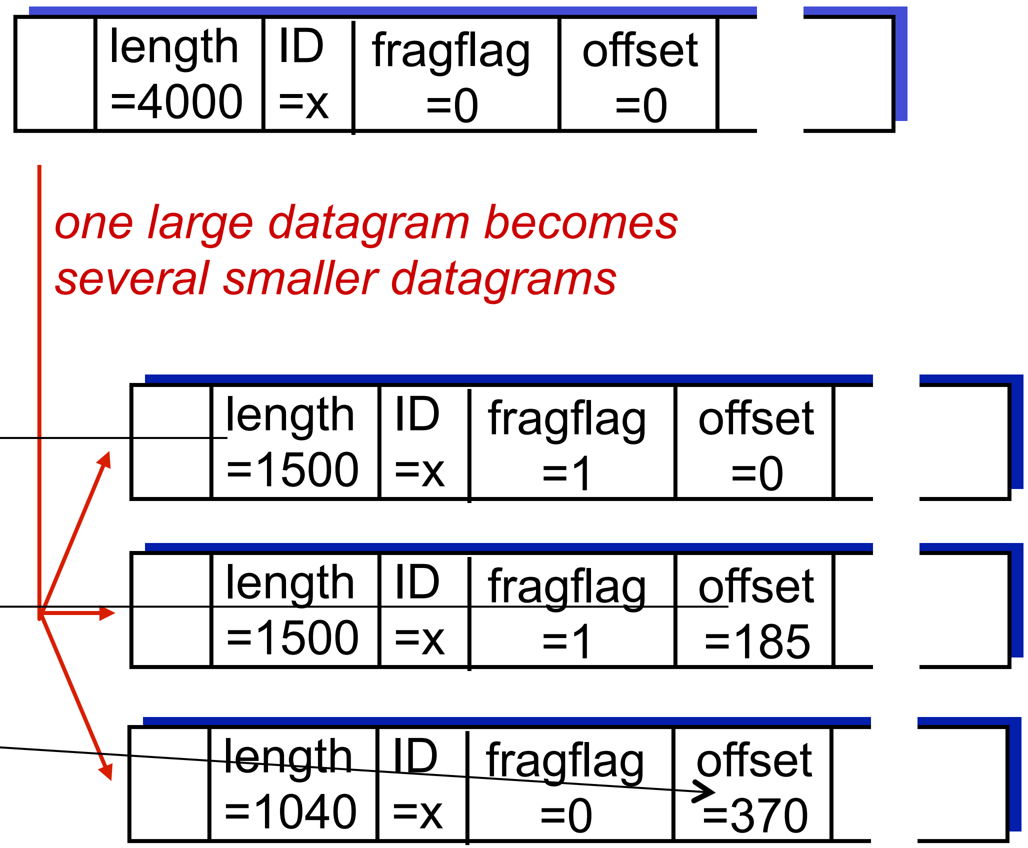
- network links have MTU (max transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



1480 bytes in data field

offset = 1480/8

offset = 185+185 =370

- 4000 byte datagram
  - Arrived Byte = 4000Byte
  - IP header =20Byte
  - Data Byte Remainder
    - 4000 Bytes – 20 Bytes = 3980 bytes

- Total datagram = 1480 + 1480 + 1020 = 3980 bytes

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

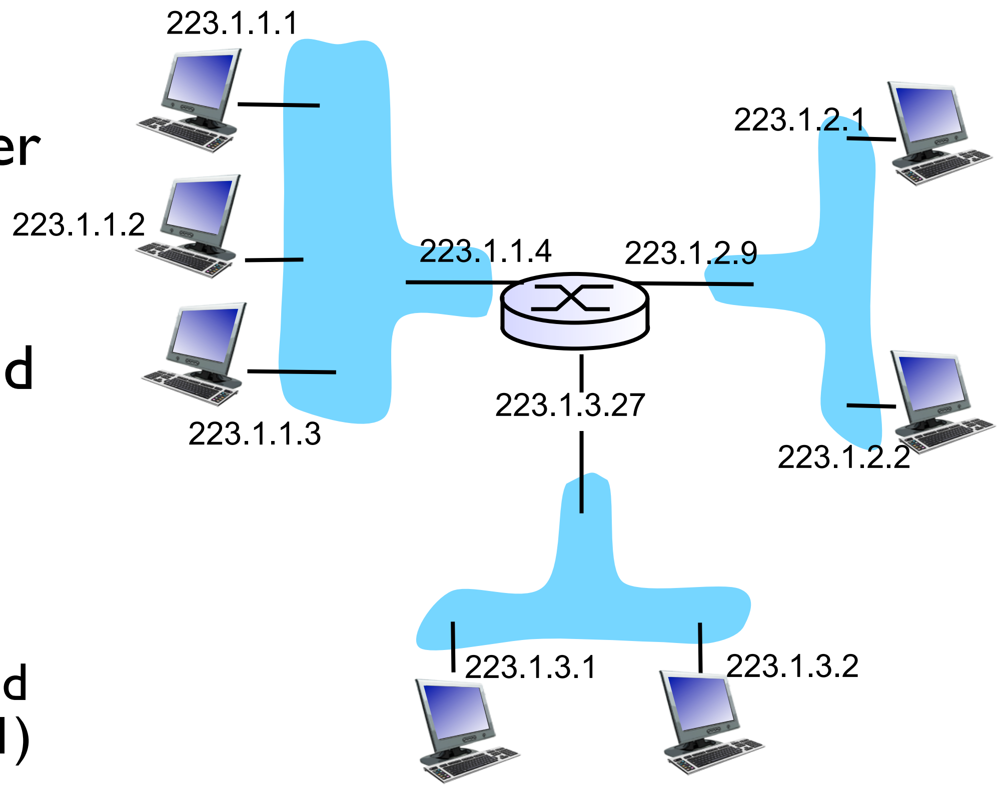
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet (intra-AS routing)

4.7 broadcast and multicast routing

IP addressing: introduction

- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *One IP addresses associated with each interface*



223.1.1.1 = 11011111 00000001 00000001 00000001

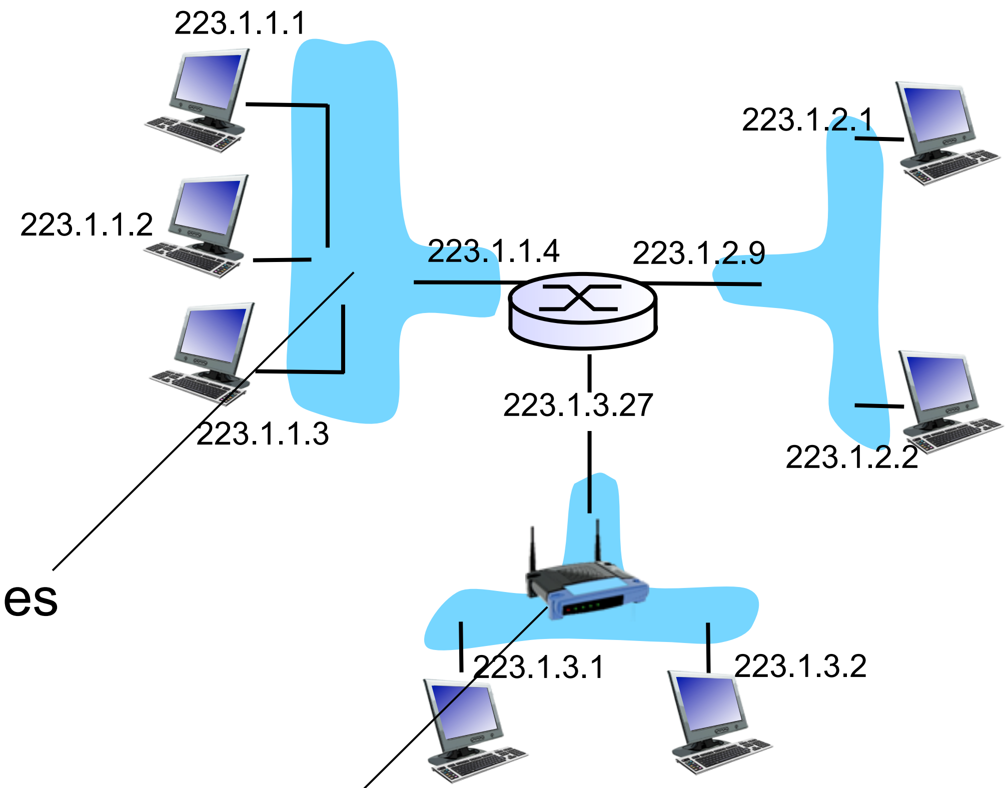
223 1 1 1

Dotted-decimal notation

IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapter 5, 6.



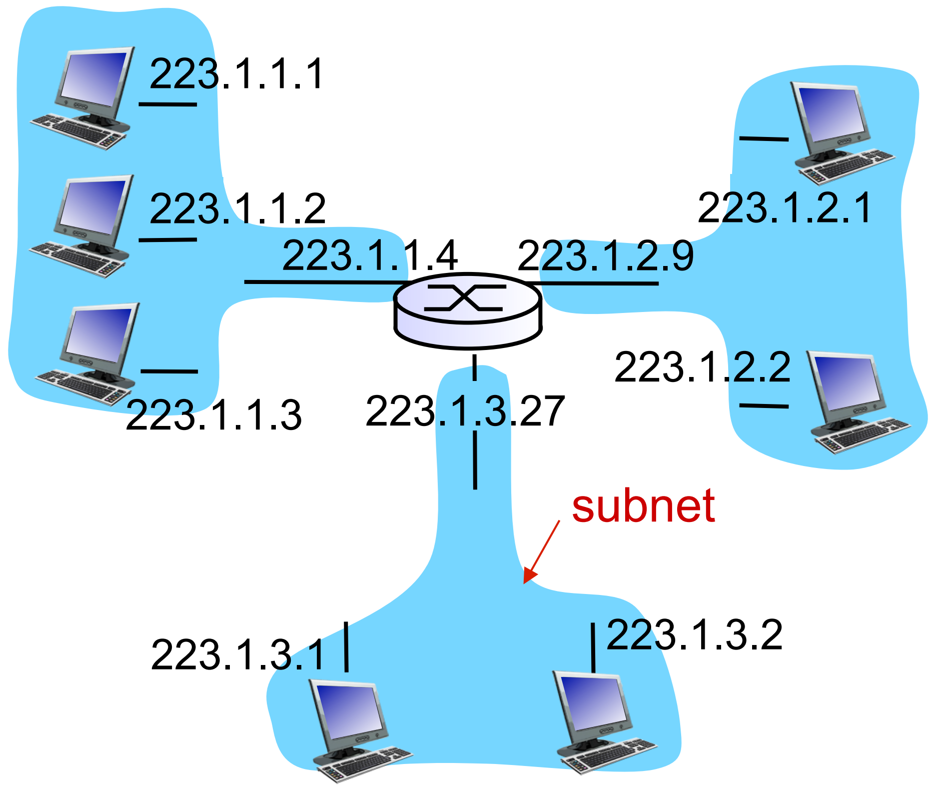
A: wired Ethernet interfaces connected by Ethernet switches (Link Layer)

For now: don't need to worry about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi base station

Subnets

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

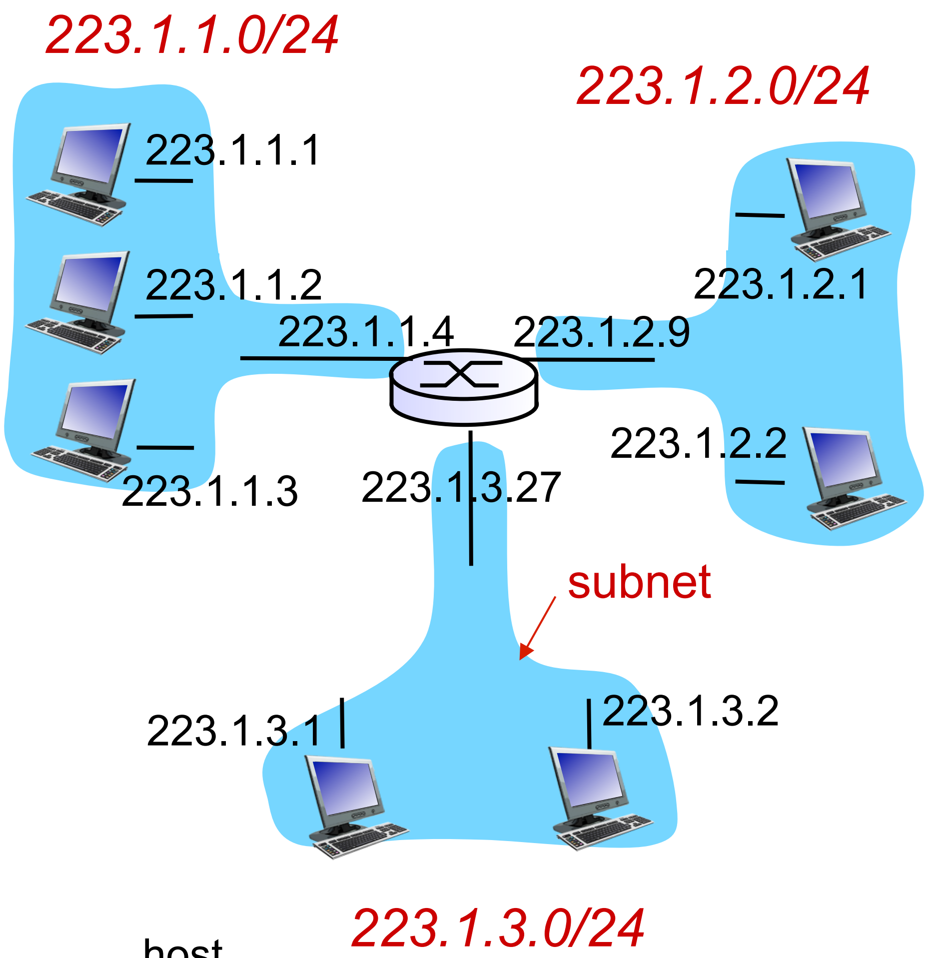


network consisting of 3 subnets

Subnets

recipe

- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



223.1.3.0/24

subnet part ← → host part

11011111 00000001 00000101 00000000

11011111 00000001 00000101 11111111

subnet mask: /24

The subnet mask

- A computer (or a router, which is simply a specialized computer) must be able to identify whether a computer with a given IP address is on its subnet or not.

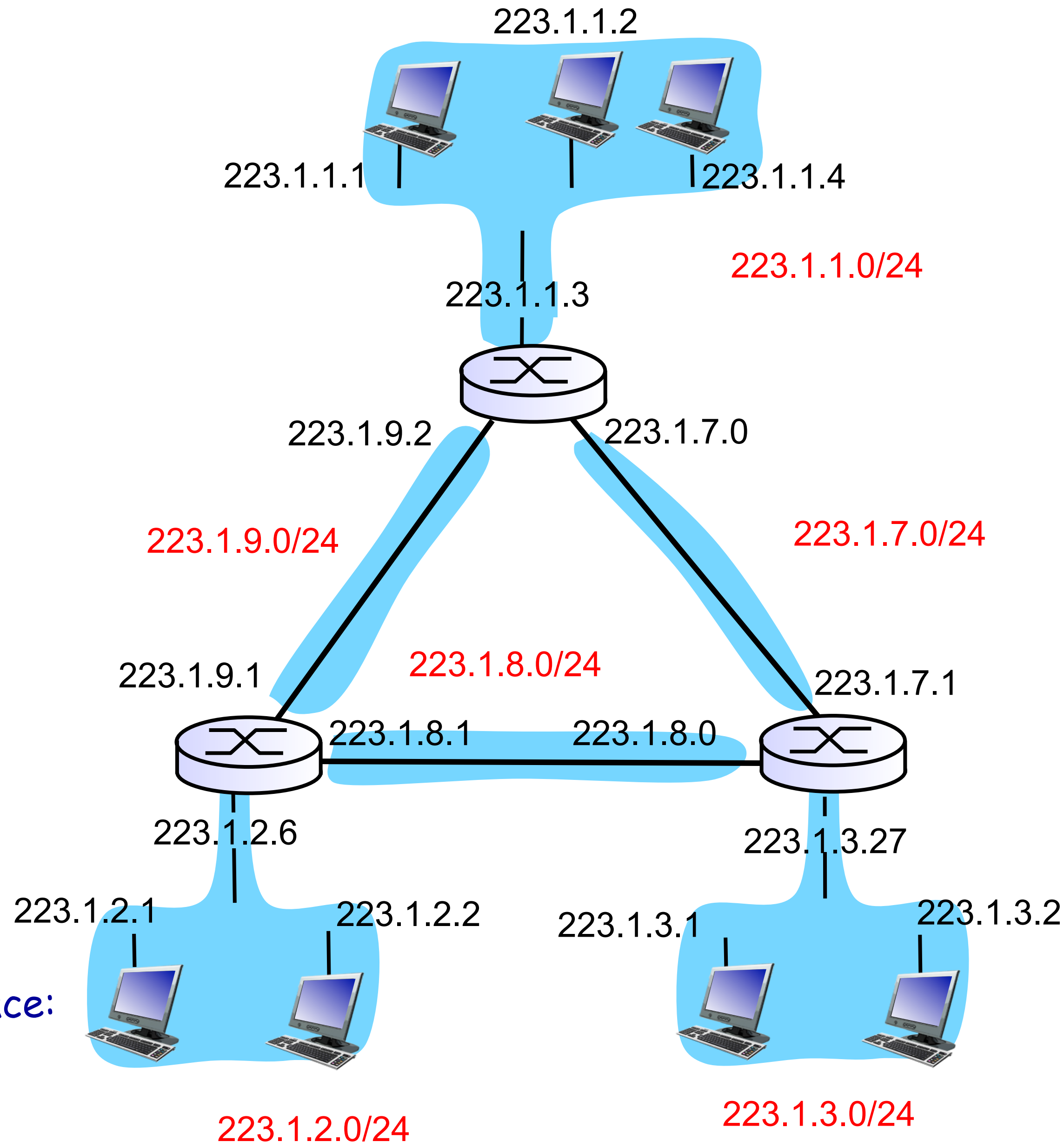
- The subnet mask is:
  - used to separate the network portion of an IP address from the host portion.
  - a set of 32 bits which the bits in the network portion of the address are set to 1s and the host portion is set to 0s.

example

- Subnet mask of 11111111.11111111.11111111.00000000 means the first 24 bits (3 bytes) are the network portion and the last 8 bits (1 byte) are host portion.
- This is called a /24 address.
- With a /24 subnet mask (255.255.255.0), IP address 192.168.1.100 would be separated into subnet portion 192.168.1 and host portion 100.
- Convert these subnet mask into decimal values
  - /27, /24, /16, /28, /22

Subnets

- How many subnets? 6
- Addressing scheme
  - Classful addressing
- Classful addressing
  - Subnet mask /8
    - *Class A*
  - Subnet mask /16
    - *Class B*
  - Subnet mask /24
    - *Class C*
- Classful addressing
  - An organization needs 2000 hosts and apply class B
  - B allocated 65534 interface: leaving more more than 63000 not used.
  - Not optimized and wasted address



Determine the number of Hosts in a subnet

Network		Host	
172	16	0	0

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 N

Network	Host	N
10101100 00010000	00000000 00000000	1
	00000000 00000001	2
	00000000 00000011	3
	⋮ ⋮	⋮
	11111111 11111101	65534
	11111111 11111110	65535
	11111111 11111111	65536

/16 SUBNET

$$65536 - 2 = 65534$$

$$2^N - 2 = 2^{16} - 2 = 65534$$

IP addressing: CIDR (Classless)

CIDR: Classless InterDomain Routing

- more flexible than original system of Internet Protocol (IP) address scheme i.e. classful addressing : A (subnet -8bit), B (subnet -16bit), C (subnet -24 bit))
- can avoid situations where large numbers of IP addresses are unused
- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

subnet part | host part

11001000 00010111 00010000 00000000

e.g. 200.23.16.0/23

IP addresses: how to get one?

Q: how does *network* get subnet part of IP addr?

A: gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
e.g. Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

IP addresses: how to get one?

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

- From local organization (e.g. UTM)
- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

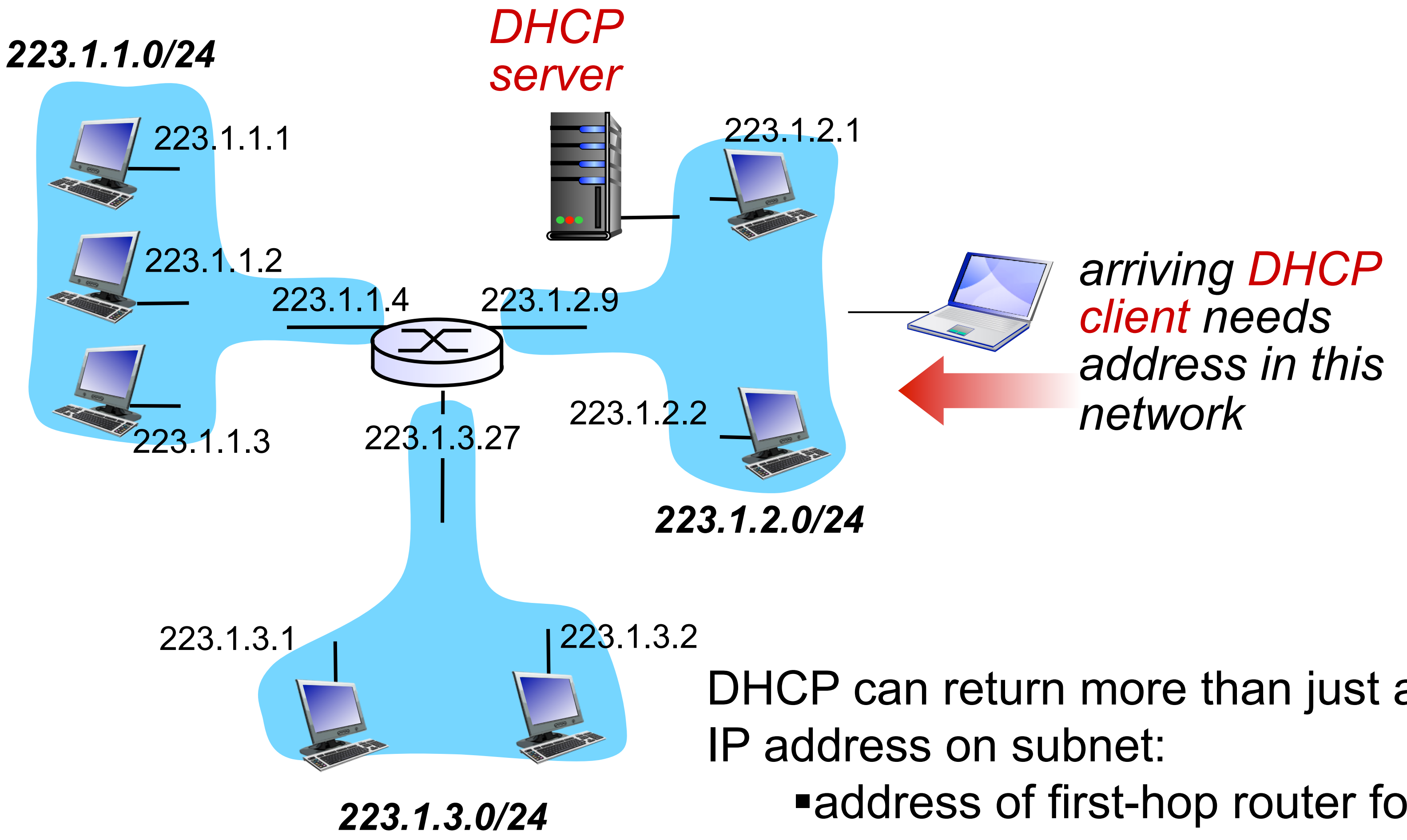
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

DHCP overview:

- host broadcasts "DHCP discover" msg [optional]
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

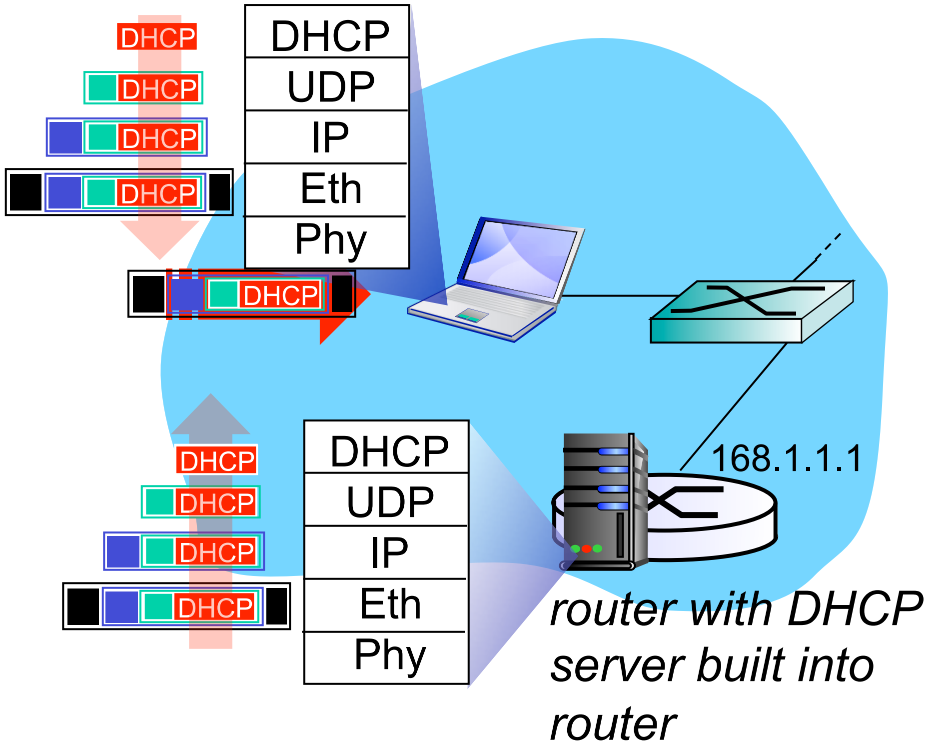
DHCP client-server scenario



DHCP can return more than just allocated IP address on subnet:

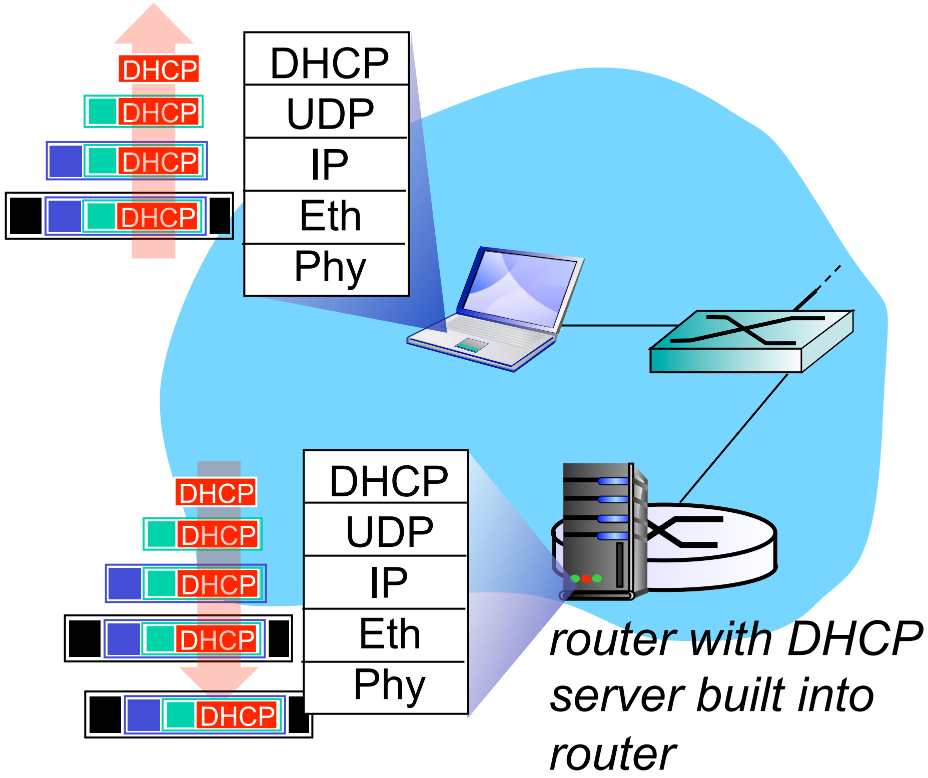
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

DHCP: Example Wireshark output (home LAN)

request

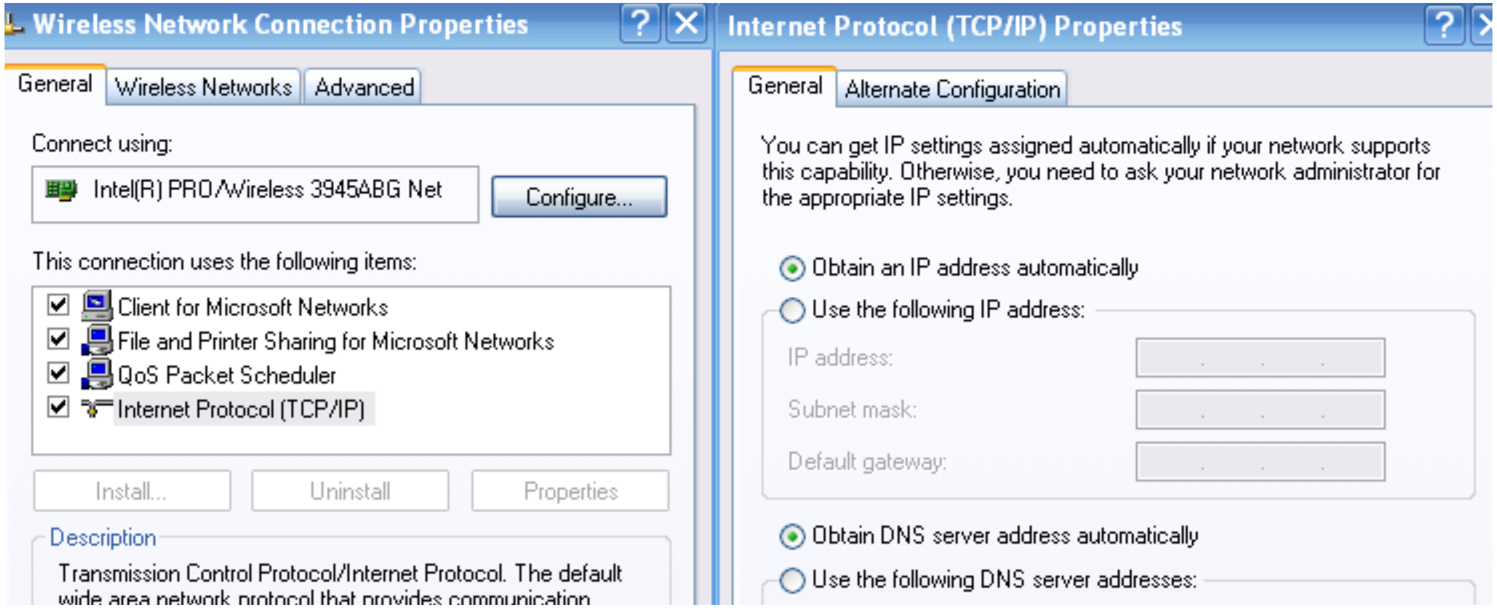
Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server
    ……

reply

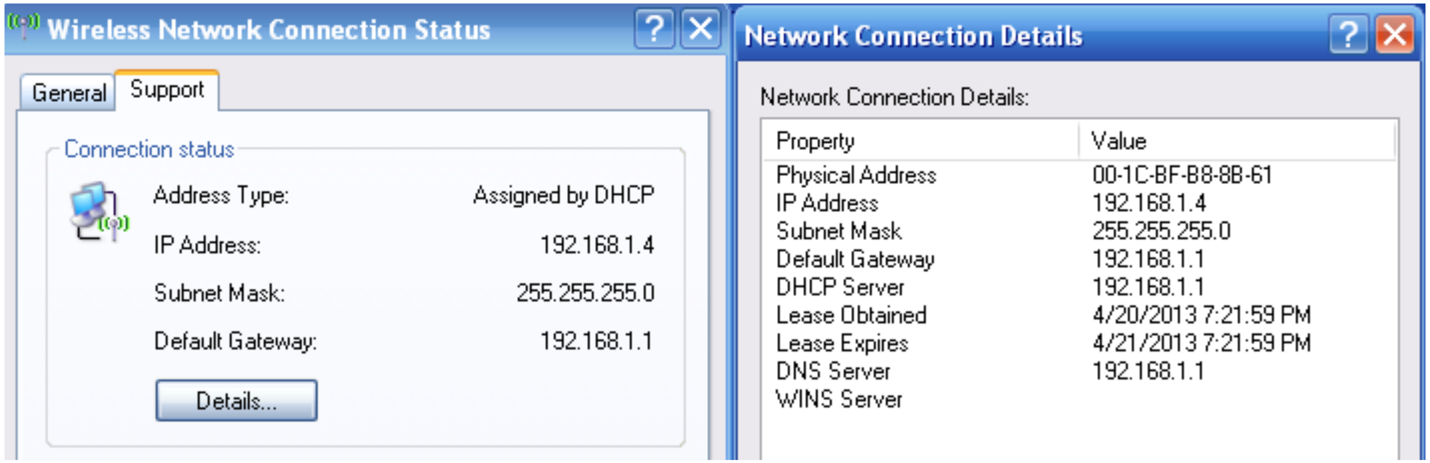
Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

DHCP: Example Assignment (home LAN)

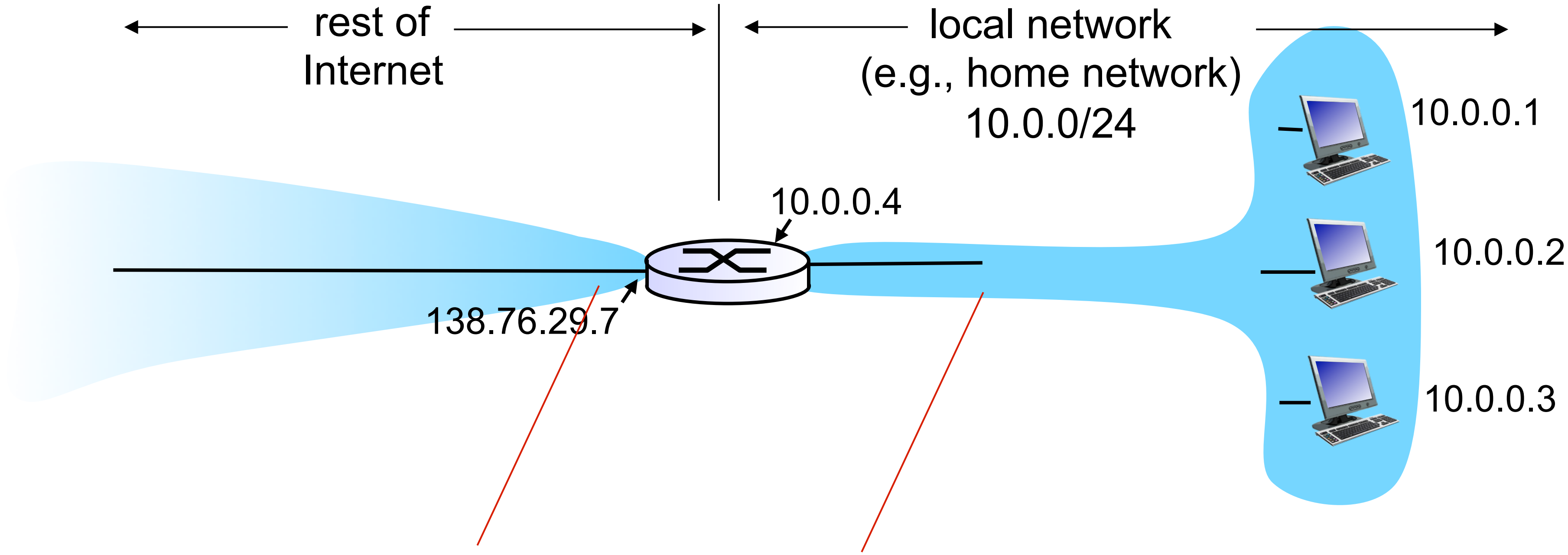
TCP/IP Client Setting



DHCP Server Assignment



NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

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

NAT: network address translation

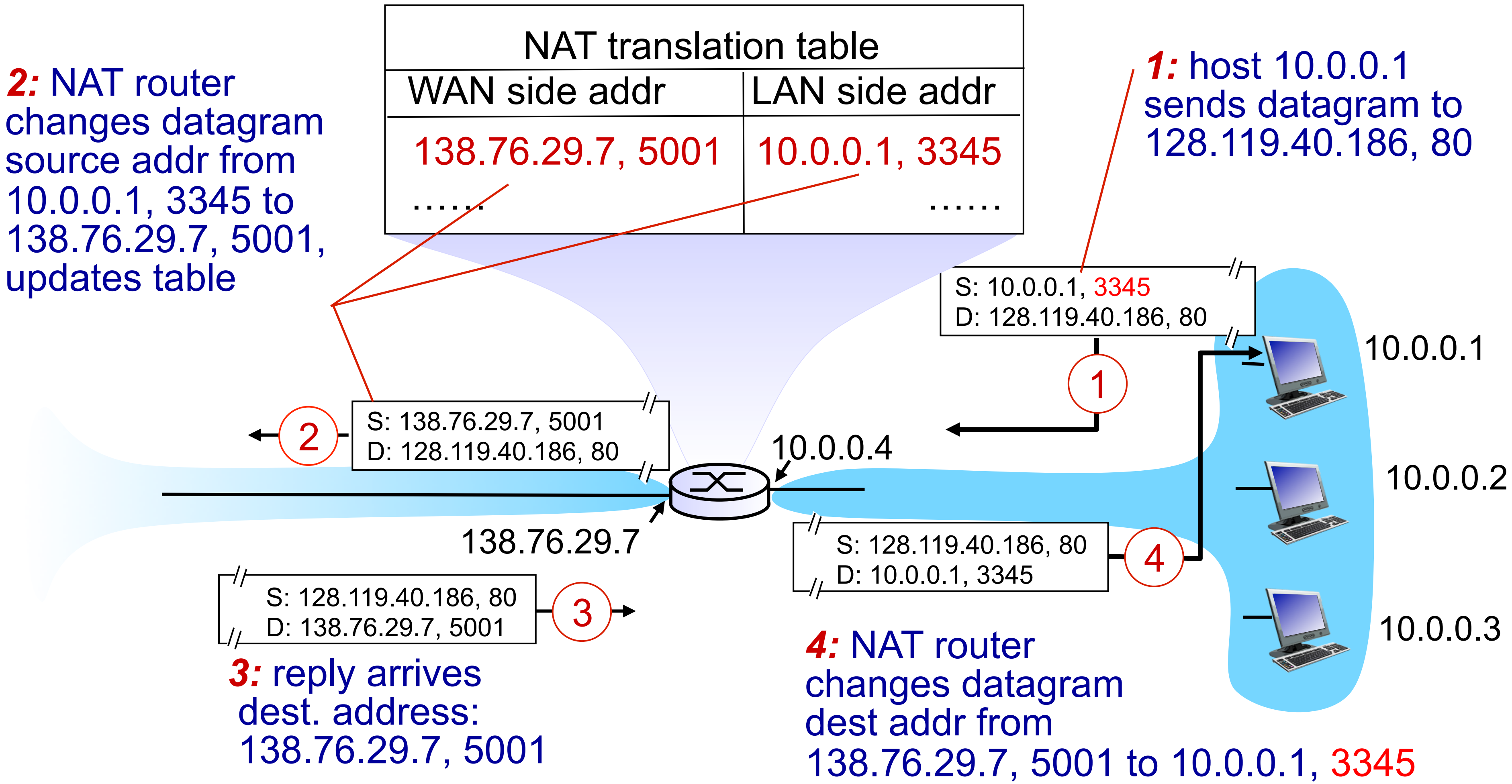
implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation



Chapter 4: outline

ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

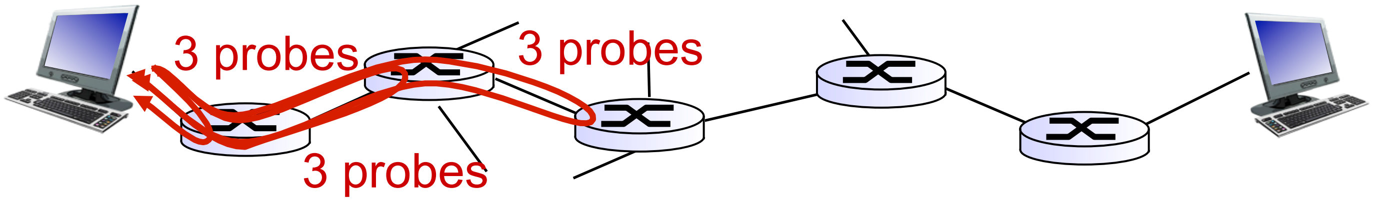
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when *n*th set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

stopping criteria:

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



IPv6: motivation

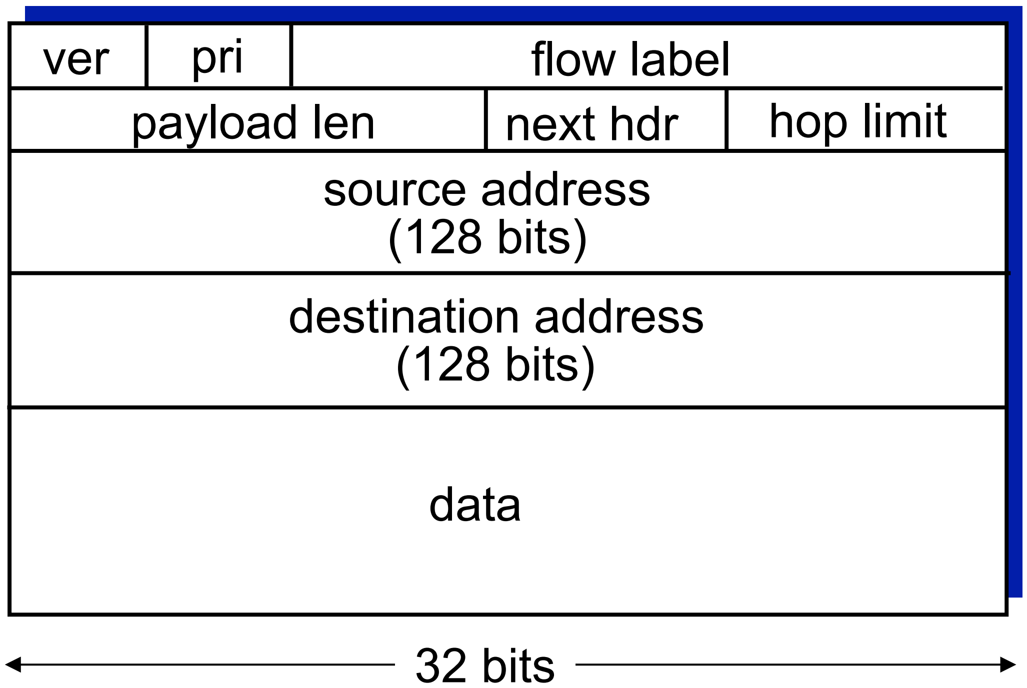
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

priority: identify priority among datagrams in flow
flow Label: identify datagrams in same "flow."
(concept of "flow" not well defined).
next header: identify upper layer protocol for data



Other changes from IPv4

- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

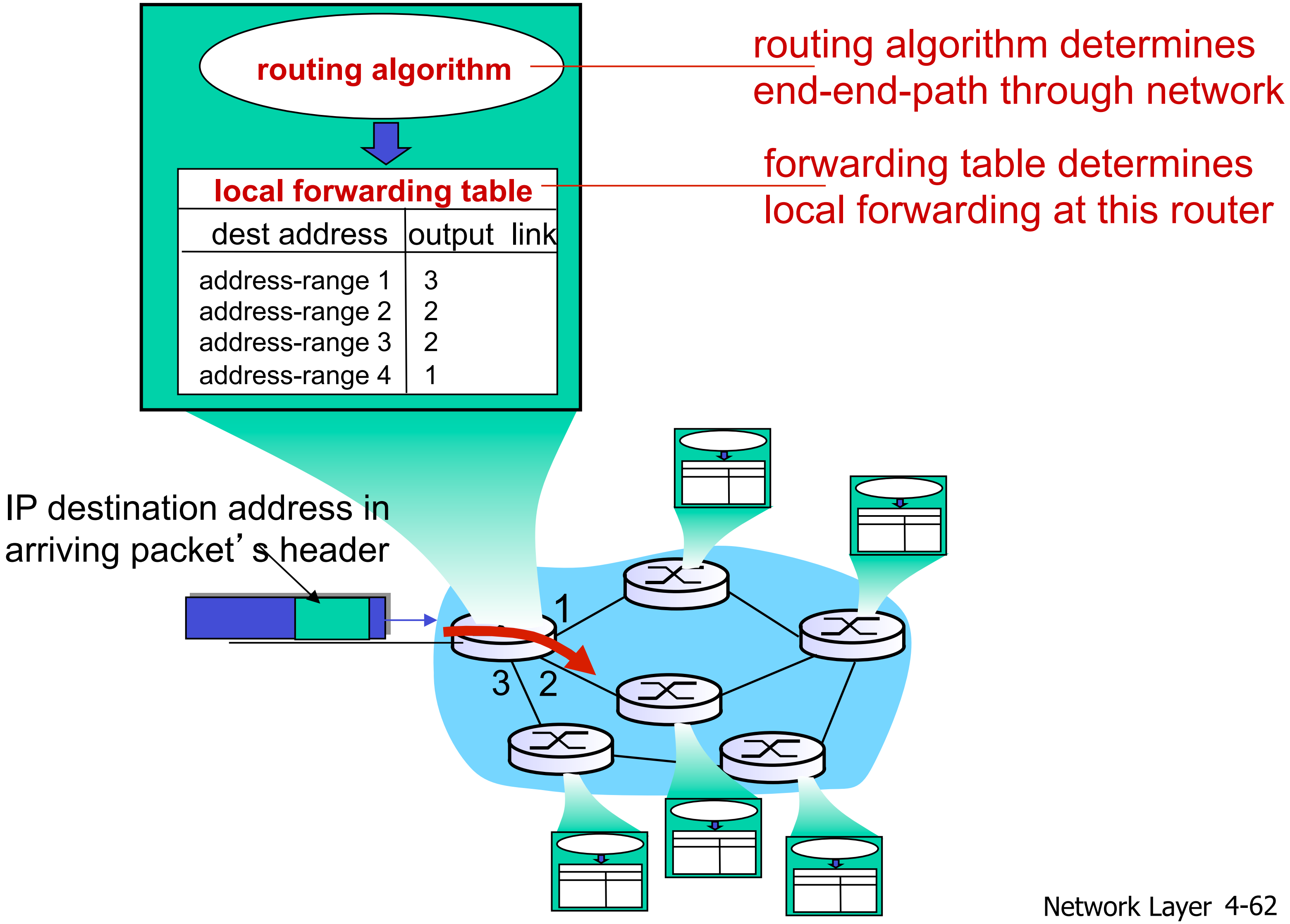
4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet (intra-AS routing)

4.7 broadcast and multicast routing

Interplay between routing, forwarding



Routing algorithm classification

Q: 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

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms
- link state (LS)
- distance vector (DV)
- hierarchical routing

4.6 routing in the Internet (intra-AS routing)

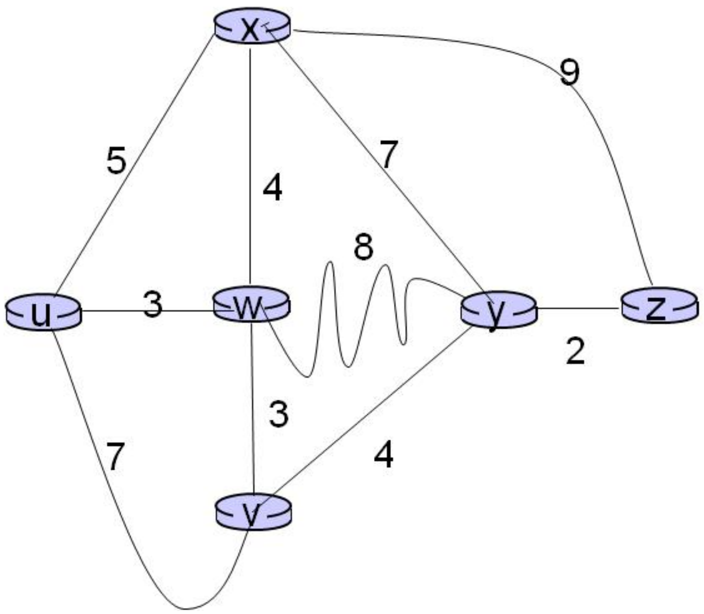
4.7 broadcast and multicast routing

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 *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

Example: Dijkstra's algorithm



Routers in nodes u,x,w,v,y and y have been assigned with link cost value as stated in the diagram:

i) Construct table least cost paths from node U to node Z
- construct shortest path tree by tracing predecessor nodes

ii) Computes least cost paths from node U to node Z

iii) Produce *forwarding table* for node U

notation:

- c(x,y): link cost from node x to y; = ∞ 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
- N': set of nodes whose least cost path definitively known

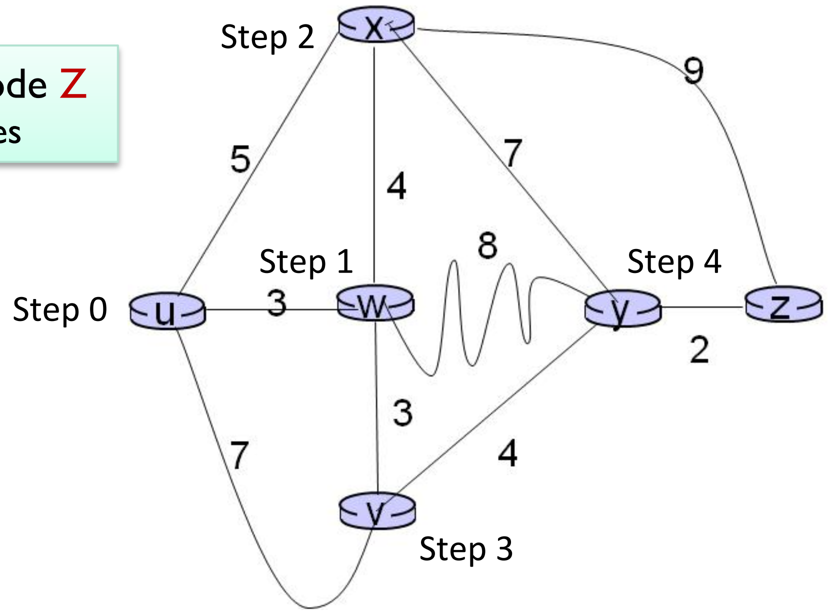
Dijkstra's algorithm:

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
       then D(v) = c(u,v)
    else D(v) = ∞

Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
 until all nodes in N'
```

Aim: Construct table least cost paths from node U to node Z

- construct shortest path tree by tracing predecessor nodes



How to determine D(v) in step 1 (N'=uw)?

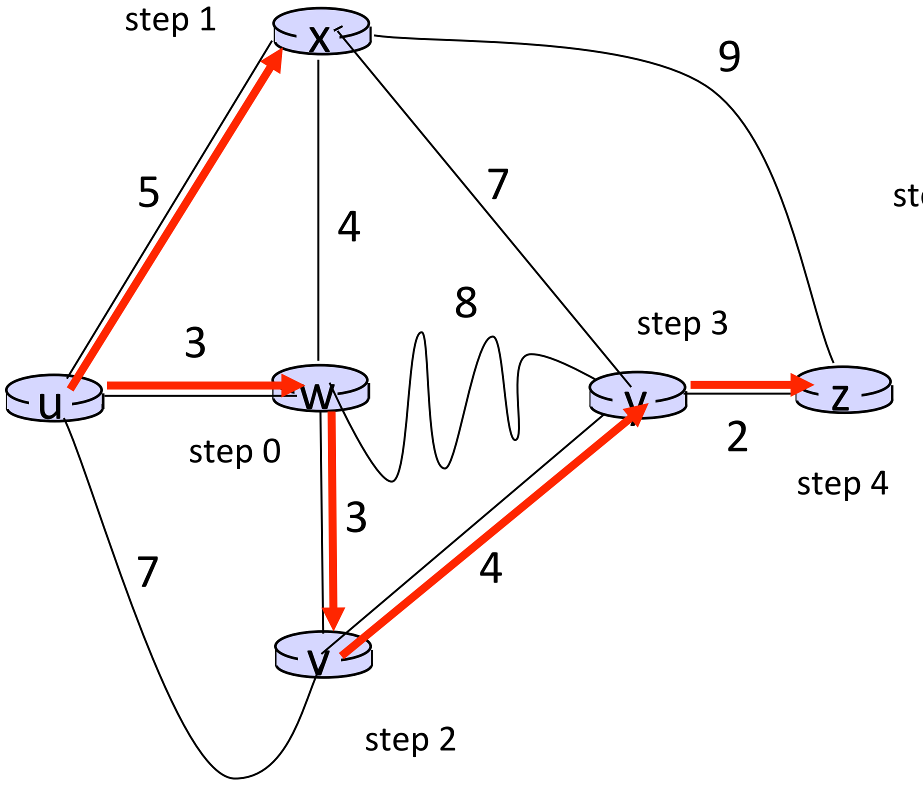
$$D(v) = \min(\text{link cost for possible route nodes } P(v))$$
$$= \min(D(v), D(w) + c(w,v))$$
$$= \min\{(7,u),(3+3,w)\} = 6w$$

Least cost paths table from node U to node Z

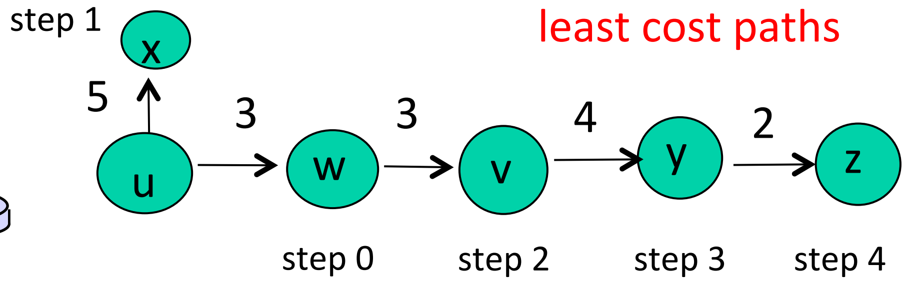
Step	N'	P(V) D(V)	P(W) D(W)	P(X) D(X)	P(Y) D(Y)	P(Z) D(Z)
0	u	(7,u) 7,u	(3,u) 3,u	(5,u) 5,u	∞	∞
1	uw	{(7,u),(6,w)} 6,w	-	{(5,u),(7,w)} 5,u	(11,w) 11,w	∞
2	uwx	{(7,u),(6,w)} 6,w	-	-	{(12,x),(11,w)} 11,w	(14,x) 14,x
3	uwxv	-	-	-	{(12,x),(11,w),(10,v} 10,v	(14,x) 14,x
4	uwxvy	-	-	-	-	{(12,y),(14,x)} 12,y

Aim: ii) Computes least cost paths from node U to node Z
iii) Produce *forwarding table* for node U

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



From Node U



forwarding table for u:

destination	Link (next u)	Least cost
v	(u,w)	6
x	(u,x)	5
y	(u,w)	10
w	(u,w)	3
z	(u,w)	12

Chapter 4: outline

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

Bellman-Ford equation (dynamic programming)

let $d_x(y)$:= cost of least-cost path from x to y

then $d_x(y) = min_v\{c(x,v) + d_v(y)\}$ for each node $y \in N$

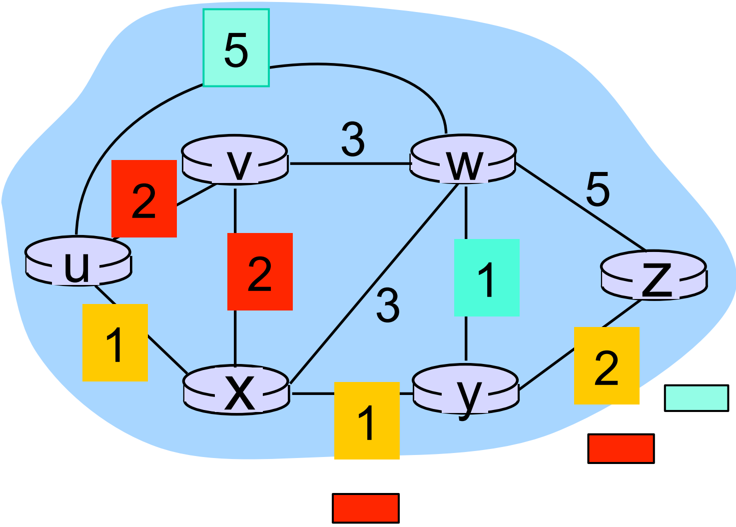
min taken over all neighbors v of x

cost to neighbor v

cost from neighbor v to dest y

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Bellman-Ford example



clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\qquad c(u,x) + d_x(z), \\ &\qquad c(u,w) + d_w(z) \} \\ &= \min \{2 + 5, \\ &\qquad 1 + 3, \\ &\qquad 5 + 3\} = 4 \end{aligned}$$

node achieving minimum is next
hop in shortest path, used in forwarding table

Distance vector algorithm

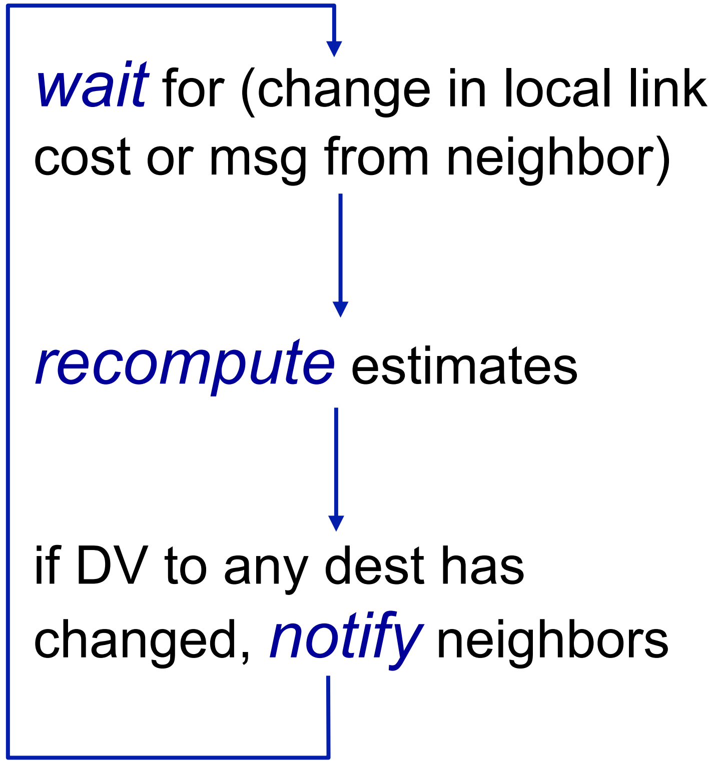
iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



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

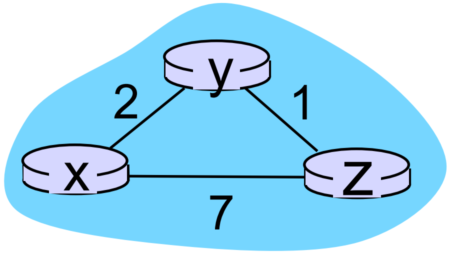
$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node

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

Symmetric Diagonal

Final Routing Table updated in node x, y, z



End target:
$D_x(y)$ converge to the actual least cost $d_x(y)$

Steps Involved:

- *Iterative & Asynchronous estimation*
- *Distributed Updated Information*

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet (Intra-AS routing)

4.7 broadcast and multicast routing

Hierarchical routing

our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 600 million destinations:

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

administrative autonomy

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

Hierarchical routing

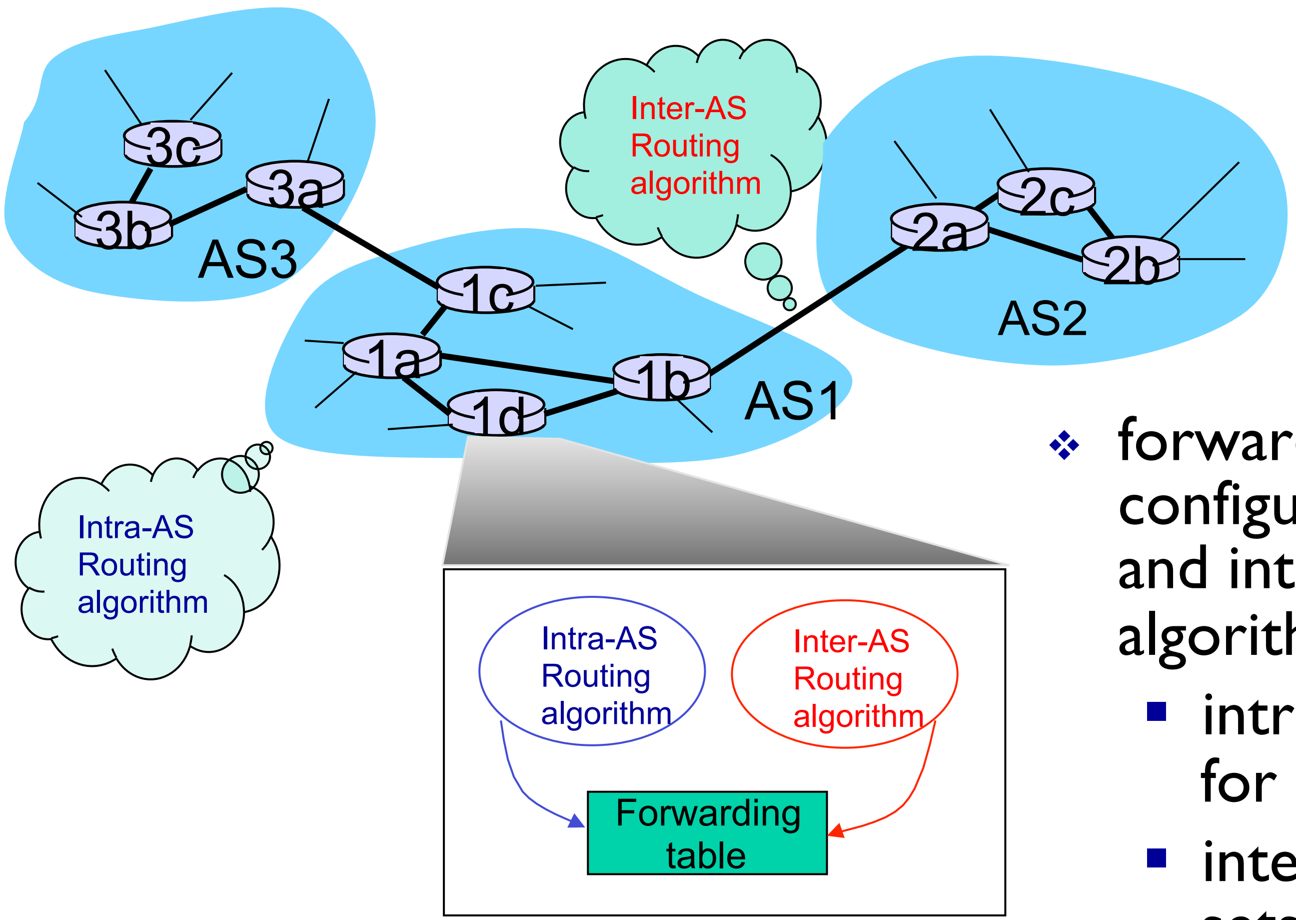
- aggregate routers into regions, “autonomous systems (AS)”
- routers in same AS run same routing protocol
  - “intra-AS” routing protocol
  - routers in different AS can run different intra-AS routing protocol

gateway router:

- at “edge” of its own AS
- has link to router in another AS

How to Update Forwarding Table Content in “autonomous system (AS)“ router

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP (Internet Control Message Protocol)
- IPv6

4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet (Intra-AS routing)

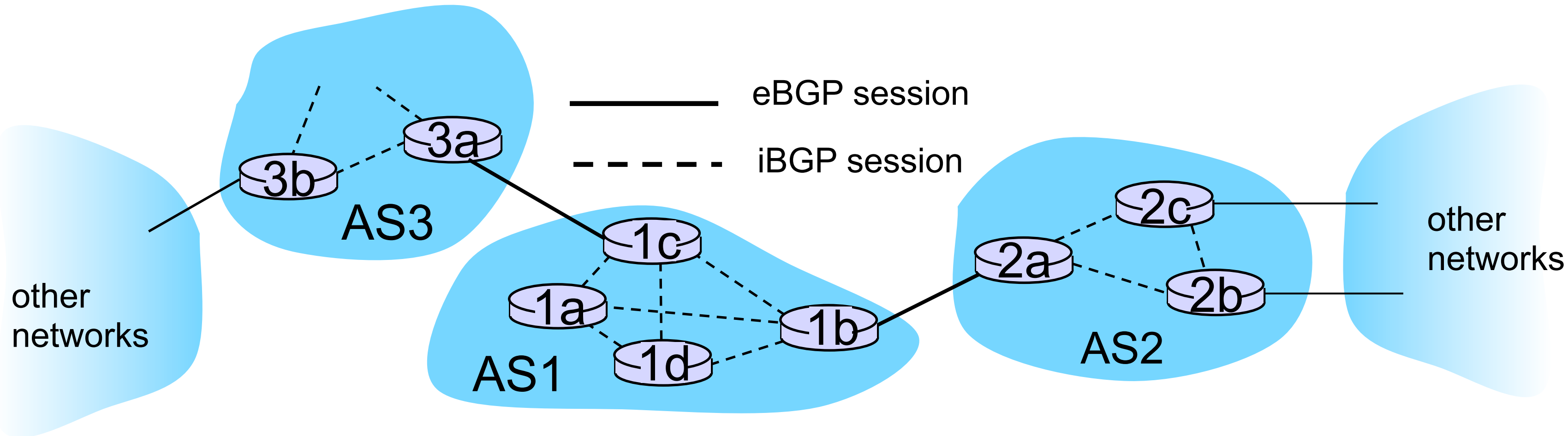
4.7 broadcast and multicast routing

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol (distance vector)
  - OSPF: Open Shortest Path First (link state)
  - IGRP: Interior Gateway Routing Protocol (distance vector with link state property)

Inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol. It is based on "path vector routing " (different from DV and LS)
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs.
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and policy.



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what’s inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

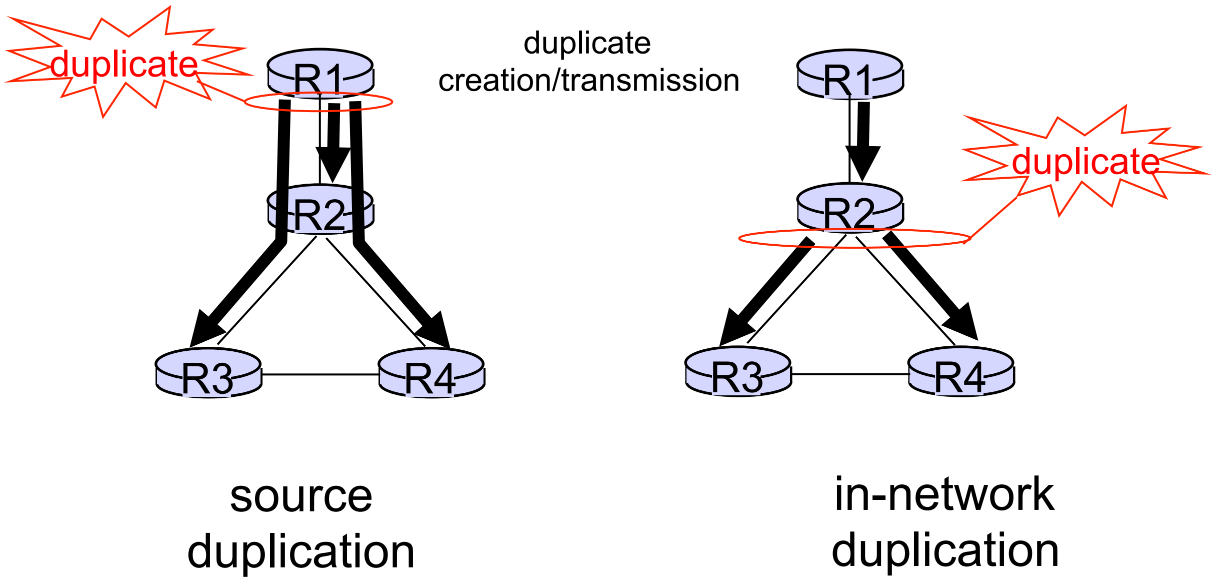
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet (Intra-AS routing)

4.7 broadcast and multicast routing

Broadcast routing

- deliver packets from source to all other nodes
- source duplication is inefficient:



- source duplication: how does source determine recipient addresses?

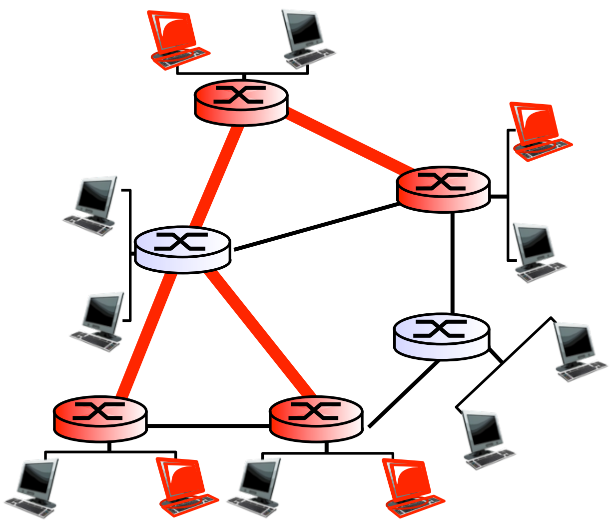
How does source determine recipient addresses? (Through in-network duplication)

- *flooding:* when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- *controlled flooding:* node only broadcasts pkt if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadcasted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- *spanning tree:*
  - no redundant packets received by any node

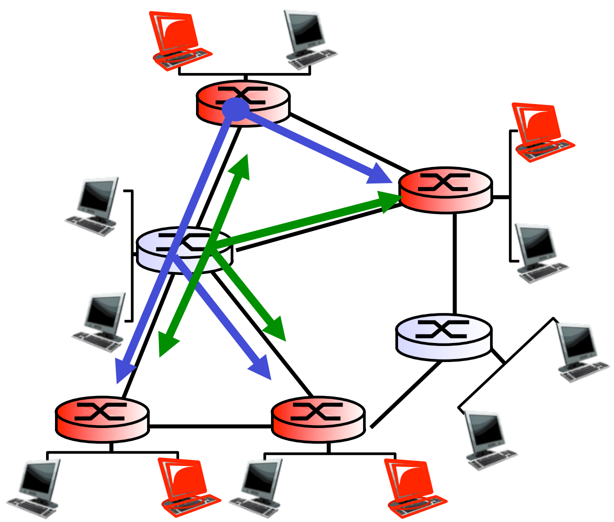
Multicast routing: problem statement

goal: find a tree (or trees) connecting routers having local mcast group members

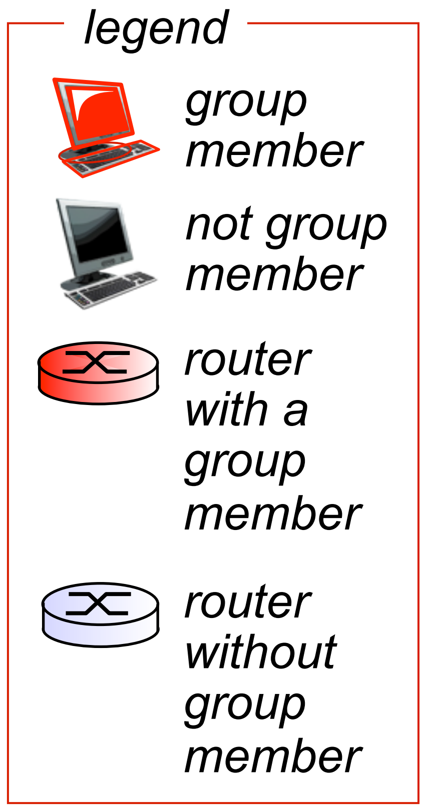
- *tree:* not all paths between routers used
- *shared-tree:* same tree used by all group members
- *source-based:* different tree from each sender to rcvrs



shared tree



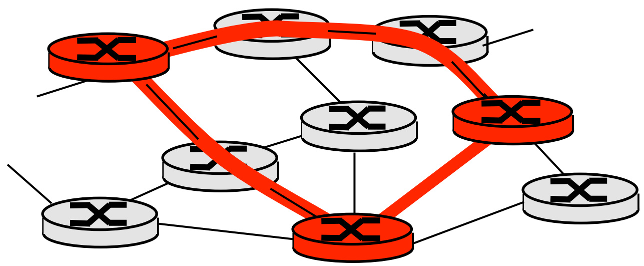
source-based trees



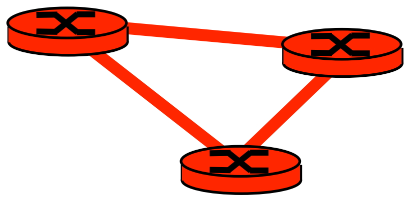
Tunneling

goal: find a tree (or trees) connecting routers having local mcast group members

Q: how to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology



logical topology

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router (recall IPv6 inside IPv4 tunneling)
- receiving mcast router unencapsulates to get mcast datagram

Chapter 4: *done!*

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format, IPv4 addressing, ICMP, IPv6

4.5 routing algorithms
- link state, distance vector, hierarchical routing

4.6 routing in the Internet
- RIP, OSPF, BGP

4.7 broadcast and multicast routing

- understand principles behind network layer services:
  - network layer service models, forwarding versus routing how a router works, routing (path selection), broadcast, multicast
- instantiation, implementation in the Internet