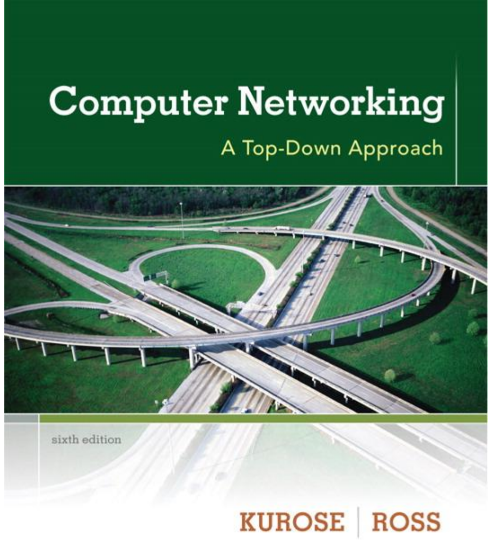
Chapter 4
Network Layer



A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

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

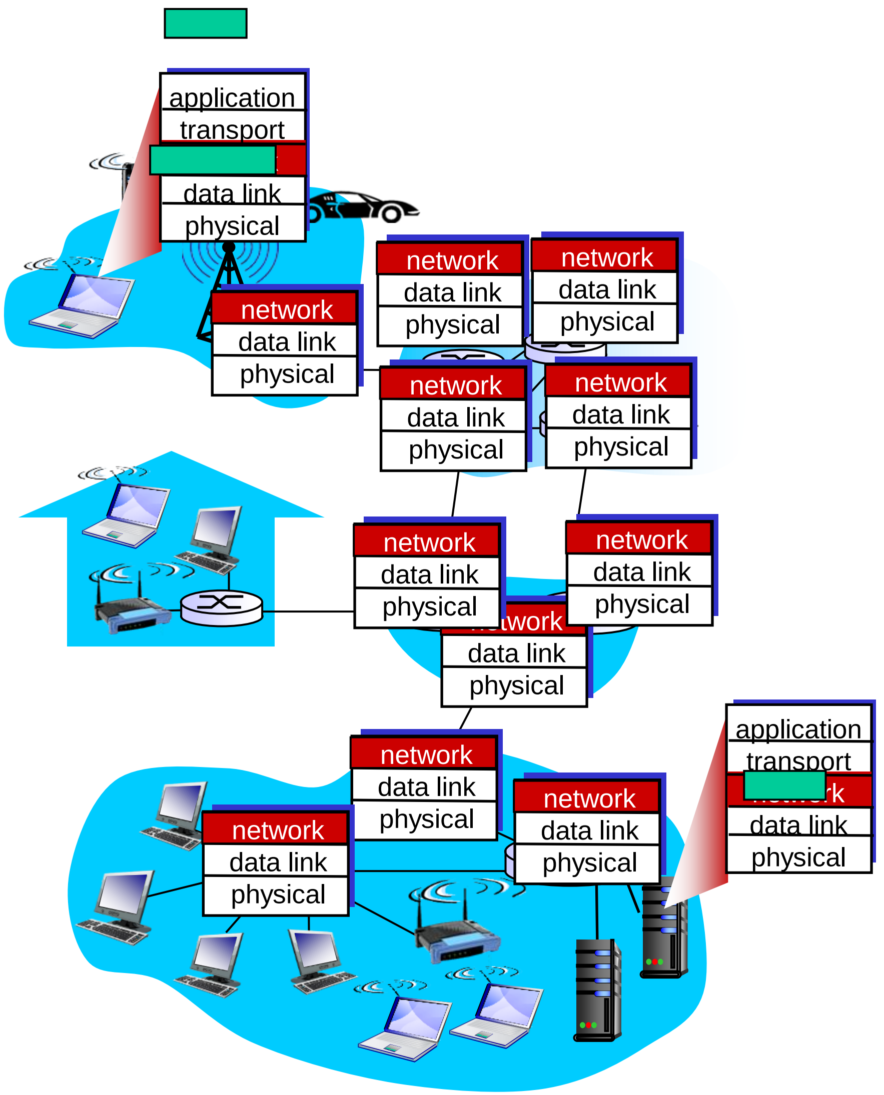
4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet
- RIP
- OSPF
- BGP

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



Two key network-layer functions

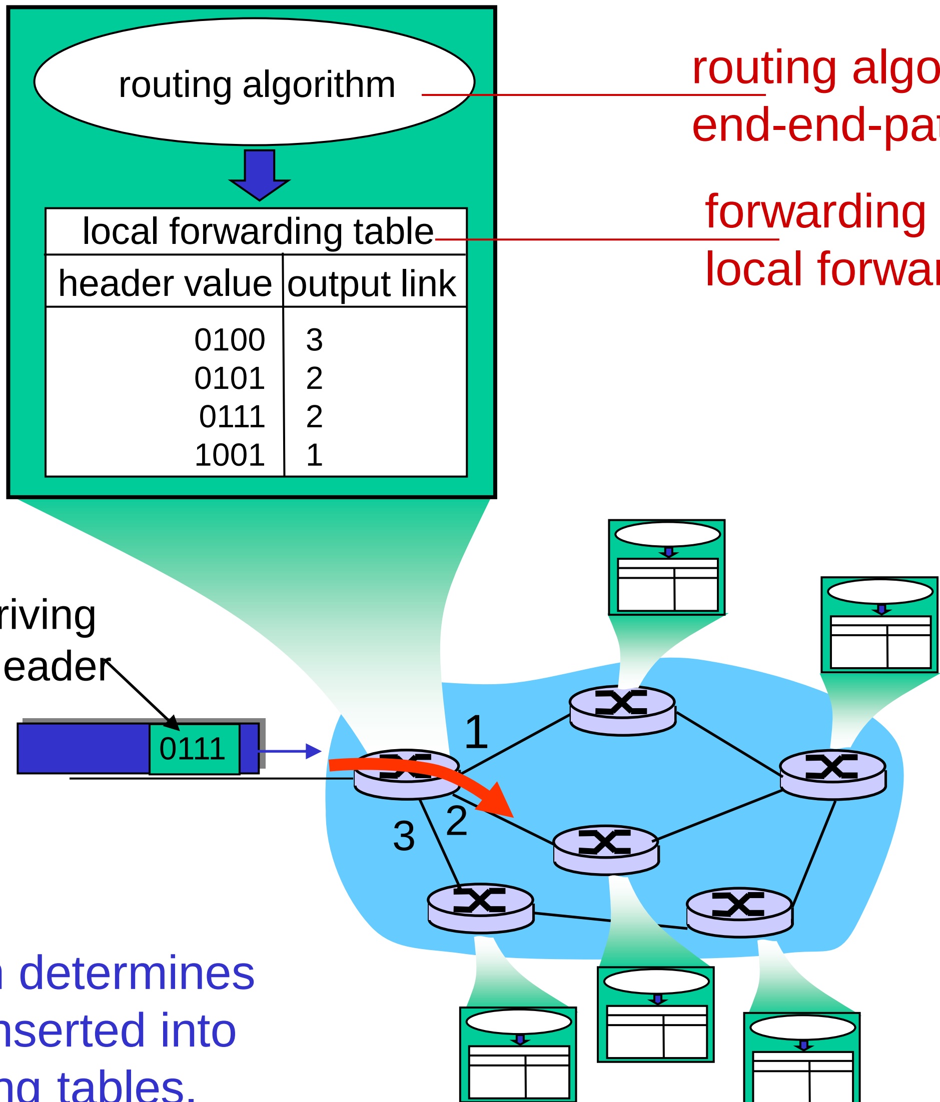
- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.
  - *routing algorithms*

analogy:

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

Interplay between routing and forwarding

value in arriving
packet's header



routing algorithm determines
end-end-path through network

forwarding table determines
local forwarding at this router

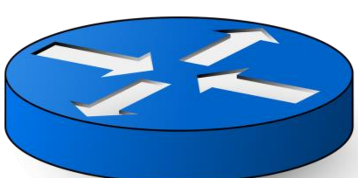
the routing algorithm determines
the values that are inserted into
the routers' forwarding tables.

Connection setup

- 3rd important function in *some* network architectures (besides routing and forwarding):
  - ATM, frame relay, X.25
- before datagrams flow, two end hosts *and* intervening routers establish virtual connection
  - 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

Some terms

- Packet switch: a general packet-switching device
- Link-layer switch: make forwarding decisions on values in the fields of the link-layer frame
- Router: make forwarding decisions on values in the fields of the network-layer datagram
  - Must implement layer 2 protocols as well



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 bounded delay (for example, less than 40 msec delay)

example services for a flow of datagrams:

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

ATM network service model

- CBR (constant bit rate)
  - End-to-end delay, jitter, and the fraction of cells that are lost or delivered late are all guaranteed to be less than specified values.
  - Values are agreed by the sending host and the ATM network when the CBR connection is first established.
- ABR (available bit rate)
  - Cells can be lost
  - Cells can not be reordered
  - A minimum cell transmission rate (MCR) is guaranteed

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

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

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Connection, connection-less service

- *Datagram (数据报)* network provides network-layer *connectionless* service
- *virtual-circuit (虚电路)* network provides network-layer *connection* service
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service:* host-to-host (not process to process)
  - *no choice:* network provides one or the other, but not both
  - *implementation:* in network core, fundamentally different

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 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)

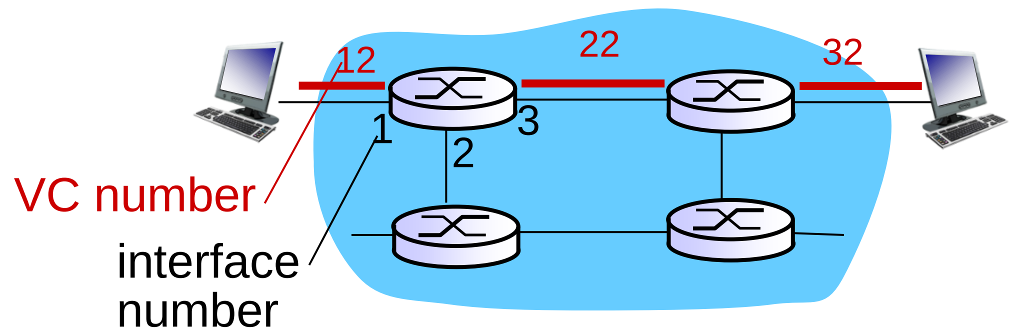
VC implementation

a VC consists of:

1. *path* from source to destination
2. *VC numbers*, one number for each link along path
3. *entries in forwarding tables* in routers along path

- packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - each intervening router must replace the VC number of each traversing packet with a new VC number
  - new VC number comes from forwarding table

VC forwarding table



forwarding table in northwest router:

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

VC routers maintain connection state information!

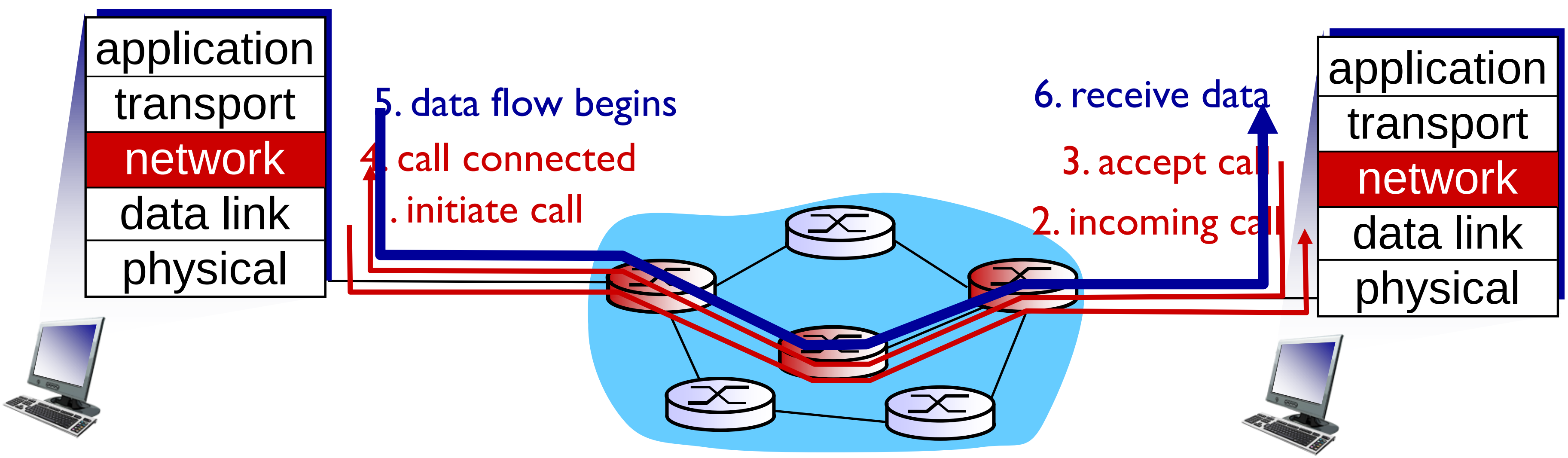
VC phases

- VC setup: determine the source to destination path; routers also determine the VC number for each link; add an entry in the forwarding table in each router along the path.
- Data transfer: packets flow along
- VC teardown: update the forwarding tables

- Difference:
  - Transport layer: involves only two end systems
  - Network layer: routers along the path between the two end systems are involved

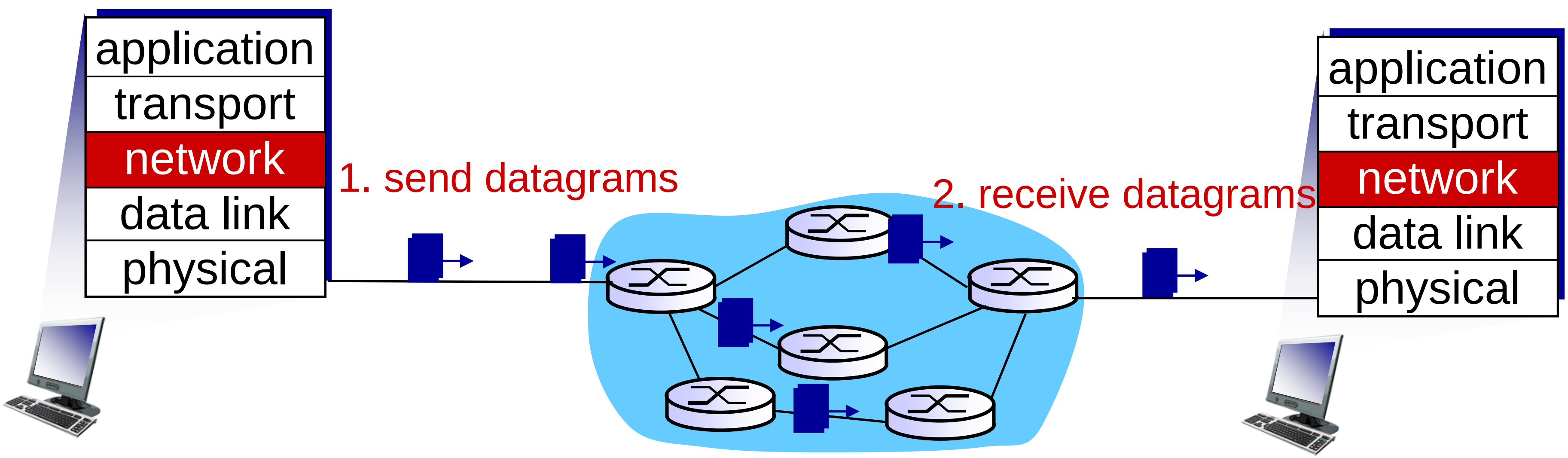
Virtual circuits: signaling protocols

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

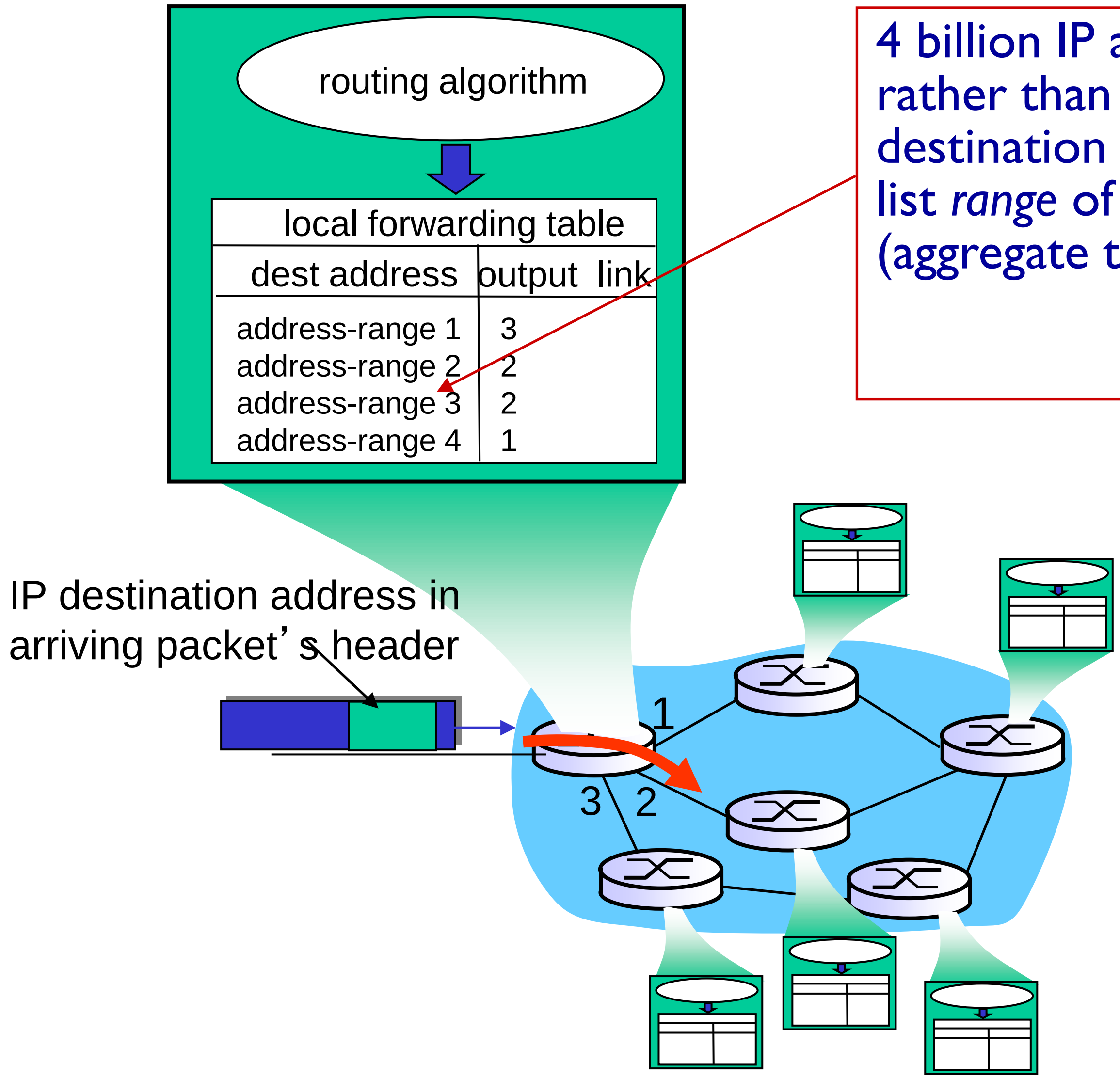


Datagram networks

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



Datagram forwarding table



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

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

200.23.16.0 - 200.23.23.255

200.23.24.0 - 200.23.24.255

200.23.25.0 - 200.23.31.255

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix matching（最长前缀匹配）

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

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

examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

Datagram routing

- Router maintain forwarding state information in the forwarding table
- Modified by routing algorithms
  - Updated every one to five minutes
- A series of packets from a same source to a same destination may follow different paths.

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

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

Router architecture overview

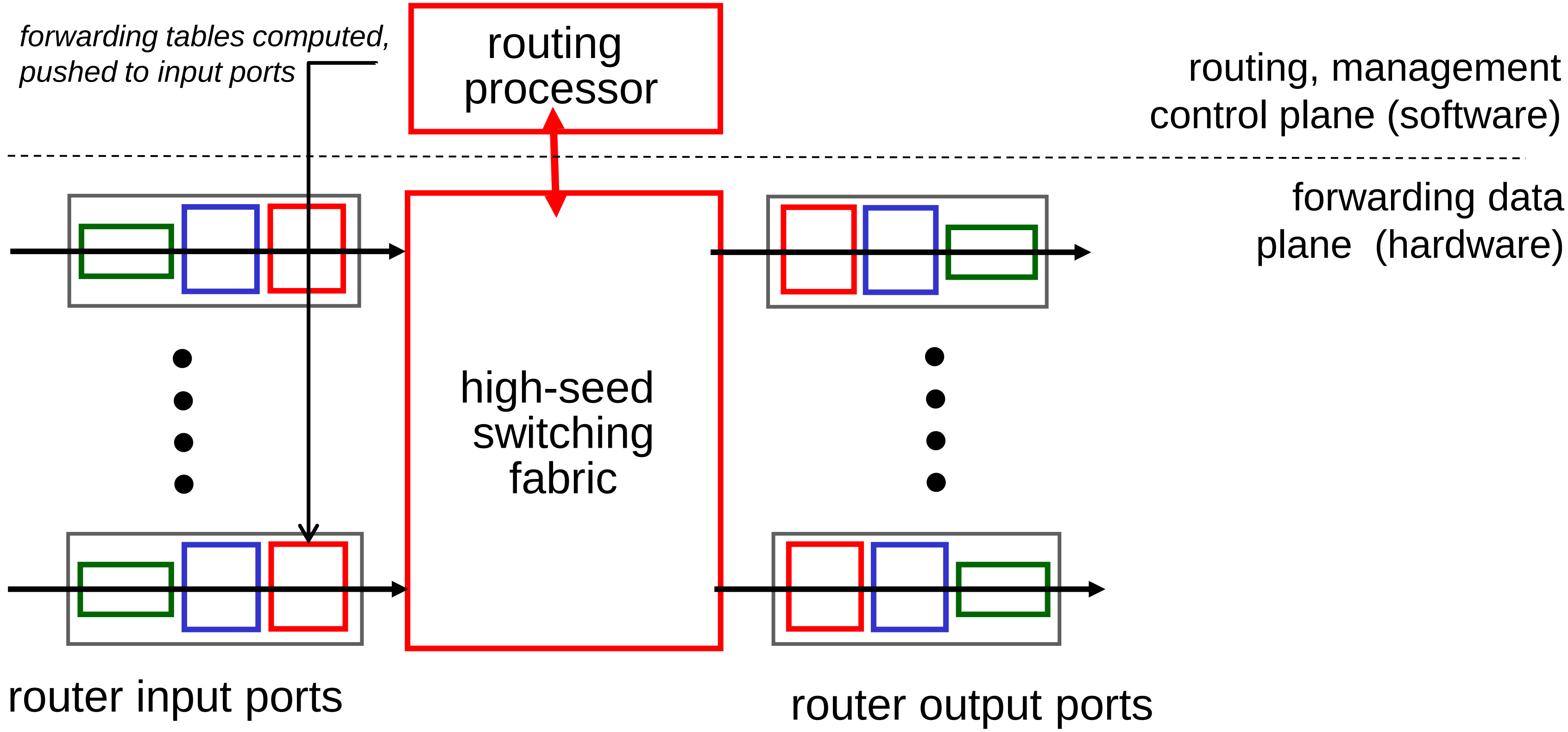
two key router functions:

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

Four components:

- Input ports: physical-layer function; link-layer function; look up the forwarding table; forward the control packet to the routing processor
- Switching fabric: connects input ports to output ports
- Output ports: buffer; link-layer and physical-layer functions
- Routing processor: executes routing protocols

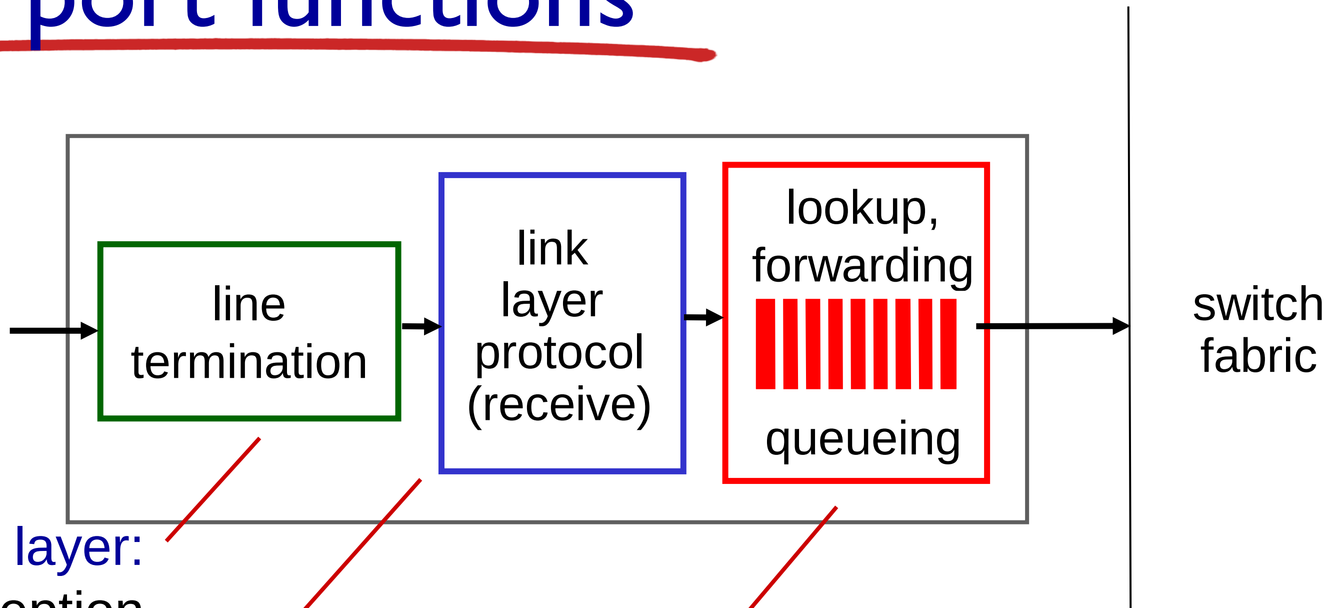
Router architecture overview



Router architecture overview

- A router's input ports, output ports, and switching fabric collectively referred to as the router forwarding plane
  - Almost always implemented in hardware
  - 10Gbps port ~ 51.2 ns to process an IP datagram
- Router' control functions—executing the routing protocols, responding to attached links that go up or down, and performing management functions, are referred to as the router control plane
  - Software running on CPU
  - Much slower

Input port functions



physical layer:
bit-level reception

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

decentralized switching:

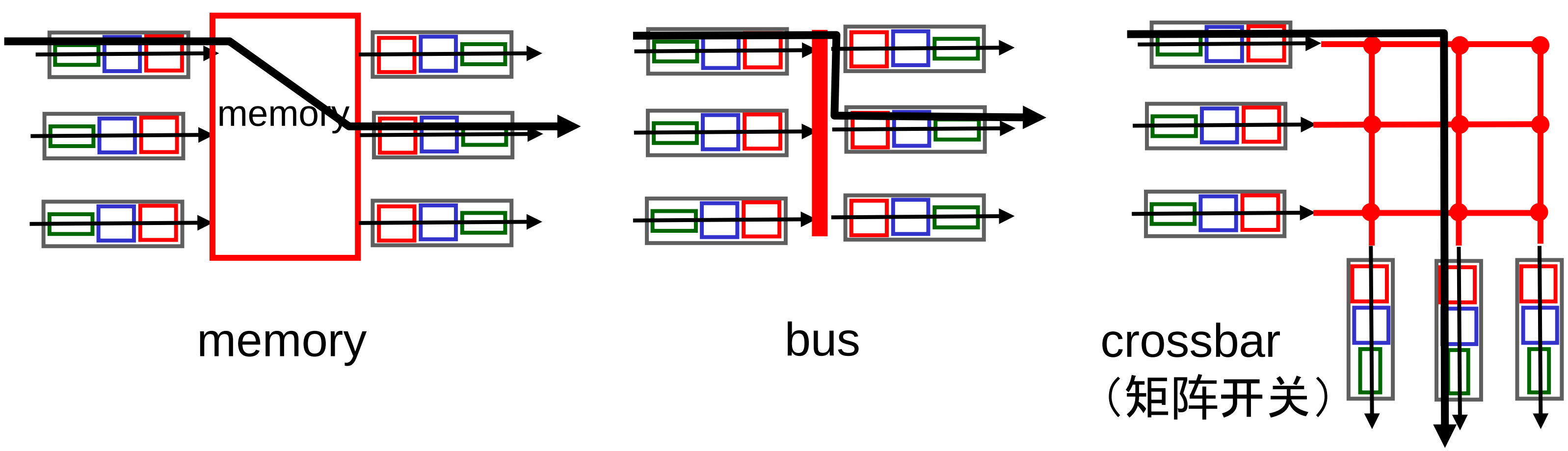
- given datagram dest., lookup output port using forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions

- Lookup the forwarding table
- The forwarding table is computed and updated by the routing processor,
  - a shadow copy typically stored at each input port.
- Longest prefix match
- After determining the output port, enters into the forwarding fabric
  - A packet may be temporally blocked
  - Queued in the input port
- Others:
  - Link- and physical- layer functions
  - Counters, ttl, etc.

Switching fabrics

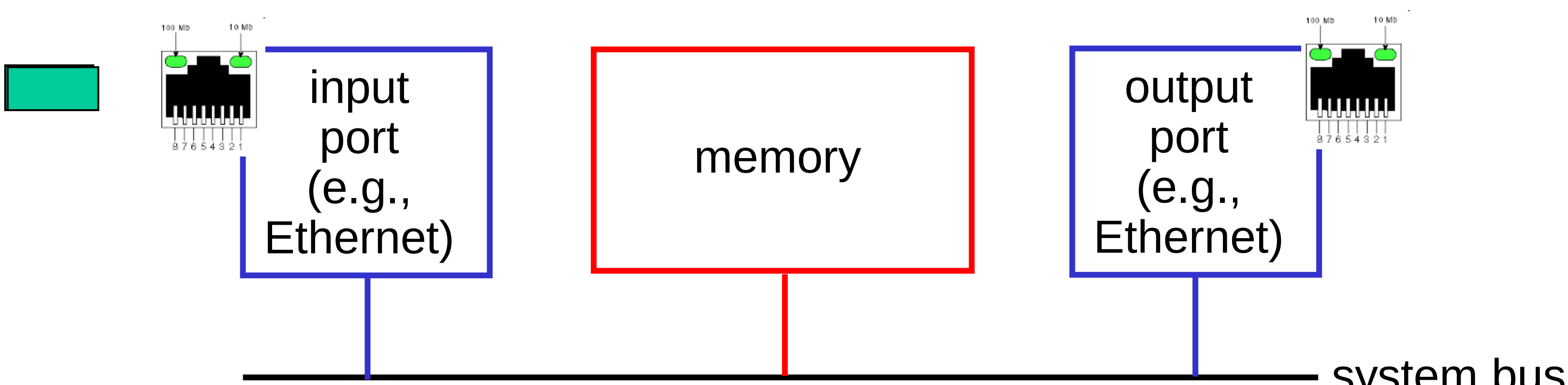
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transferred from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



Switching via memory

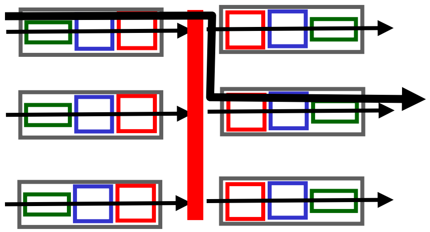
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- CPU extracts dest address from packet's header, looks up output port in forwarding table, copies to output port
- speed limited by memory bandwidth (2 bus crossings per datagram)
- one packet at a time



Switching via a bus

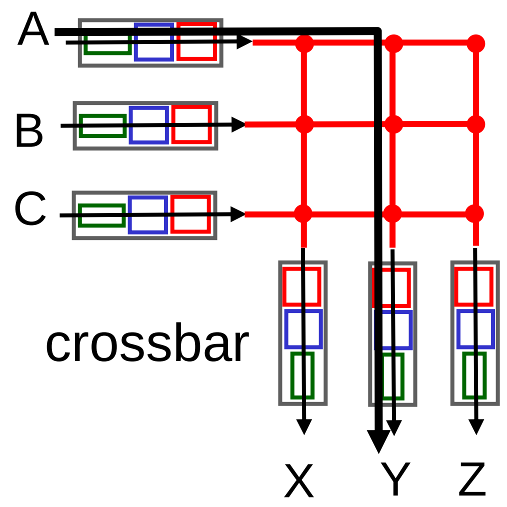
- datagram from input port memory to output port memory via a shared bus
  - Pre-pend switch internal label
  - packet received by all output ports, but only the port that matches the label will keep the packet
- *bus contention:* switching speed limited by bus bandwidth
- Limited by bus speed, one packet a time
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



bus

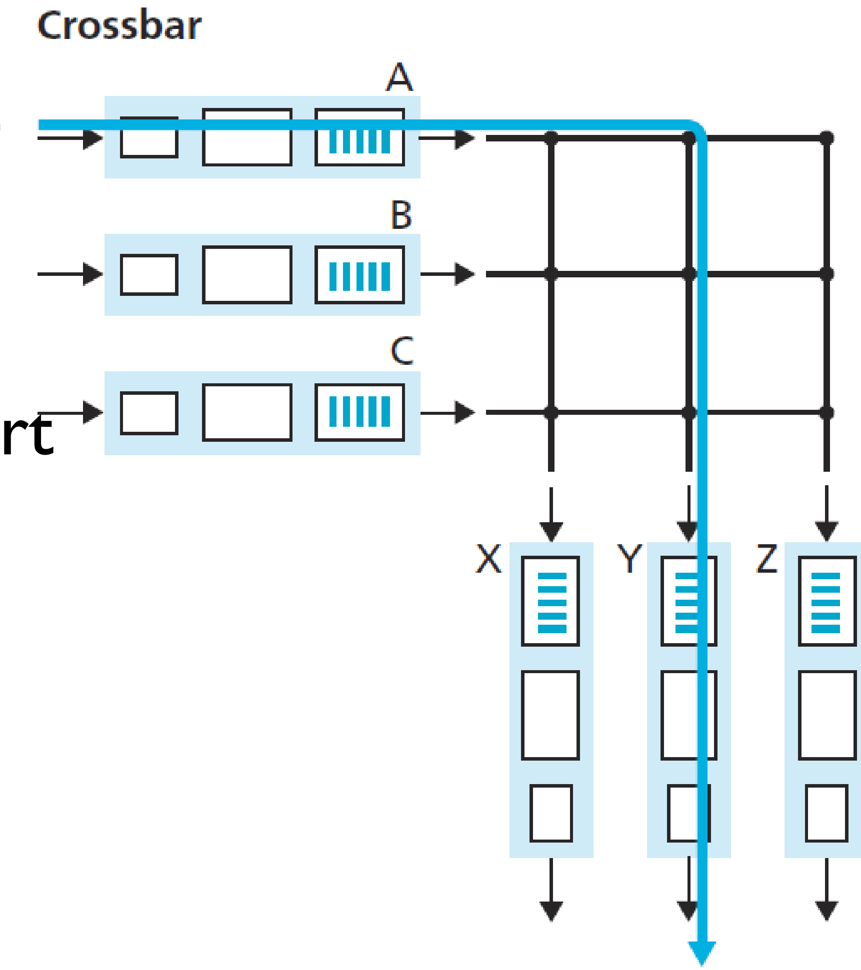
Switching via interconnection network

- use a more sophisticated interconnection network, such as those that have been used in the past to interconnect processors in a multiprocessor architecture.
  - N input ports and N output ports, 2N buses.
  - Each vertical bus intersects with each horizontal bus, at a cross point
  - Can be open and close by the fabric controller

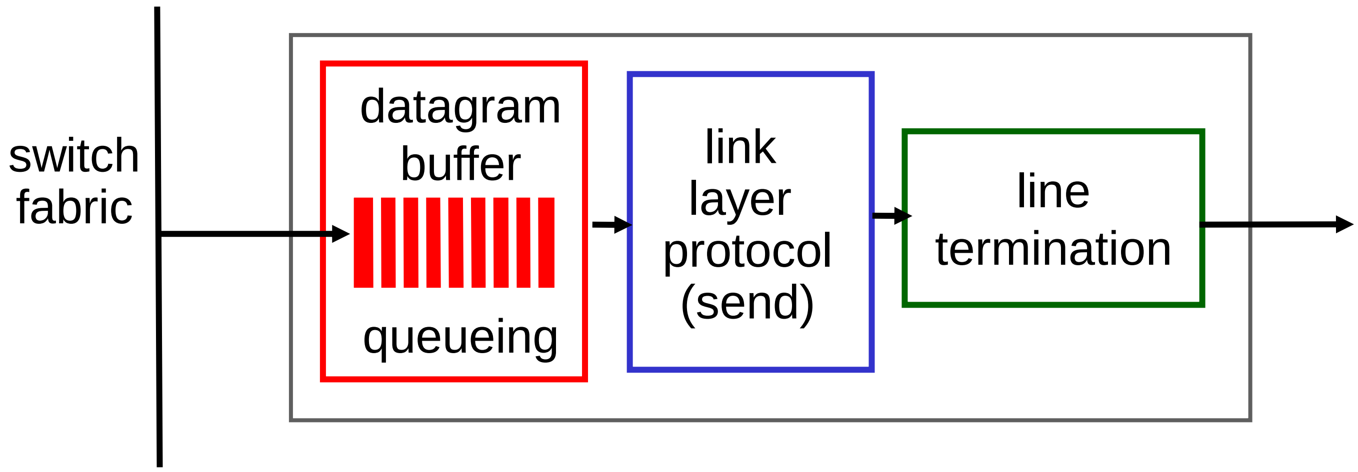


Switching via interconnection network

- When packet from port A needs to forwarded to port Y, controller opens cross point at intersection of two buses
- Note that a packet from port B can be forwarded to port X at the same time

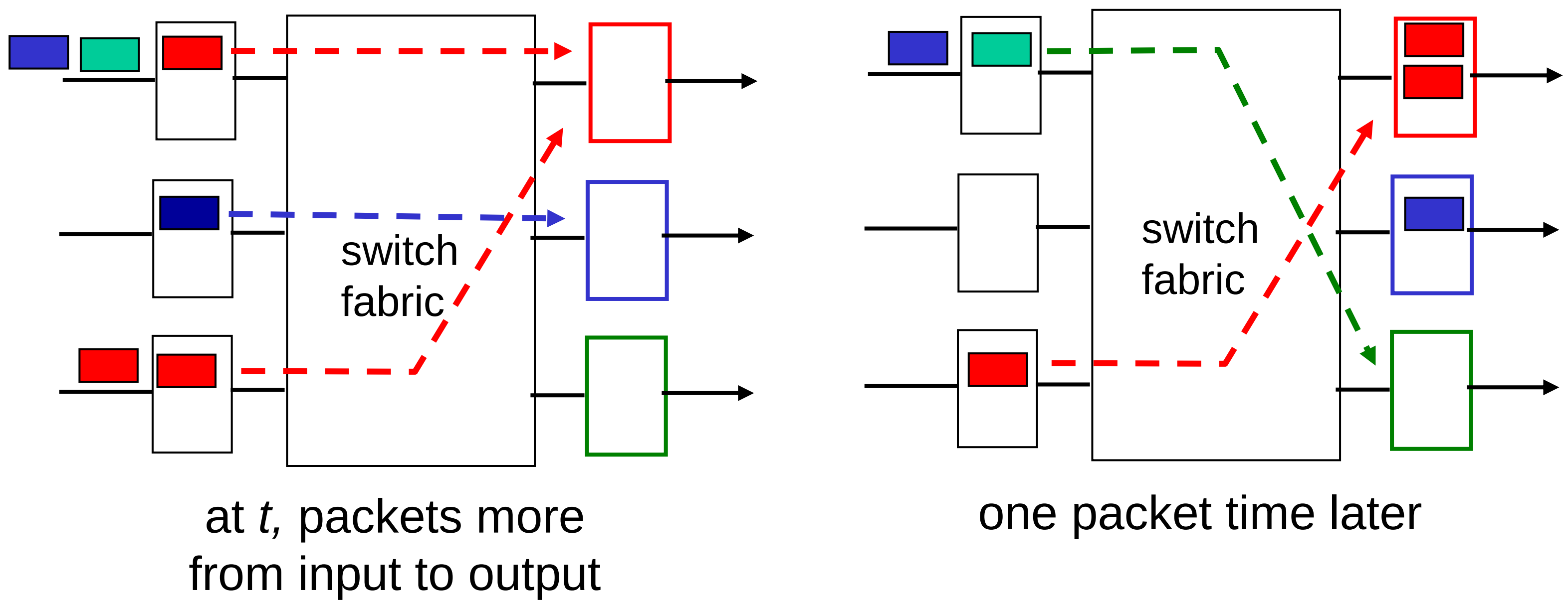


Output ports



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

Output port queueing



- suppose R_{switch} is N times faster than R_{line}
- still have output buffering when multiple inputs send to same output
- *queueing (delay) and loss due to output port buffer overflow!*

How much buffering?

- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: with *N* TCP flows passing through a link, buffering equal to

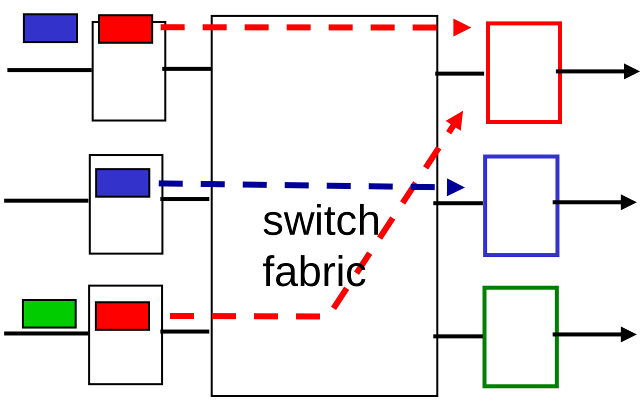
$$\frac{\mathrm{RTT} \cdot \mathrm{C}}{\sqrt{\mathrm{N}}}$$

Active queue management (AQM)

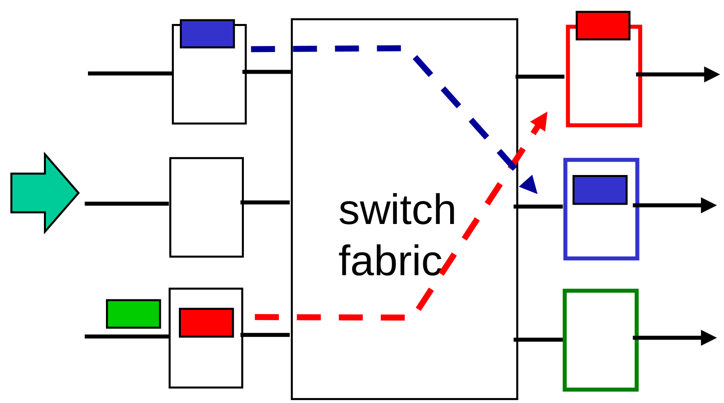
- Random Early Detection (RED) algorithm
  - Queue length less than a minimum threshold min_{th}, admit packet into the queue
  - Queue length greater than a maximum threshold max_{th}, the packet is marked or dropped
  - Queue length in [min_{th}, max_{th}], the packed is marked or dropped with probability

Input port queuing

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



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



one packet time later:
green packet experiences HOL blocking

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

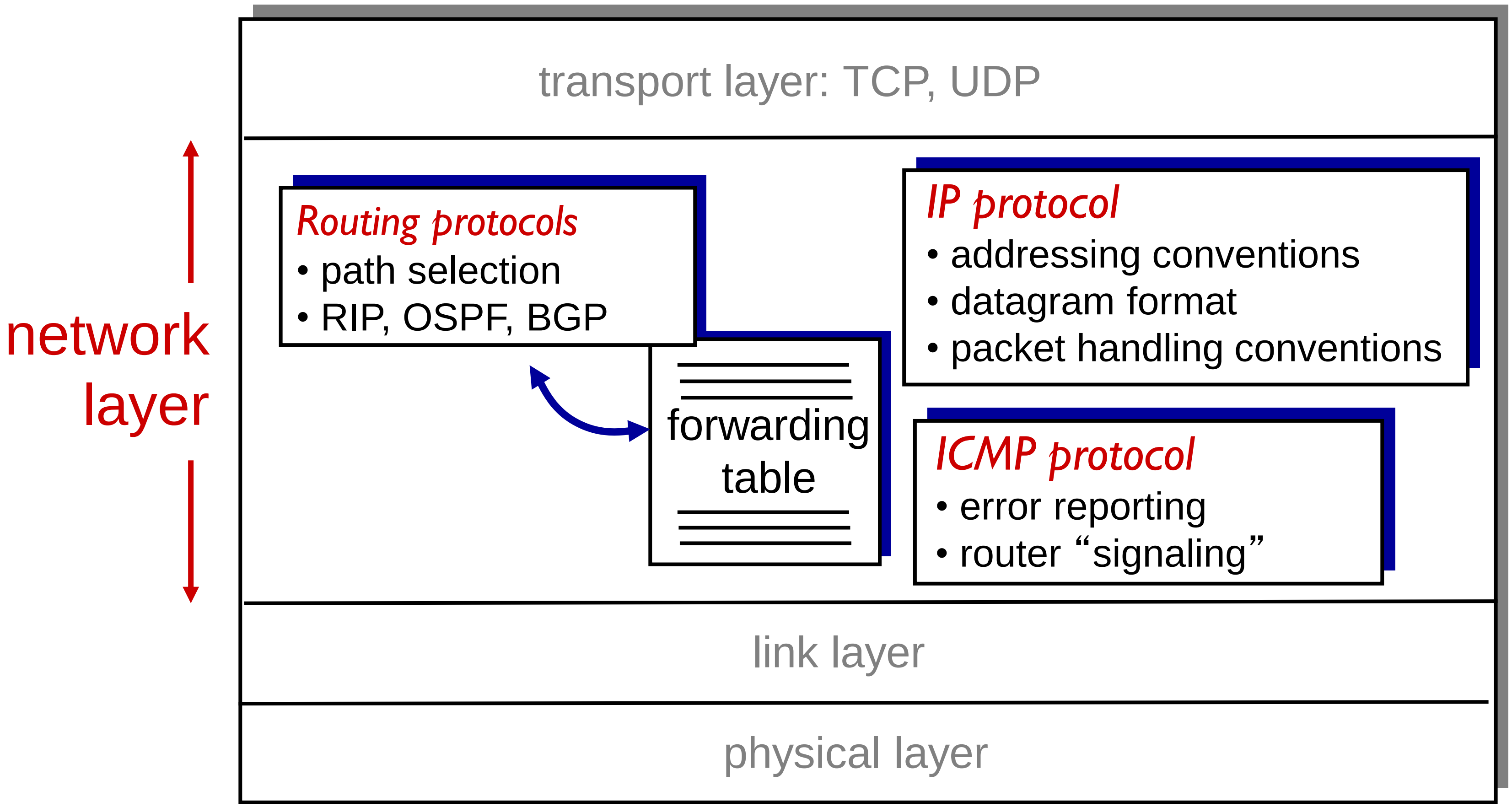
4.6 routing in the Internet

- RIP
- OSPF
- BGP

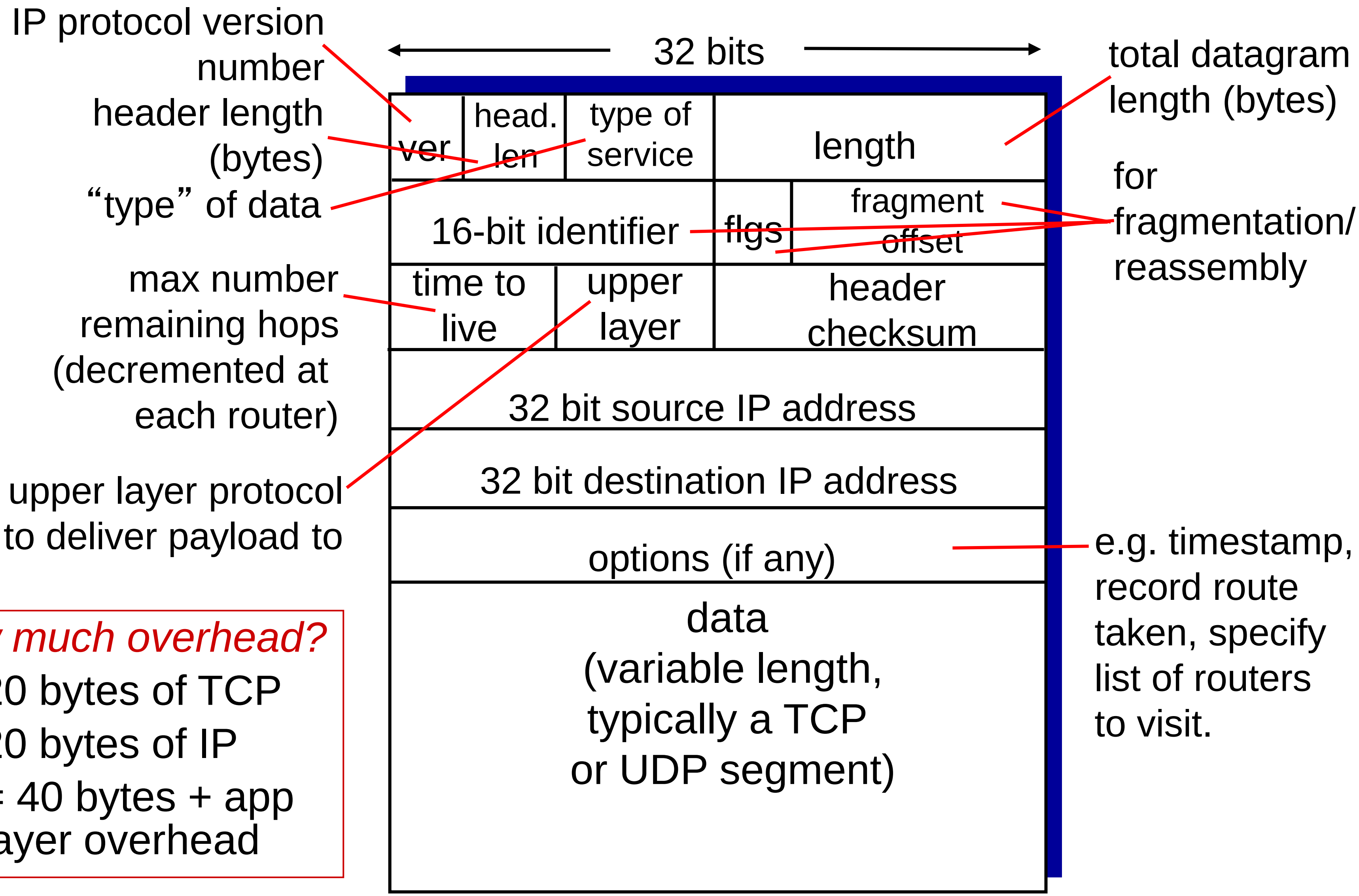
4.7 broadcast and multicast routing

The Internet network layer

host, router network layer functions:



IP datagram format



how much overhead?

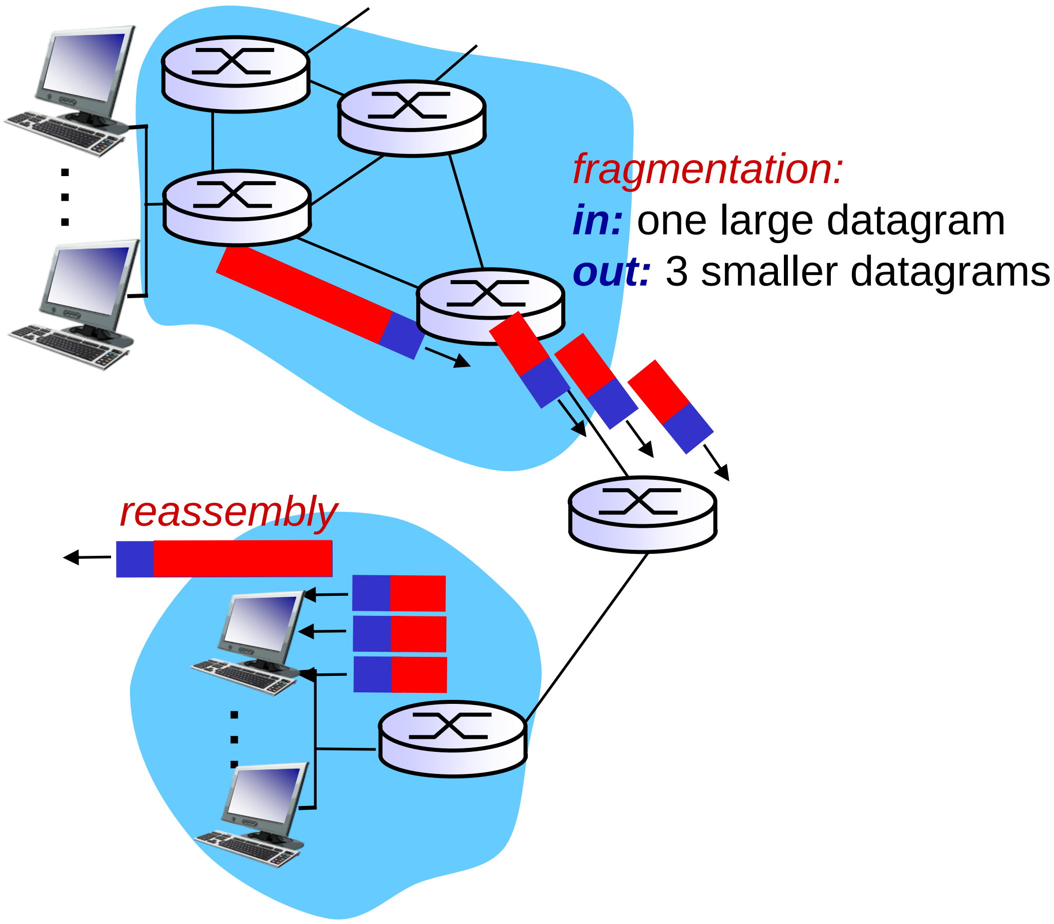
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

IP checksum

- The header checksum is computed by treating each 2 bytes in the header as a number and summing these numbers using 1s complement arithmetic.
- Routers typically discard datagrams for which an error has been detected.
- Checksum must be recomputed and stored again at each router, as the TTL field, and possibly the options field as well, may change.

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



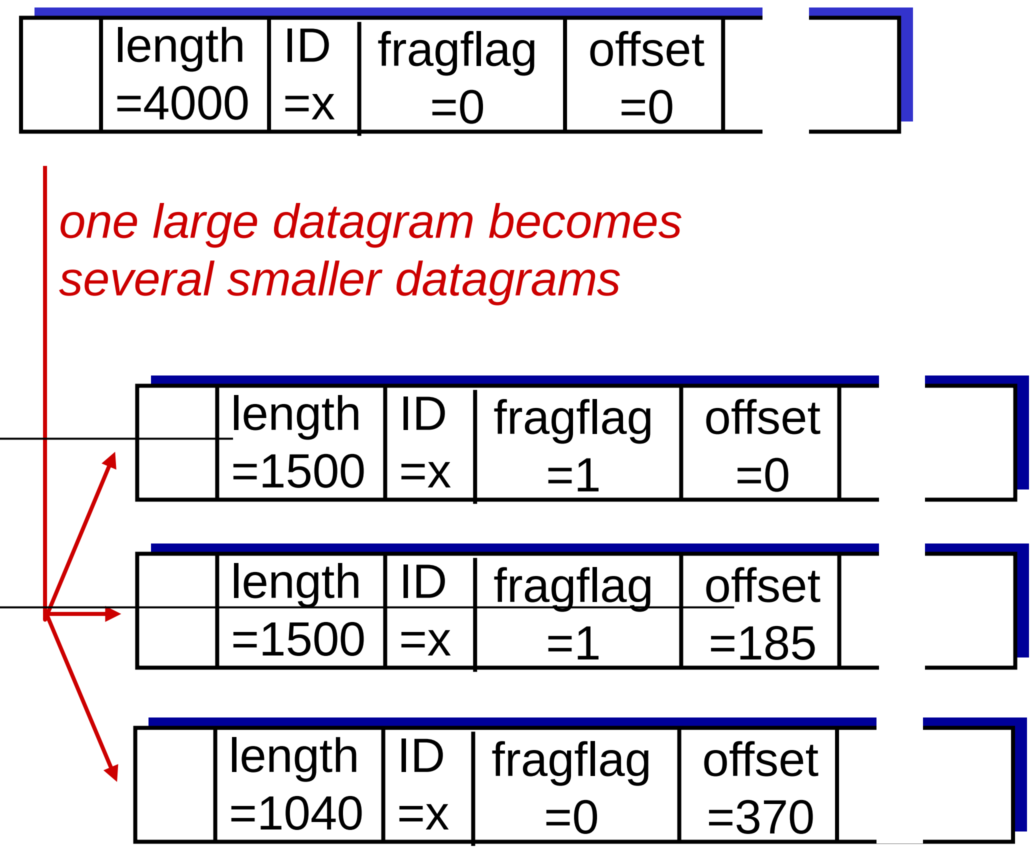
IP fragmentation, reassembly

- Identification: stamped by the sending host
  - Increase for every datagram sent by the host
  - Fragments have a same identification as the original datagram
- Multiple fragments for one original datagram, the last fragment has a flag bit set to 0, whereas all the other fragments have this flag bit set to 1.
- The offset field is used to specify where the fragment fits within the original IP datagram.

IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



1480 bytes in data field

offset = 1480/8 meaning the data should be inserted beginning at byte 1,480.

- Cost of fragmentation:
  - Complication at routers and hosts
  - DoS attack

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

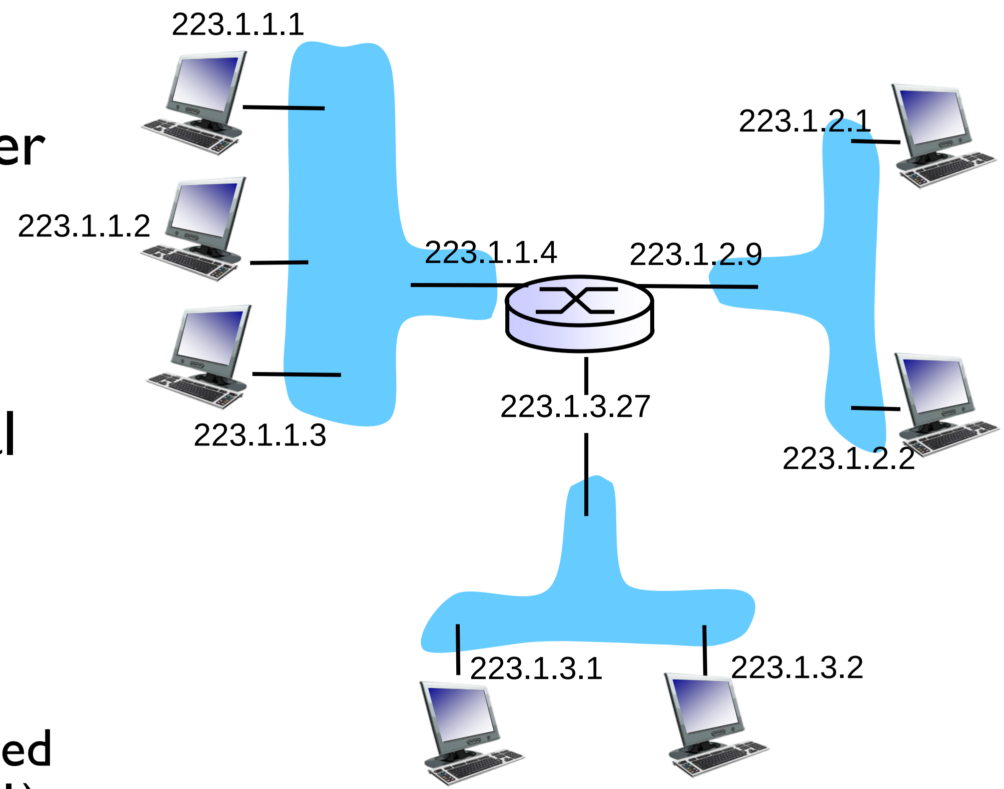
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

IP addressing: introduction

- *IP address:* 32-bit identifier for host, router *interface*
- *Interface/ (接口):* connection between host/router and physical link
  - routers typically have multiple interfaces
  - host typically has one active interface (e.g., wired Ethernet, wireless 802.11)
- *one IP address associated with each interface*

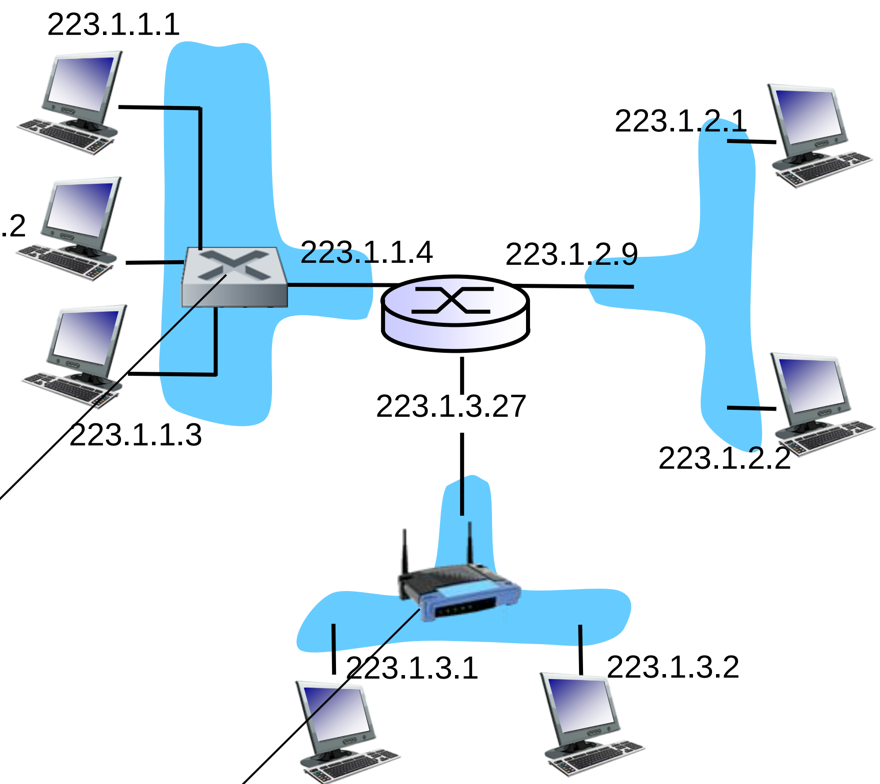


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?

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



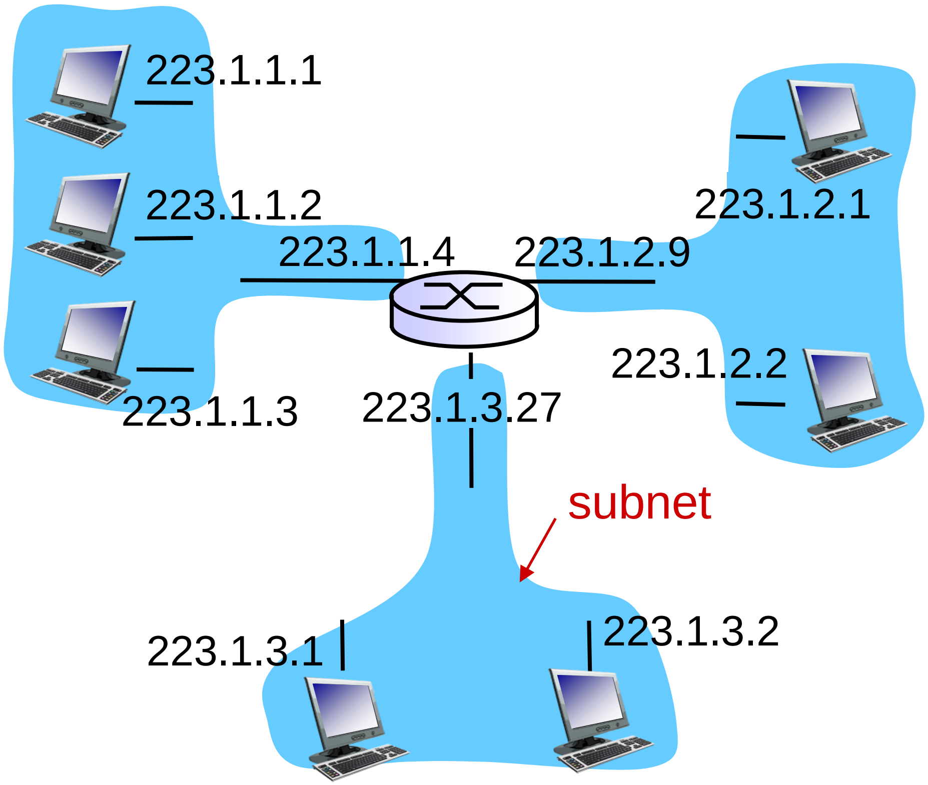
A: wired Ethernet interfaces connected by Ethernet switches

A: wireless WiFi interfaces connected by WiFi base station

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

Subnets

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



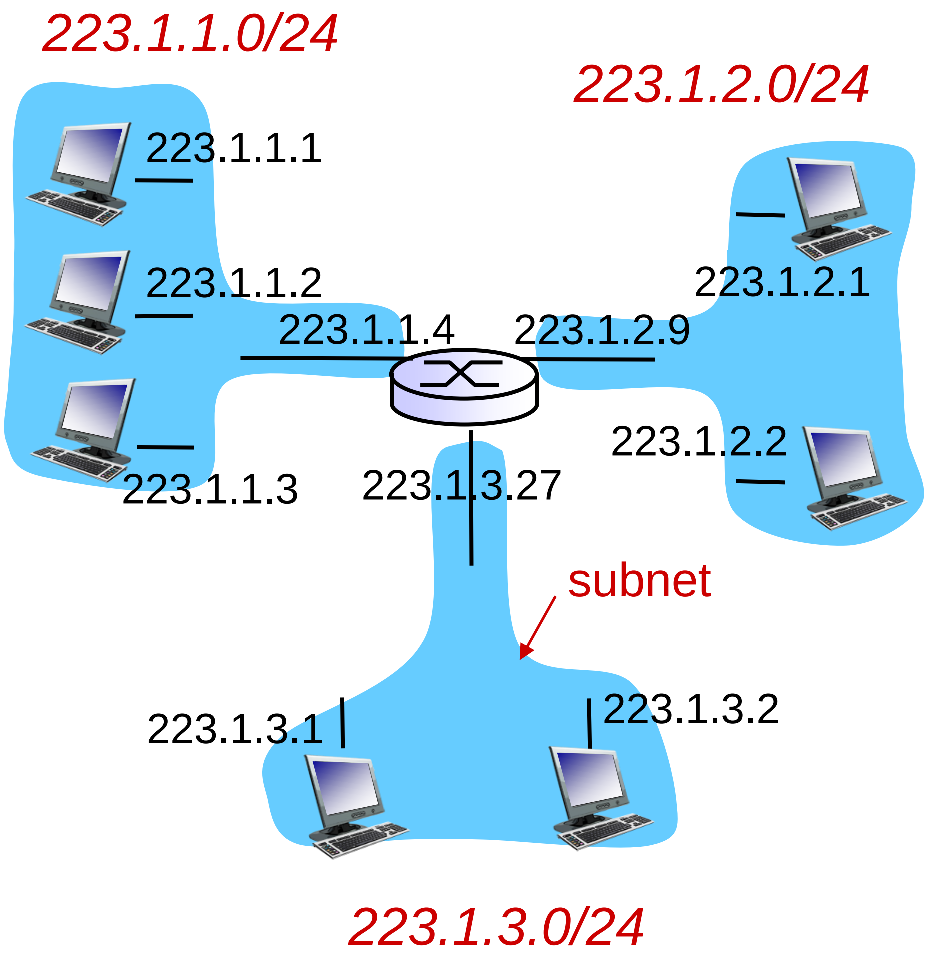
network consisting of 3 subnets

逻辑上、物理上都有要求

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*

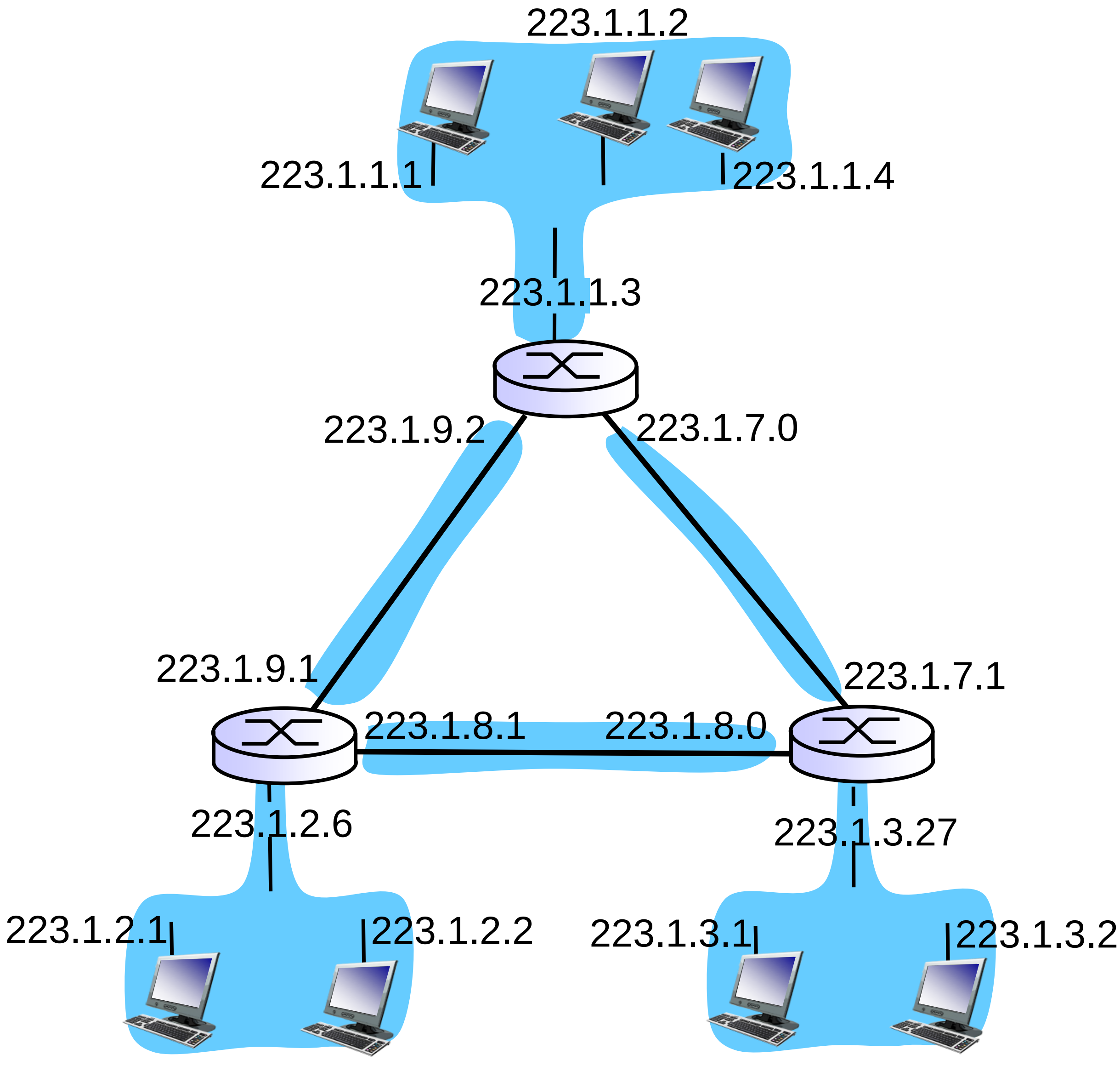


Any additional hosts attached to the 223.1.1.0/24 subnet would be *required to have an* address of the form 223.1.1.xxx.

subnet mask: /24

Subnets

how many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address
- The x most significant bits of an address is called a *prefix*

11001000 00010111 00010000 00000000

200.23.16.0/23

200.23.16.0 --- 200.23.17.255

IP addressing: CIDR

- The lower-order bits may (or may not) have an additional subnetting structure
  - 200.23.16.0/23 may contain two subnets: 200.23.16.0/24 and 200.23.17.0/24.
- Before CIDR, an addressing scheme known as classful addressing,
  - subnets with 8-, 16-, and 24-bit subnet addresses were known as class A, B, and C networks, respectively.
- Broadcast address: 255.255.255.255
  - Delivered to all hosts in same subnet

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP addresses: how to get one?

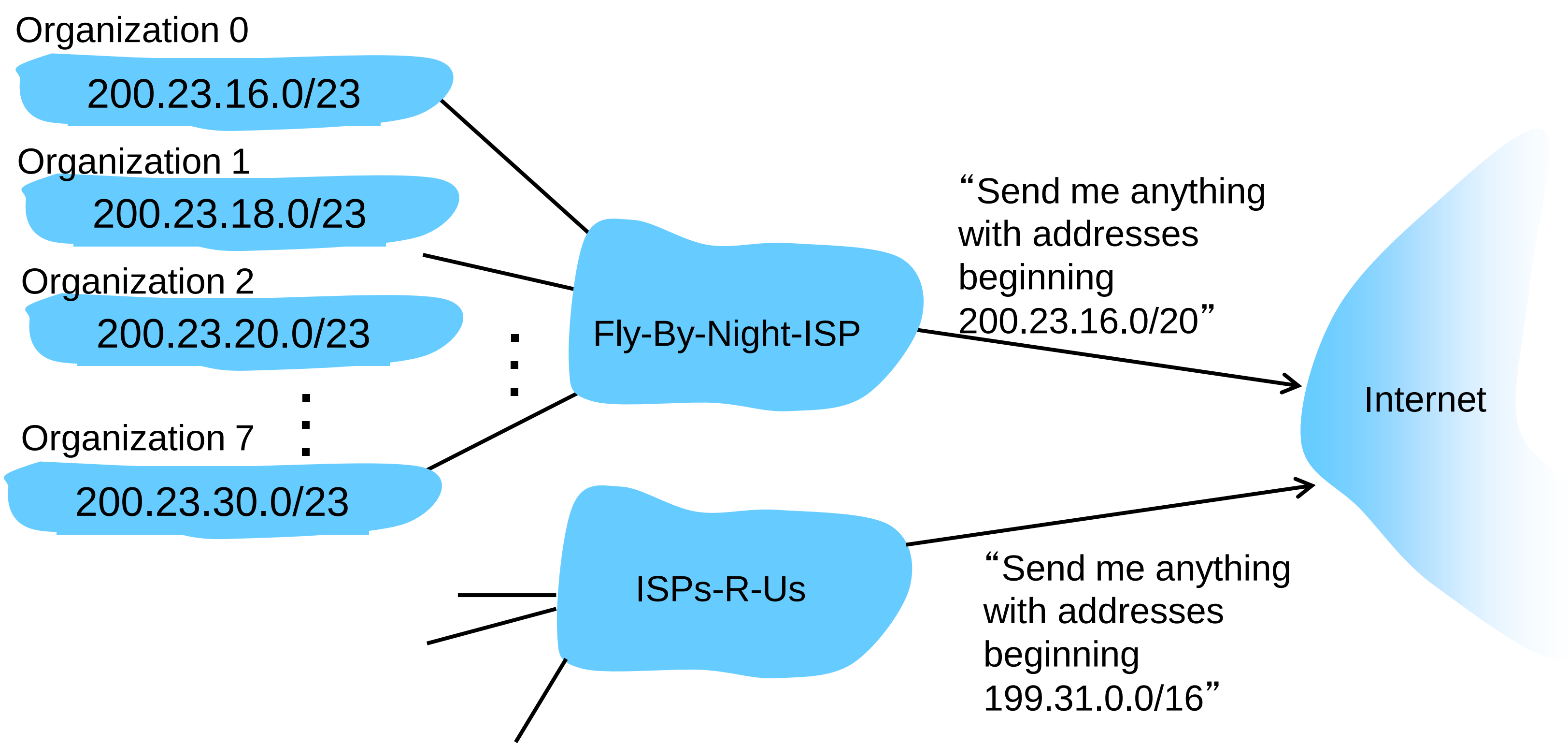
Q: how does *network* get subnet part of IP addr?

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

ISP's block	11001000	00010111	00010000	00000000	200.23.16.0/20
Organization 0	11001000	00010111	00010000	00000000	200.23.16.0/23
Organization 1	11001000	00010111	00010010	00000000	200.23.18.0/23
Organization 2	11001000	00010111	00010100	00000000	200.23.20.0/23
...		…..		….	….
Organization 7	11001000	00010111	00011110	00000000	200.23.30.0/23

Hierarchical addressing: route aggregation

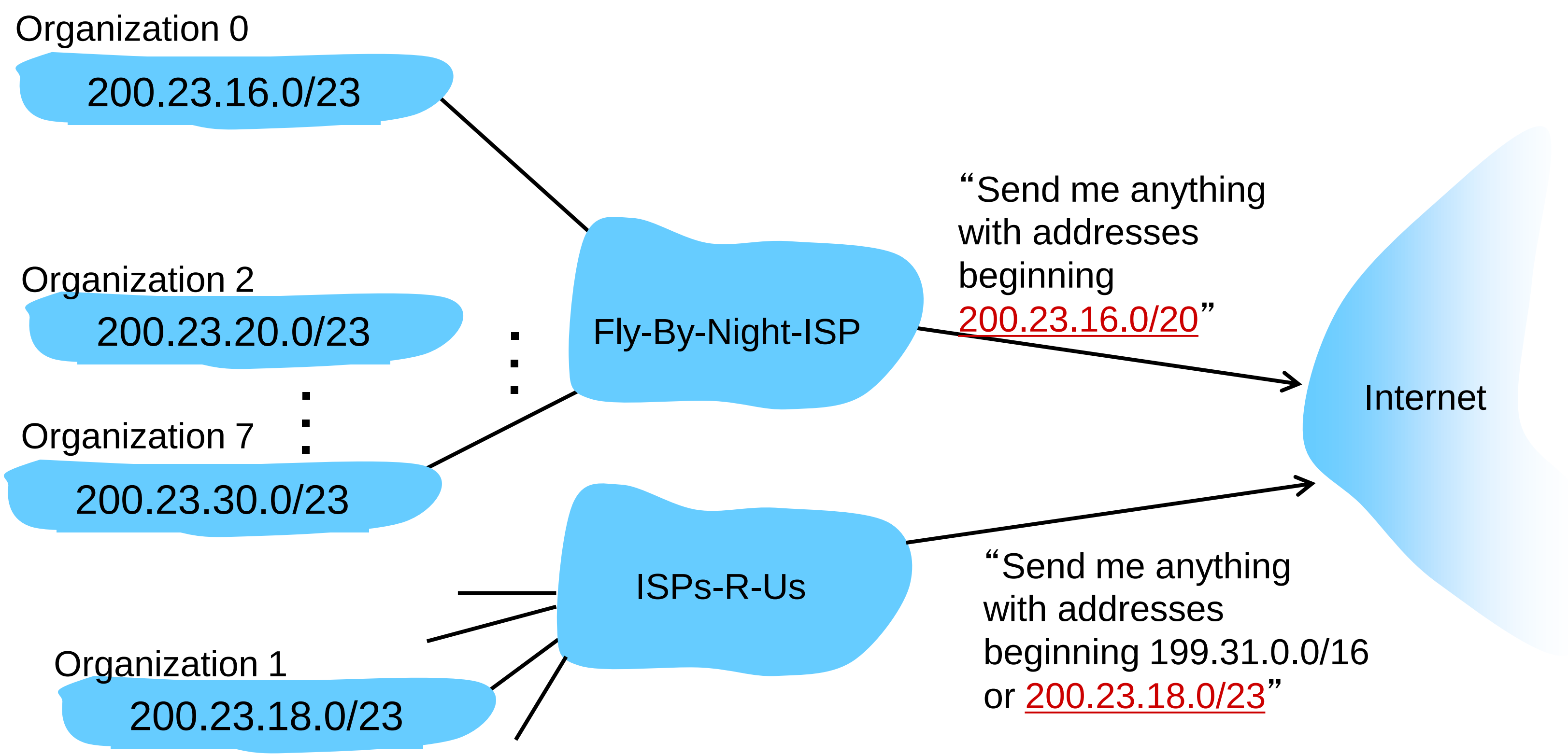
hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

Org. 1 moves
ISPs-R-Us has a more specific route to Organization 1

Traffic for org.1 goes to ISPs-R-Us because of longest prefix match!



IP addressing: how to get a block?

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

The ICANN allocates addresses to regional Internet registries (for example, ARIN, RIPE, APNIC, and LACNIC

IP addresses: how to get one?

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

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

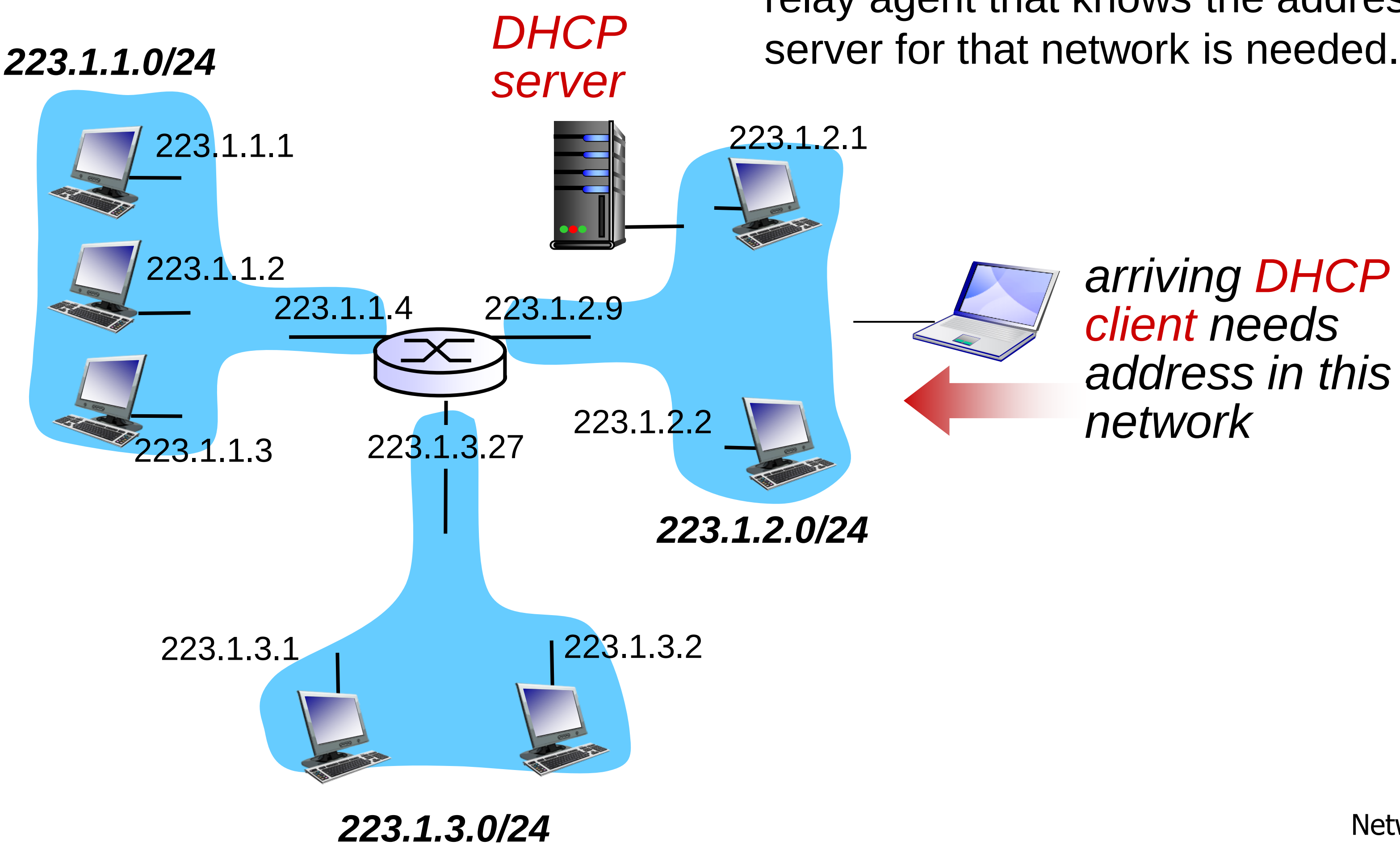
- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

DHCP overview:

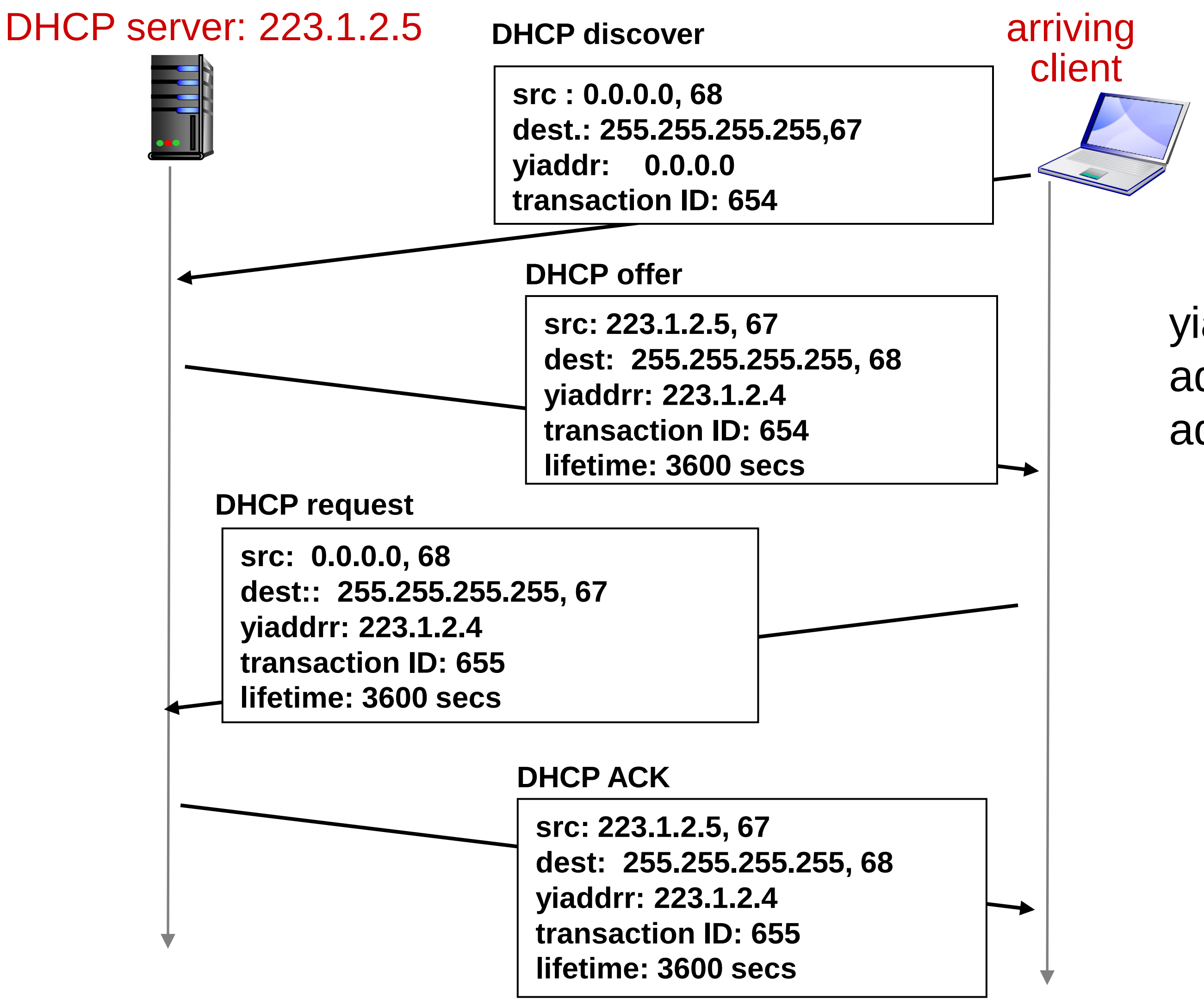
- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario

•In the simplest case, each subnet will have a DHCP server.
•If no server is present on the subnet, a DHCP relay agent that knows the address of a DHCP server for that network is needed.



DHCP client-server scenario



yiaddr: your Internet address, indicates the address being allocated

DHCP: more than IP addresses

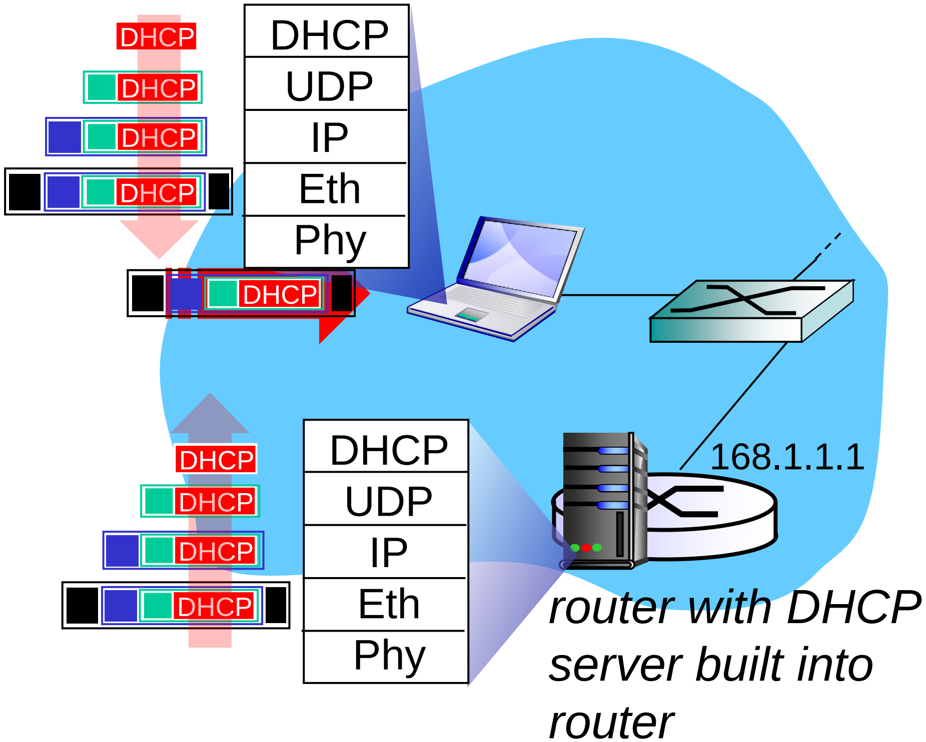
DHCP returns:

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

Mobility issue:

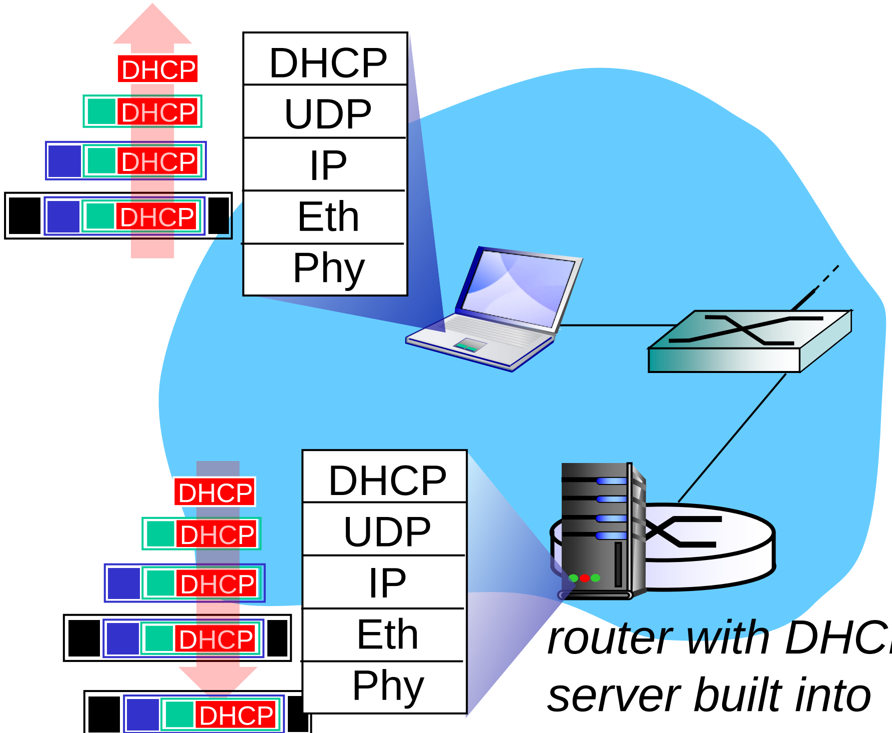
- Obtain new address when moving to a new subnet
- Can not maintain TCP connections
- Mobile IP

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 802.3 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



router with DHCP server built into router

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

DHCP: example

On Windows
C:> ipconfig --release
C:> ipconfig --renew

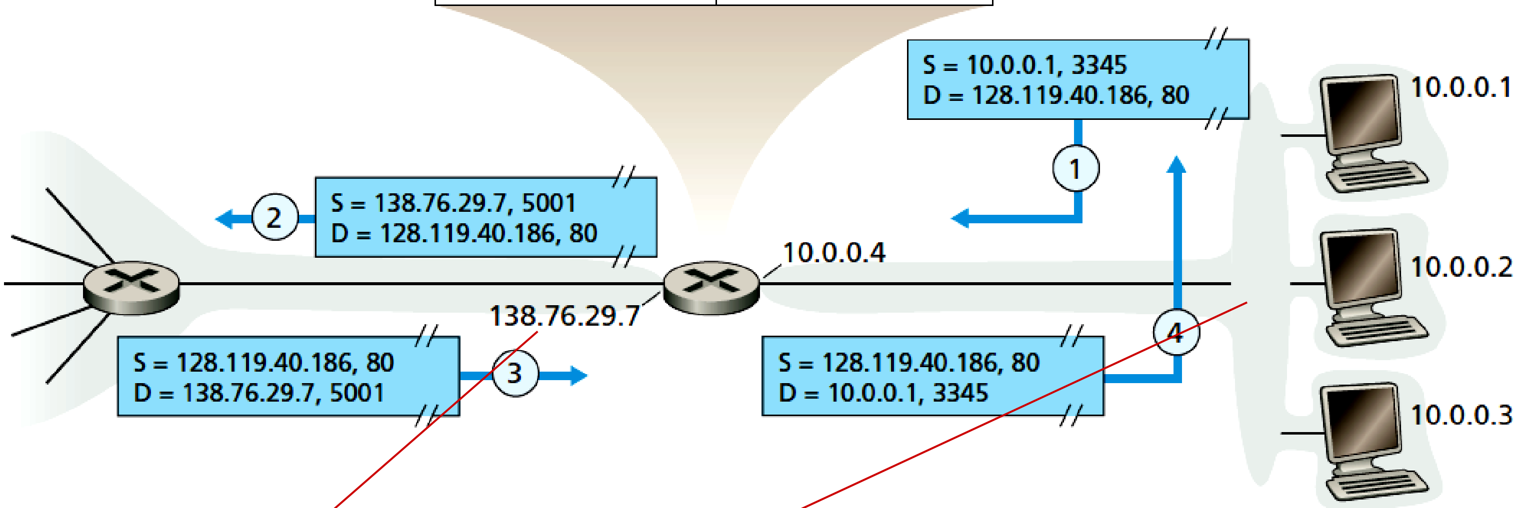
No.	Time	Source	Destination	Protocol	Length	Info
59	8.411174	192.168.1.100	192.168.1.1	DHCP	342	DHCP Release
87	11.872892	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover
95	12.369850	192.168.1.1	192.168.1.100	DHCP	590	DHCP Offer
96	12.370265	0.0.0.0	255.255.255.255	DHCP	362	DHCP Request
97	12.373023	192.168.1.1	192.168.1.100	DHCP	590	DHCP ACK

DHCP offer

```
Your (client) IP address: 192.168.1.100
Next server IP address: 0.0.0.0
Relay agent IP address: 0.0.0.0
Client MAC address: IntelCor_80:f4:34 (8c:70:5a:80:f4:34)
Client hardware address padding: 00000000000000000000
Server name option overloaded by DHCP
Boot file name option overloaded by DHCP
Magic cookie: DHCP
Option: (53) DHCP Message Type (Offer)
Option: (54) DHCP Server Identifier
Option: (1) Subnet Mask
    Length: 4
    Subnet Mask: 255.255.255.0
Option: (51) IP Address Lease Time
    Length: 4
    IP Address Lease Time: (7200s) 2 hours
```

NAT: network address translation

NAT translation table	
WAN side	LAN side
138.76.29.7, 5001	10.0.0.1, 3345
. . .	. . .



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 addressable, visible by outside world (a security plus)

NAT: network address translation

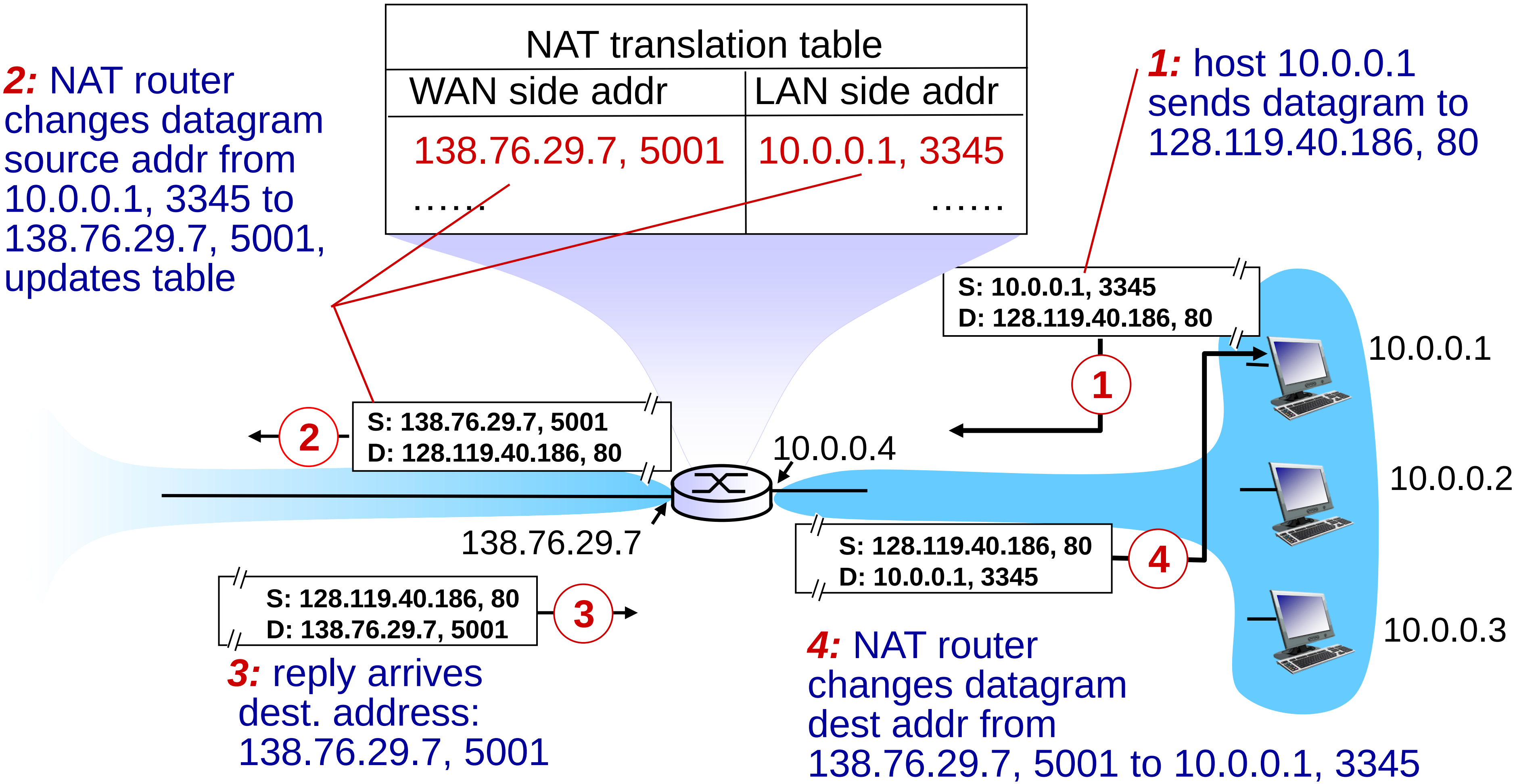
- 3 private IP address ranges
  - 10.0.0.0/8 : 10.0.0.0 to10.255.255.255.
  - 172.16.0.0/12 : 172.16.0.0 to 172.31.255.255.
  - 192.168.0.0/16: 192.168.0.0 to 192.168.255.255

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



NAT translation table

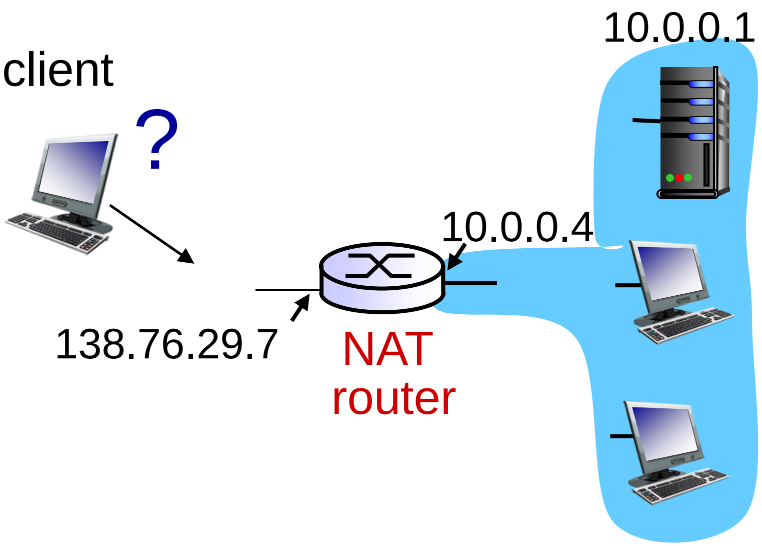
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - Port should be used for addressing process
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6

NAT traversal problem

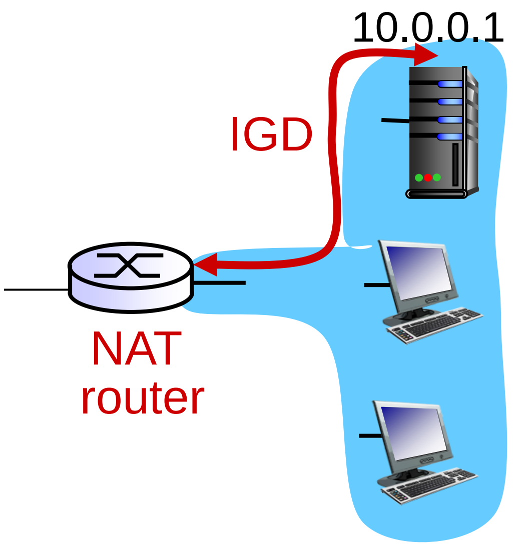
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- *solution1:* statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 25000) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

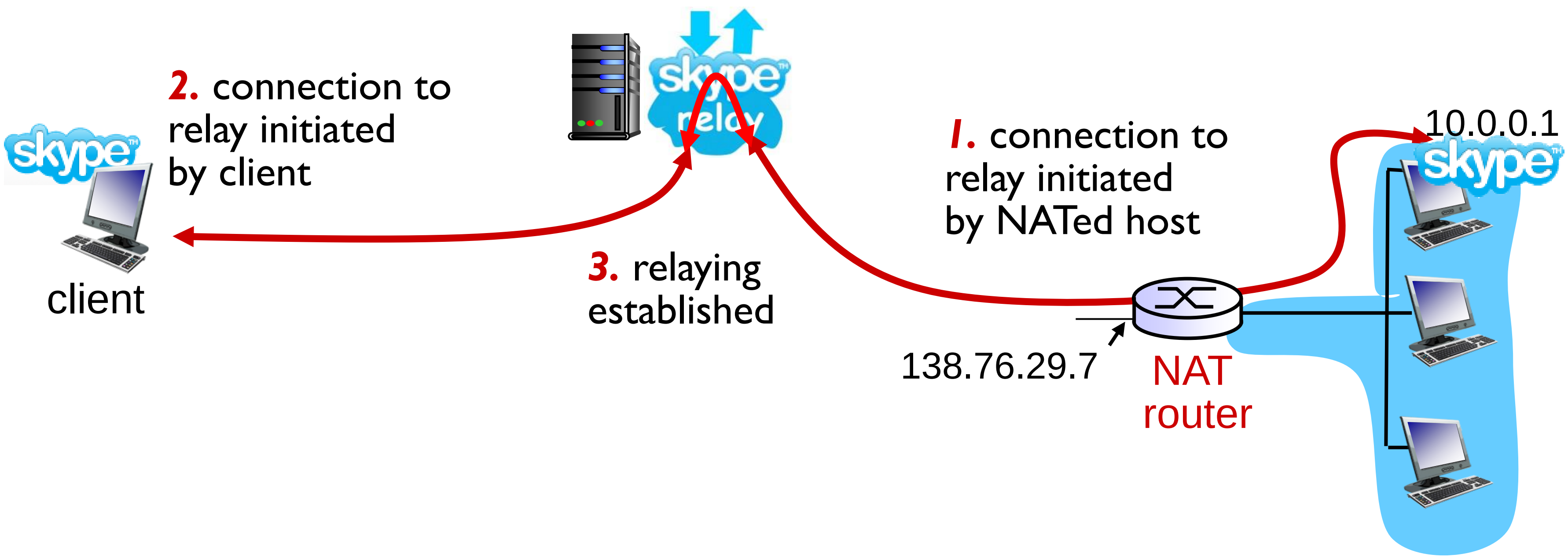
- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - request a NAT mapping between its (private IP address, private port number) and the (public IP address, public port number)

  i.e., automate static NAT port map configuration



NAT traversal problem

- *solution 3:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between two connections



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

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

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 (8-bit), code (8-bit) plus first 8 bytes of the IP datagram that cause error

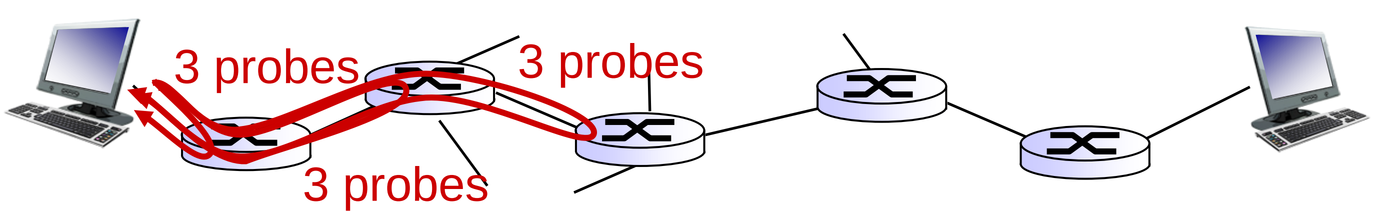
Type	Code	description
0	**0**	**echo reply (ping)**
3	**0**	**dest. network unreachable**
3	1	dest host unreachable
3	2	dest protocol unreachable
3	**3**	**dest port unreachable**
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	**0**	**echo request (ping)**
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 *n*th 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

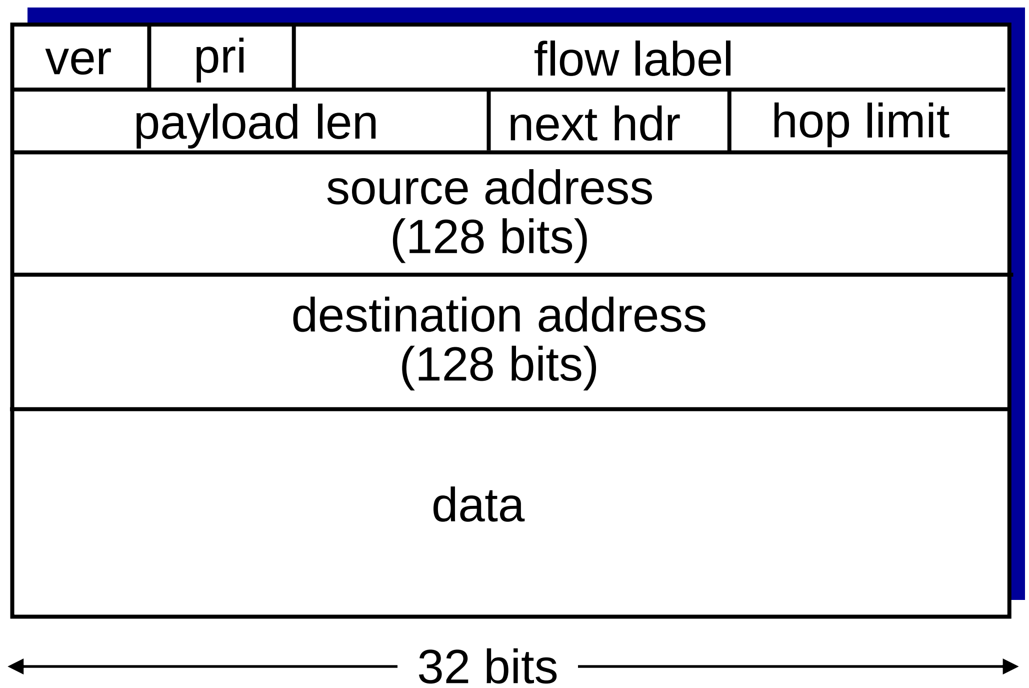


IPv6: motivation

- *initial motivation:* 32-bit address space soon to be completely allocated.
  - 已经全部分配
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - Internet of Things

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



IPv6 datagram format

IPv6 datagram format:

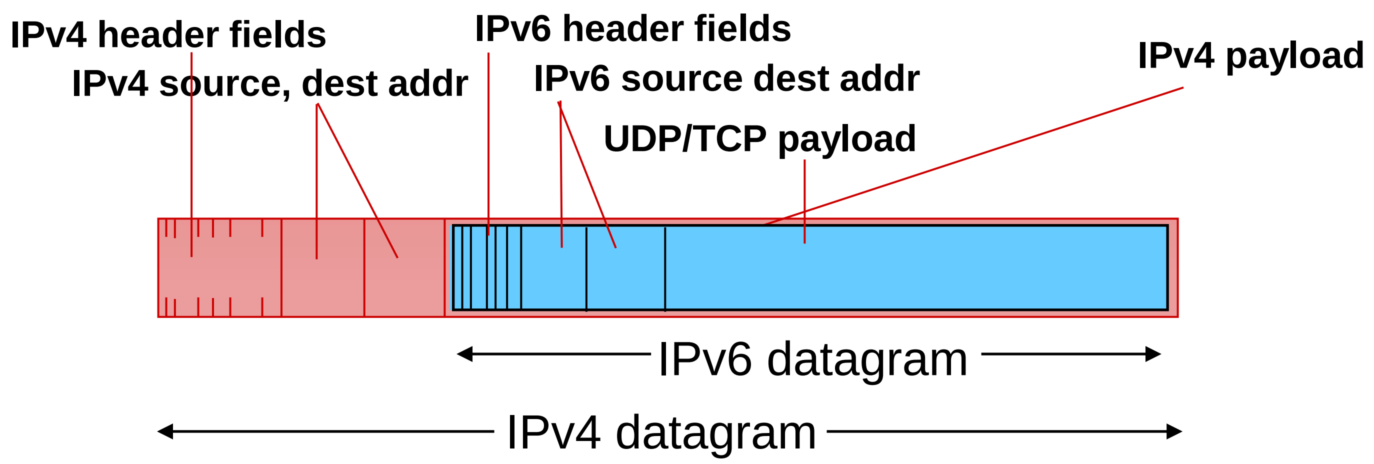
- fixed-length 40 byte header
- no fragmentation allowed
  - If an IPv6 datagram received by a router is too large to be forwarded over the outgoing link, the router simply drops the datagram and sends a "Packet Too Big" ICMP error message back to the sender.

Other changes from IPv4

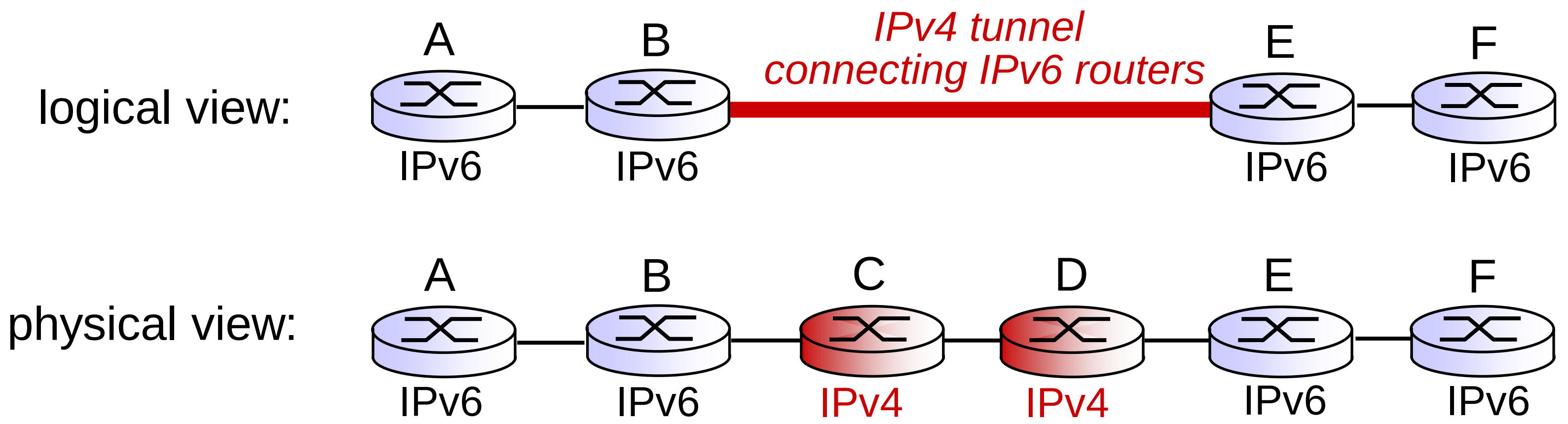
- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition from IPv4 to IPv6

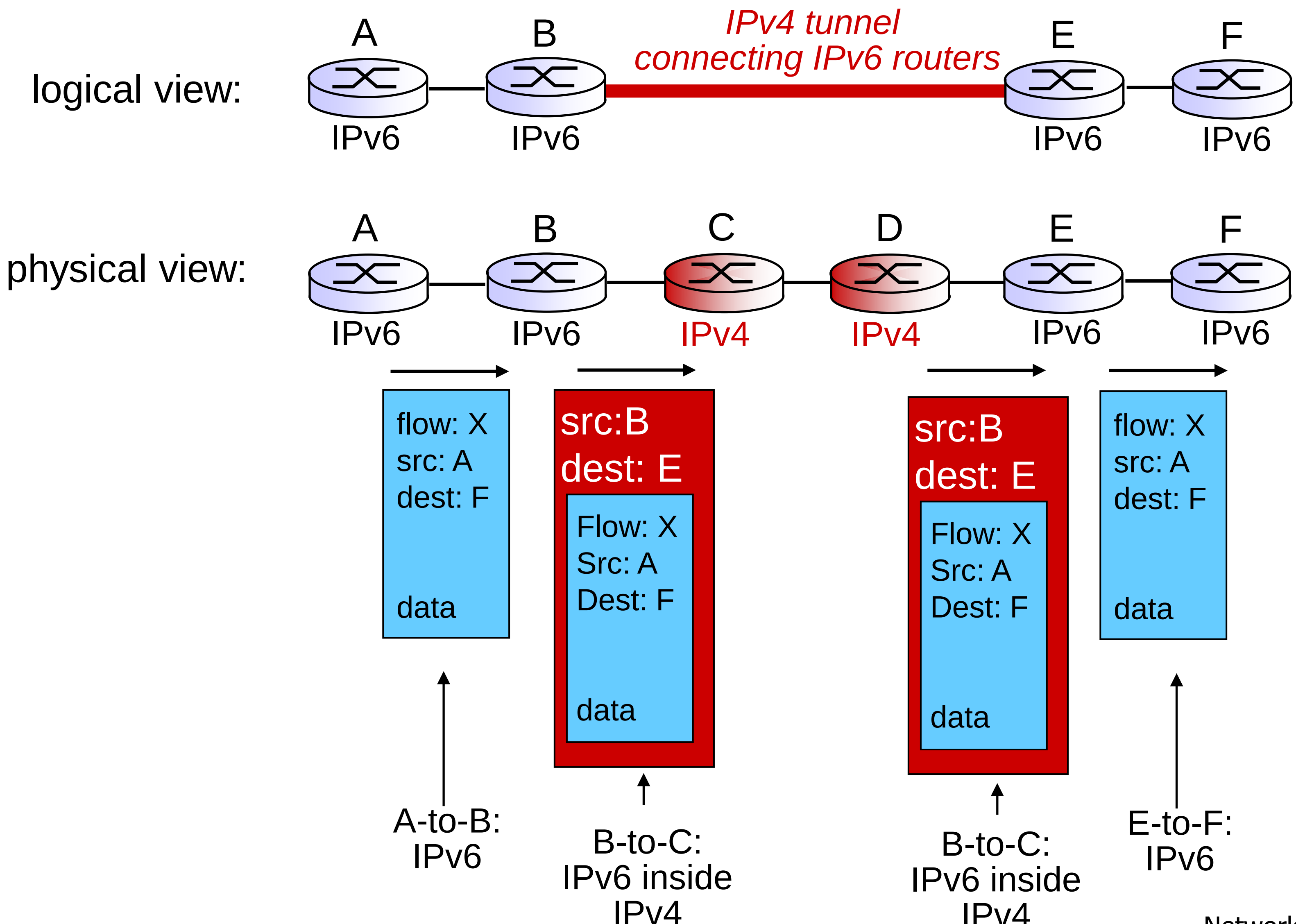
- not all routers can be upgraded simultaneously
  - no "flag days", Internet is huge
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling

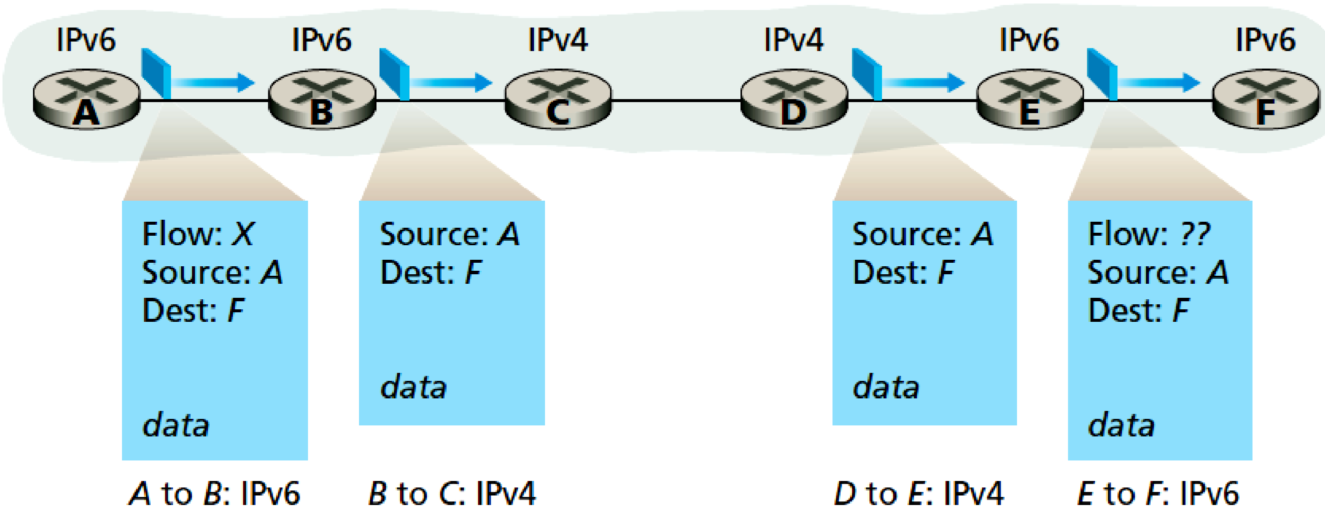

logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling


IPv4 tunnel
connecting IPv6 routers
logical view:
A
B
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

Dual-stack

- Router has the ability to send and receive both IPv4 and IPv6 datagrams
- IPv4 to IPv4 transition loses v6 field values



even though E and F can exchange IPv6 datagrams, the arriving IPv4 datagrams at E from D do not contain all of the fields that were in the original IPv6 datagram sent from A.

IPSec

- Popular secure network-layer protocol
- Transport mode
  - *Connection oriented*: two hosts establish an IPsec session
  - *Cryptographic agreement:* agree on cryptographic algorithms and keys
  - *Encryption of IP datagram payloads:* Sender encrypts the IP datagram payload.
  - *Data integrity:* Receiver verifies that the datagram' s header fields and encrypted payload were not modified
  - *Origin authentication:* ensure that source IP is the actural sender

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

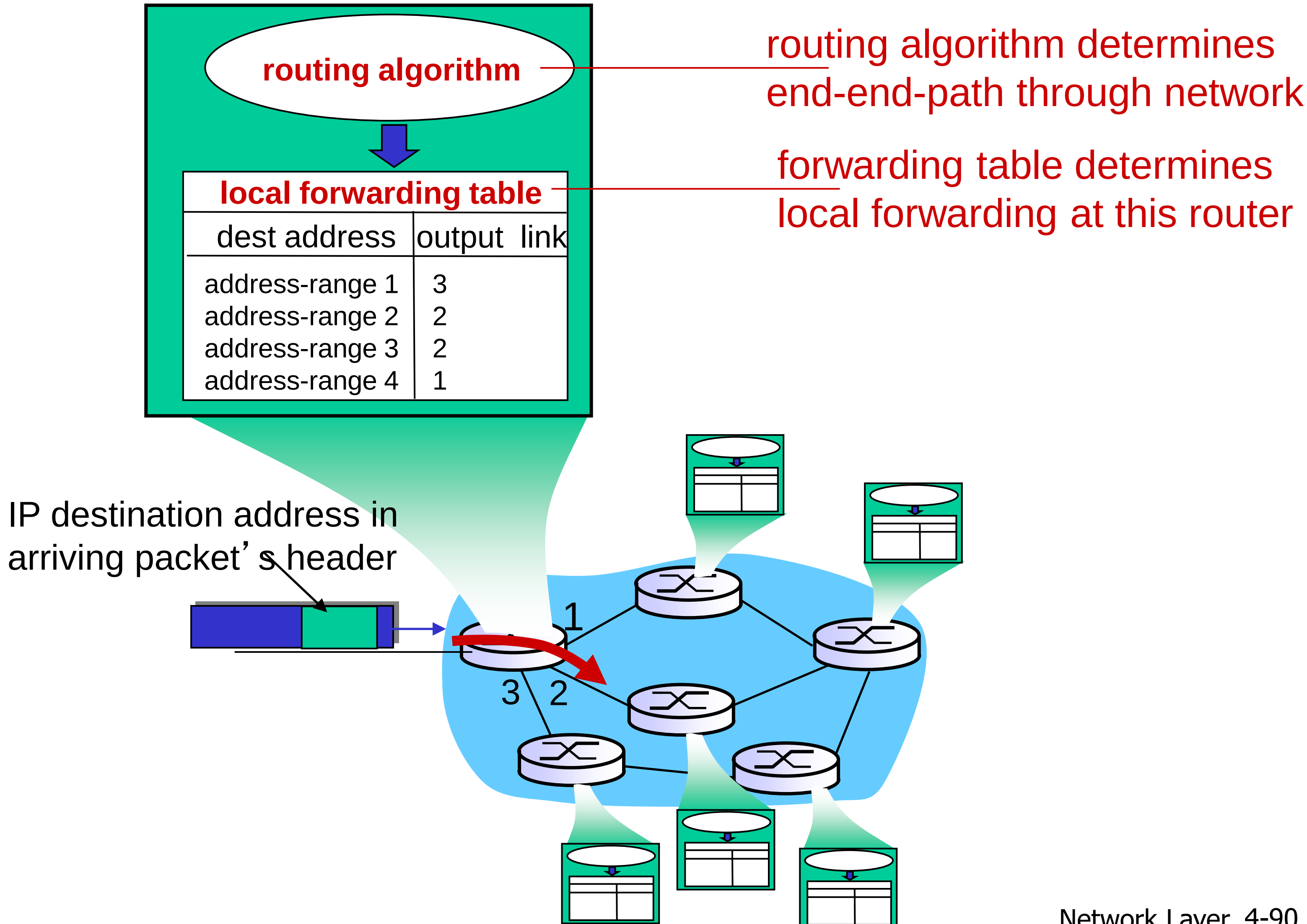
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

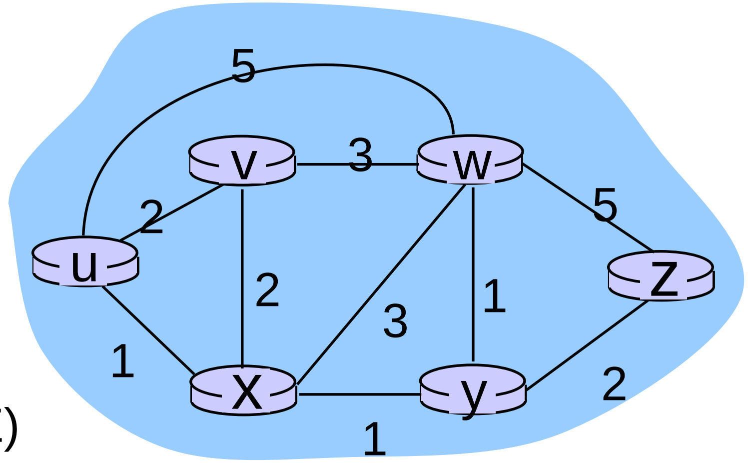
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Interplay between routing, forwarding



Graph abstraction



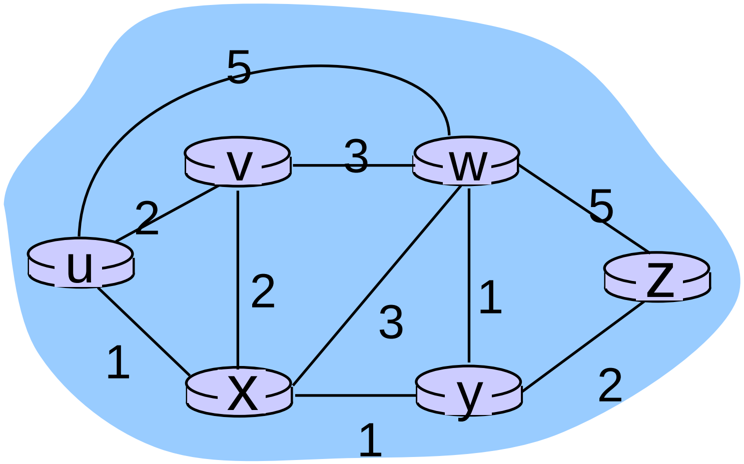
Undirected graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph abstraction: costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or related to physical length, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- "link state（链路状态）" algorithms

decentralized:

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

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

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

Q: load sensitive or insensitive:

- Today's Internet routing is load insensitive

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
- RIP
- OSPF
- BGP

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 destinations

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

Dijsktra'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'
```

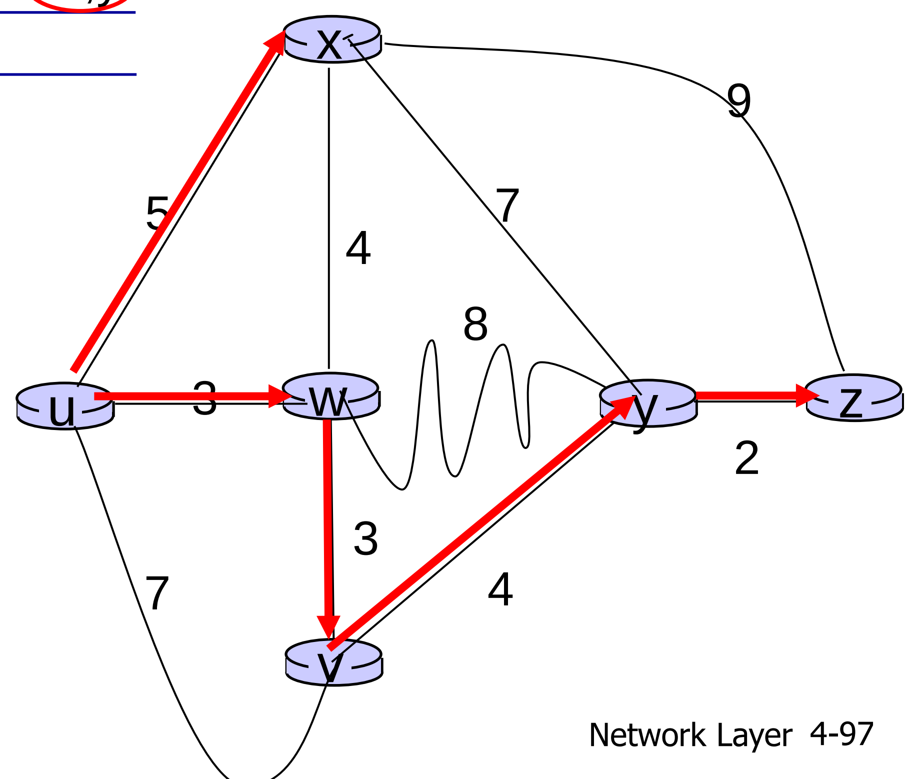
Dijkstra's algorithm: example

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

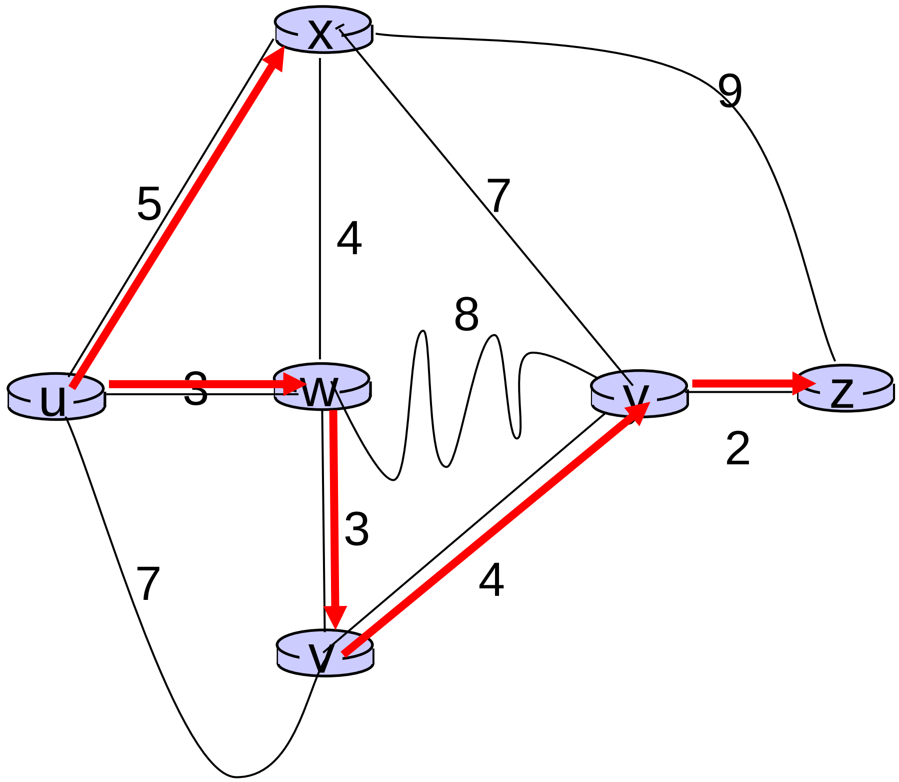
$$e.g.,\ D(v) = \min(D(v), D(w) + c(w,v))$$
$$= \min\{7, 3+3\} = 6$$

notes:

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



Dijkstra's algorithm: example



resulting forwarding table in u:

destination	link
v	(u,w)
x	(u,x)
y	(u,w)
w	(u,w)
z	(u,w)

Dijkstra's algorithm, discussion

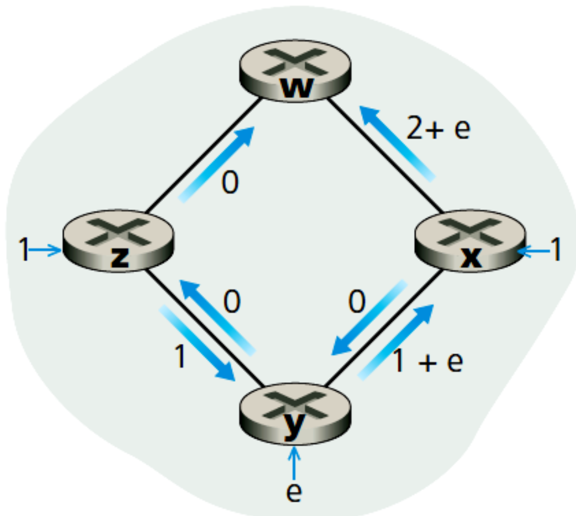
algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N'
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

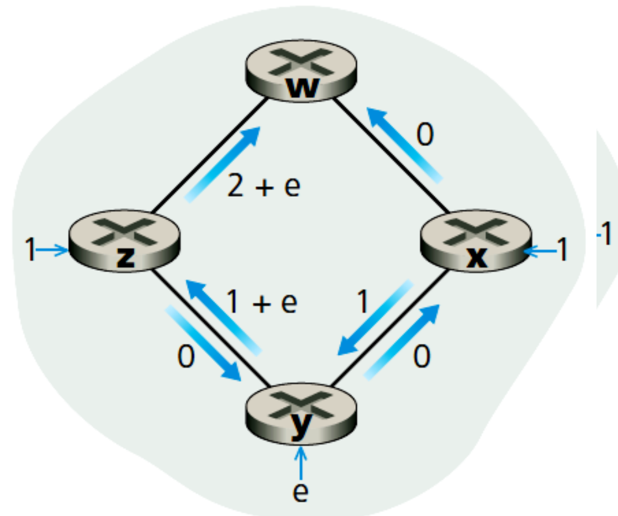
Dijkstra's algorithm, discussion

oscillations possible:

- e.g., suppose link cost equals amount of carried traffic:



c. *x, y, z* detect better path to *w*, counterclockwise



d. *x, y, z,* detect better path to *w*, clockwise

How to avoid?

- Link cost not depend on traffic amount
- Routers not run LS algorithm at the same time

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
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

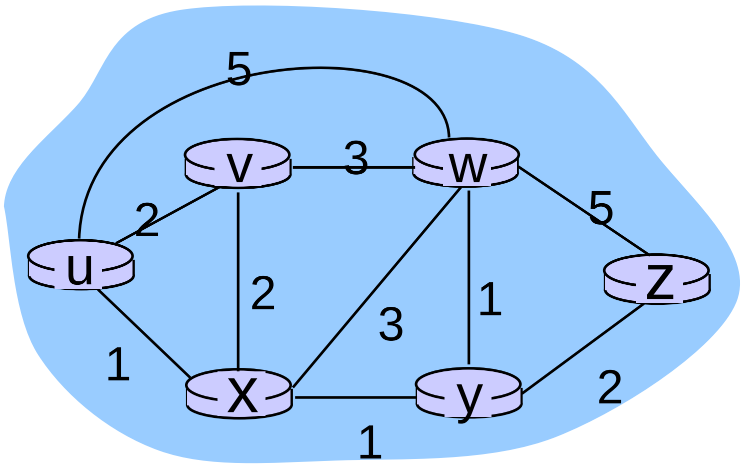
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

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

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

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:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

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

Distance vector algorithm

```
Initialization:
   for all destinations y in N:
      D_x(y) = c(x,y)   /* if y is not a neighbor then c(x,y) = ∞ */
   for each neighbor w
      D_w(y) = ? for all destinations y in N
   for each neighbor w
      send distance vector D_x = [D_x(y): y in N] to w

loop
   wait (until I see a link cost change to some neighbor w or
         until I receive a distance vector from some neighbor w)

   for each y in N:
      D_x(y) = min_v{c(x,v) + D_v(y)}

   if D_x(y) changed for any destination y
      send distance vector D_x = [D_x(y): y in N] to all neighbors

forever
```

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:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

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

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

node x table

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

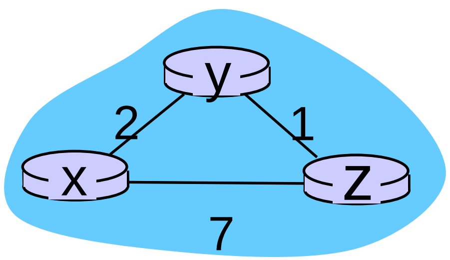
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

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



time

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

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

node x table

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

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

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

node y table

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

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

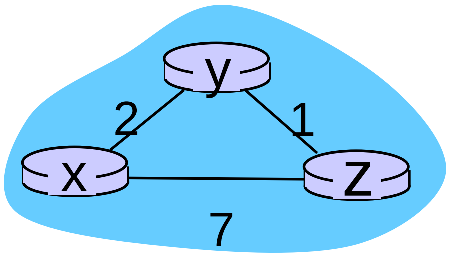
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

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

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

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

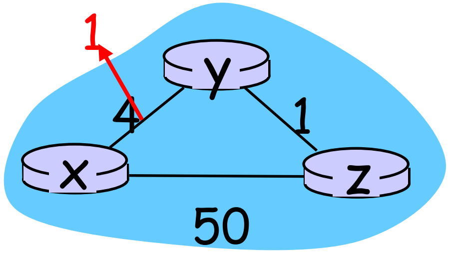


time

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*' s update, updates its distance table. *y*' s least costs do *not* change, so *y* does *not* send a message to *z*.

Distance vector: link cost changes

t_0: y detect link-cost change, y compute its new min-cost to x as

$$D_y(x) = \min\{c(y,x) + D_x(x),\ c(y,z) + D_z(x)\} = \min\{60 + 0,\ 1 + 5\} = 6$$

based on the belief that z can reach x with a minimum path of cost 5

t_1: y inform z of its new DV

t_2: z receives y's new DV, update its minimum cost to z as

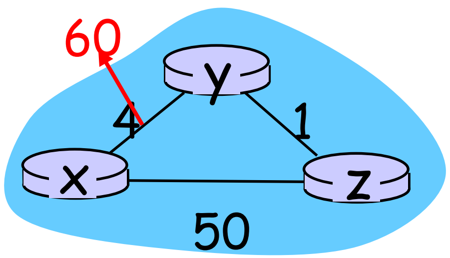
t_3: y update its DV and send to z $D_z(x) = \min\{50 + 0, 1 + 6\} = 7$

….

Count-to-infinity problem

link cost changes:

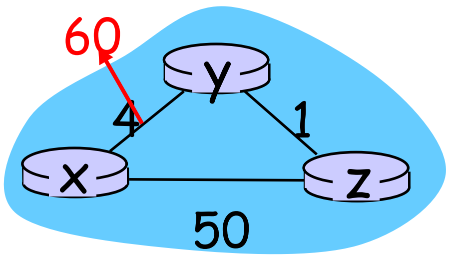
- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes



Distance vector: link cost changes

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)



t_0: y updates its table and continues to route directly to x, and informs z of its new cost to x, that is, $D_y(x) = 60$.
t_1: After receiving the update, z immediately shifts its route to x to be via the direct (z, x) link at a cost of 50.
t_2: z now informs y that $D_z(x) = 50$.
t_3: After receiving the update from z, y updates its distance table with $D_y(x) = 51$. Also, y poisons the reverse path from z to x by informing z that $D_y(x) = \infty$

Can solve the general count-to-infinity problem?
No. Doesn't work on more complicate networks

Comparison of LS and DV algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

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

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
- RIP
- OSPF
- BGP

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 links!

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

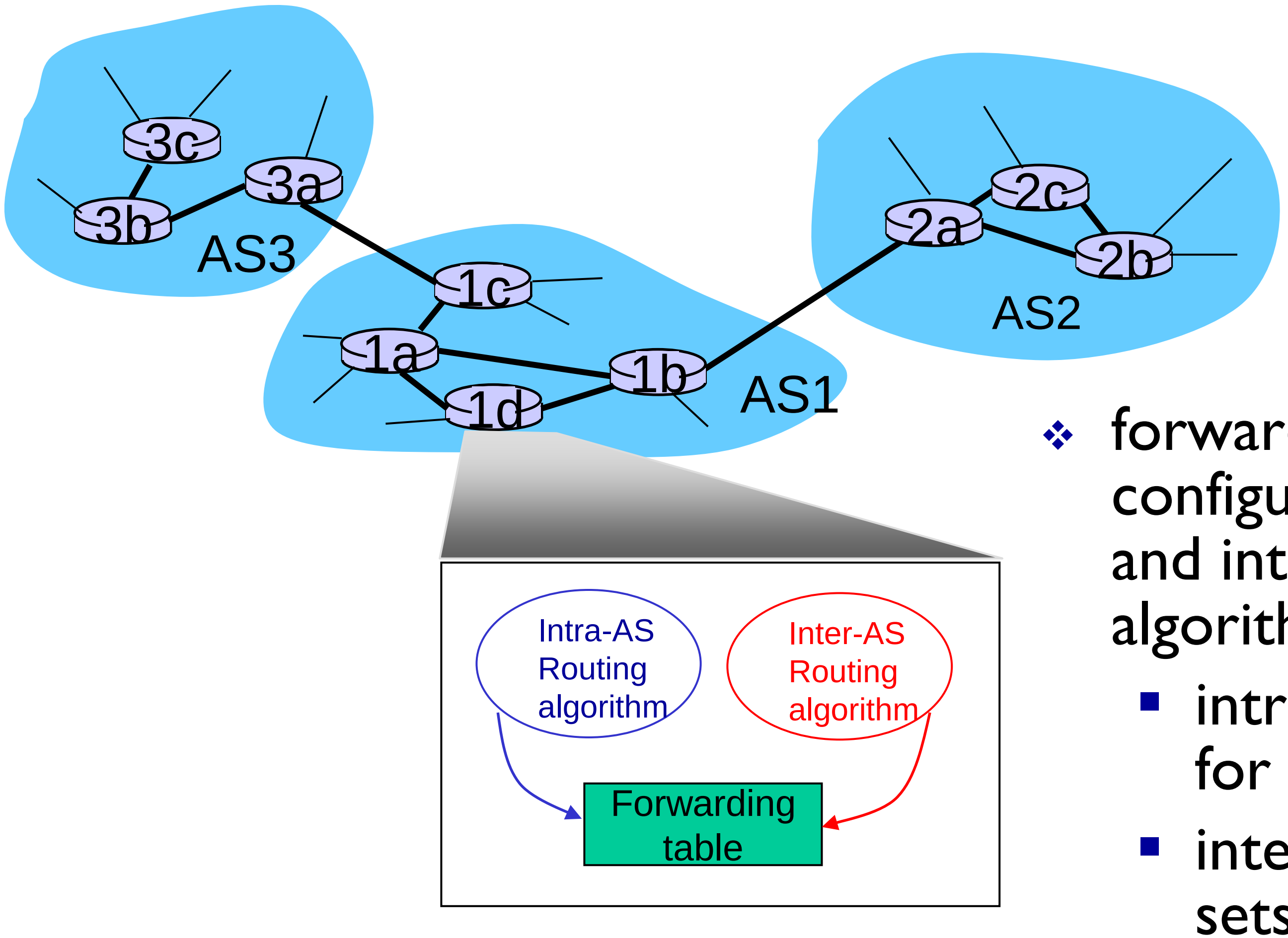
Hierarchical routing

- collect routers into regions, "autonomous systems" (AS)
- Each AS within an ISP
  - ISP may consist of one or more ASes
- 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

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

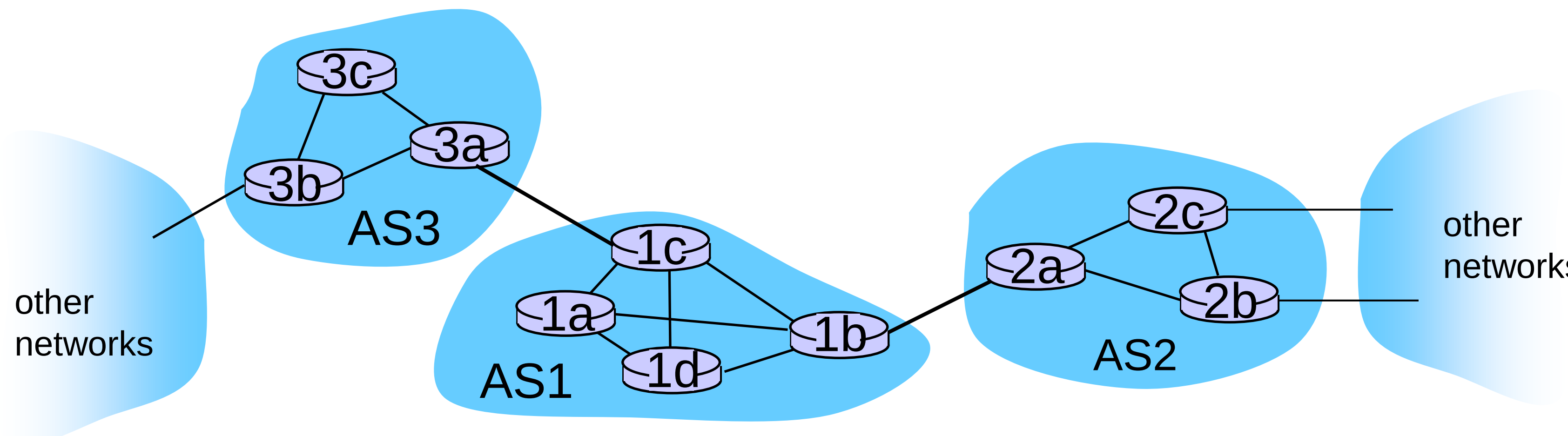
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

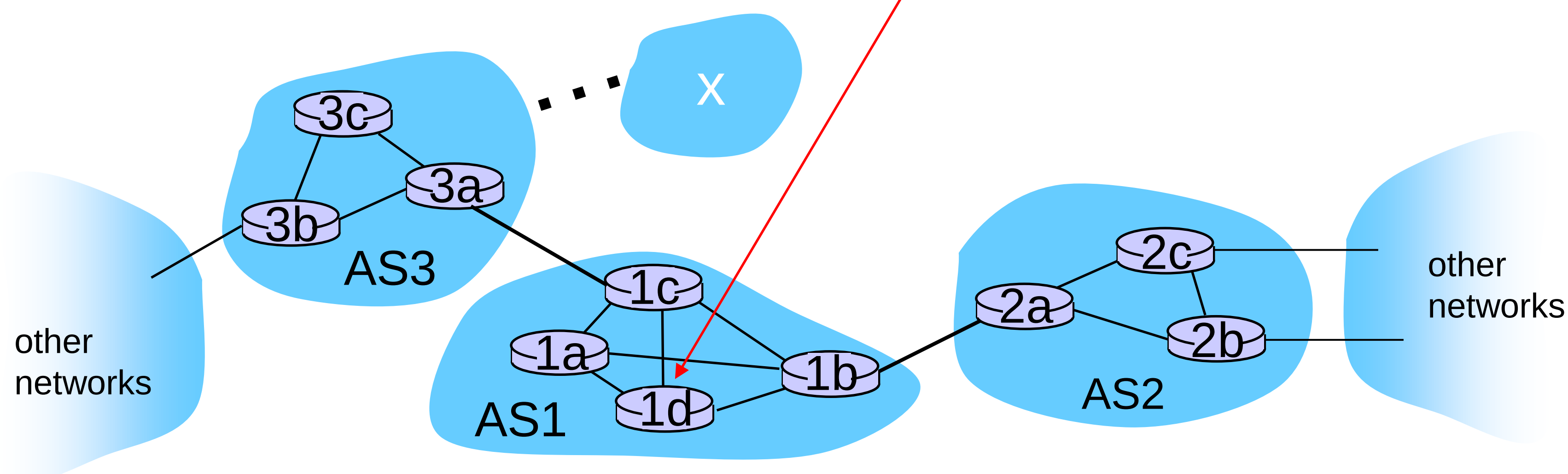
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



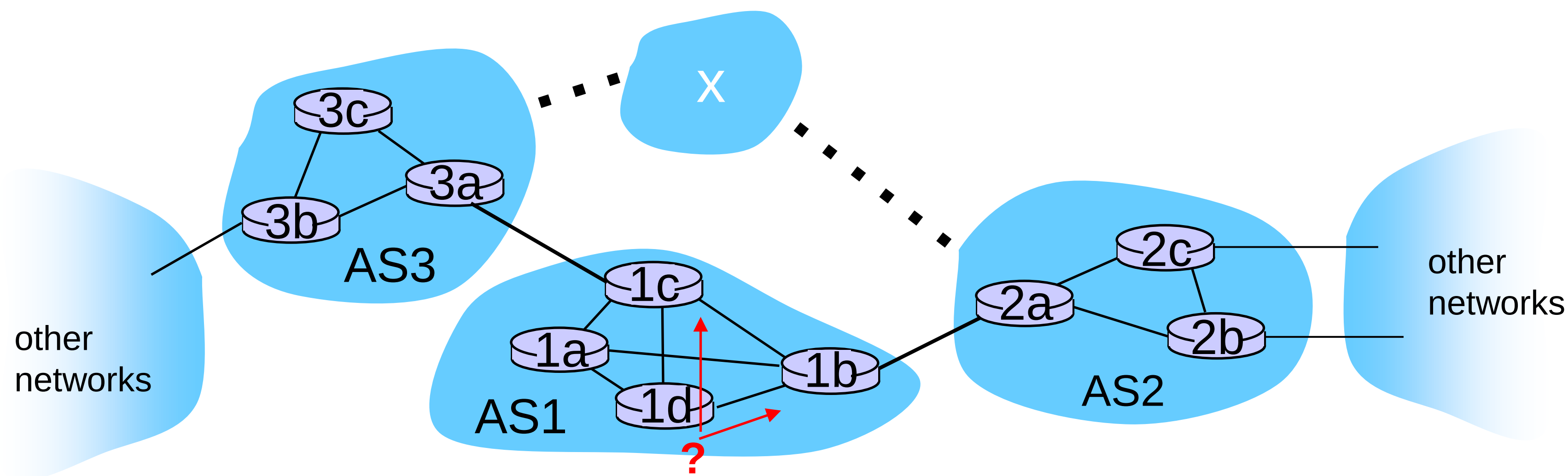
Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
  - installs forwarding table entry *(x,I)*



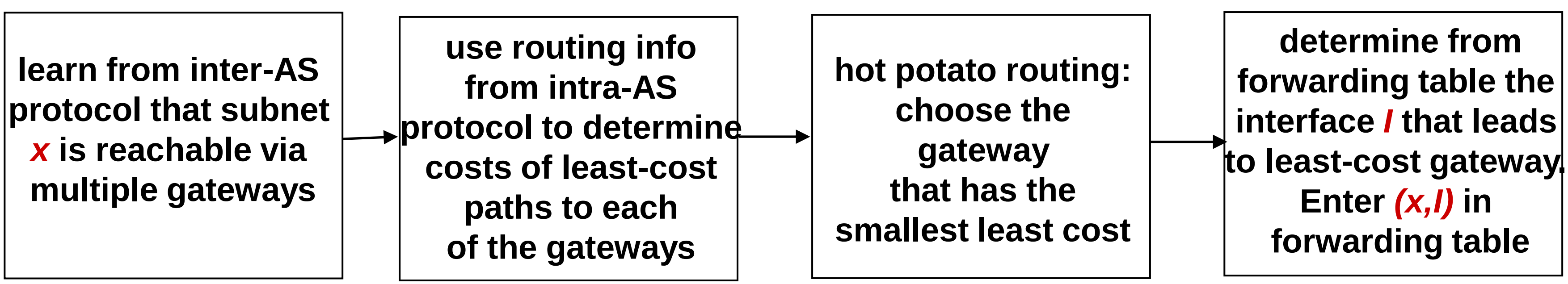
Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x
  - this is also job of inter-AS routing protocol!



Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x
  - this is also job of inter-AS routing protocol!
- *hot potato routing:* send packet towards closest of two routers.
  - Get rid of the packet as soon as possible.



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
- RIP
- OSPF
- BGP

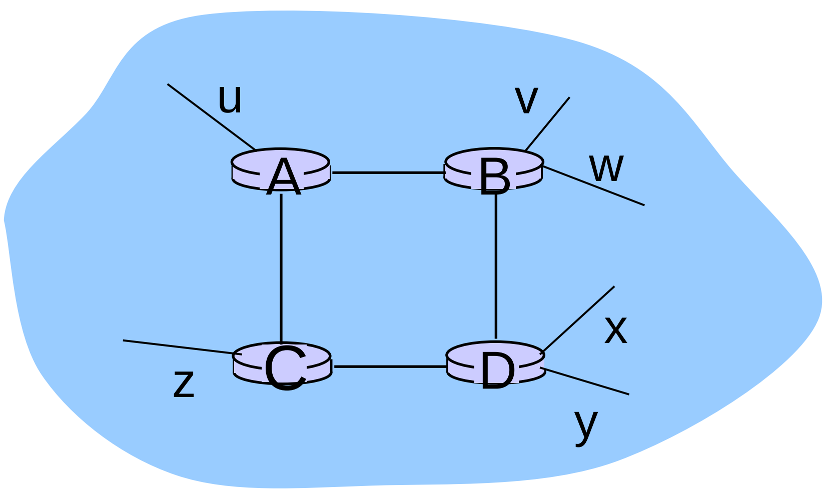
4.7 broadcast and multicast routing

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

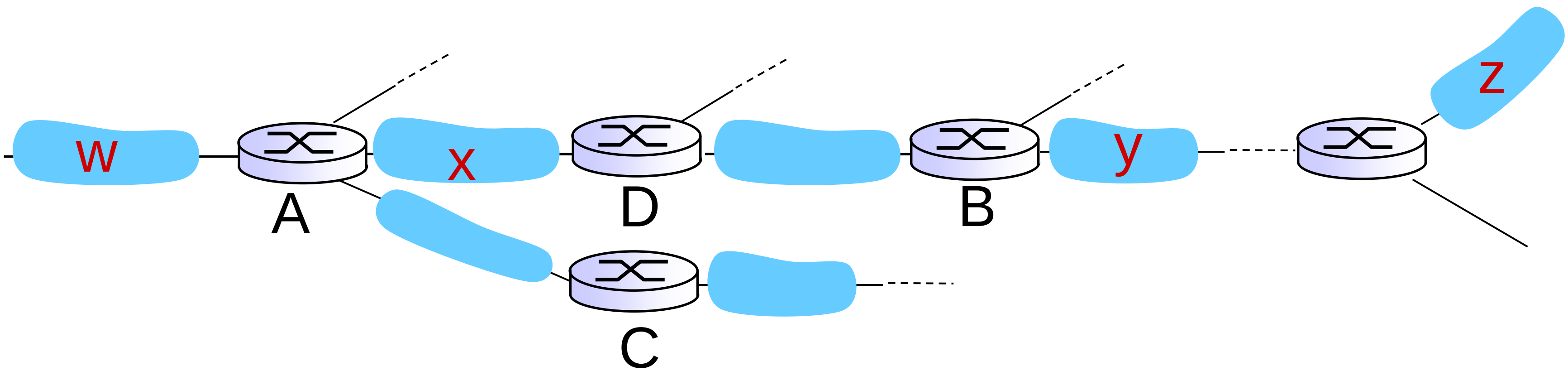
- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - AS should have diameter less than 15 hops
  - DVs exchanged with neighbors every 30 sec in RIP Response Message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets* *(in IP addressing sense)*



from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP: example



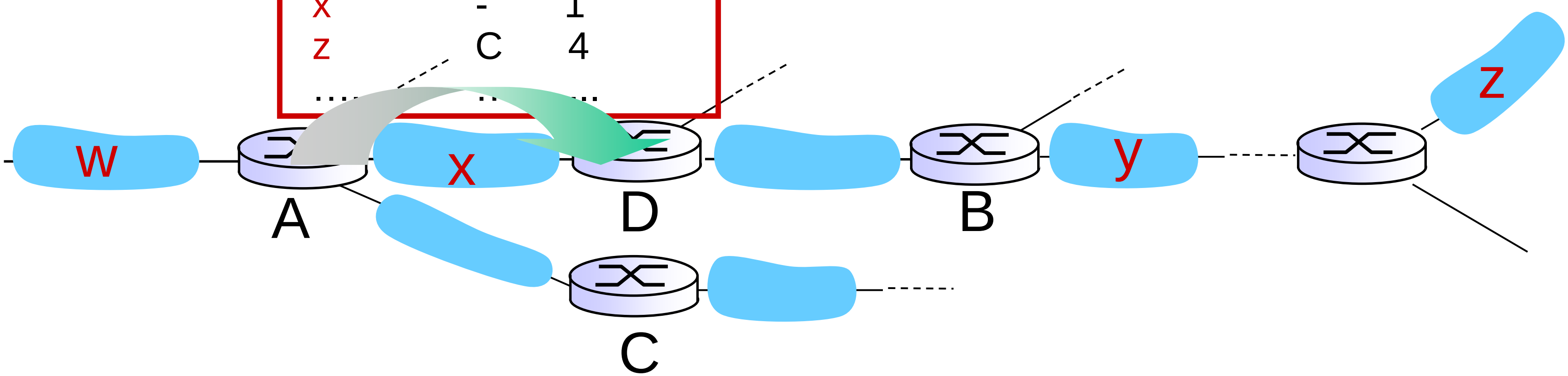
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

RIP: example

A-to-D advertisement

dest	next	hops
w	-	1
x	-	1
z	C	4
….	…	...



routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	....

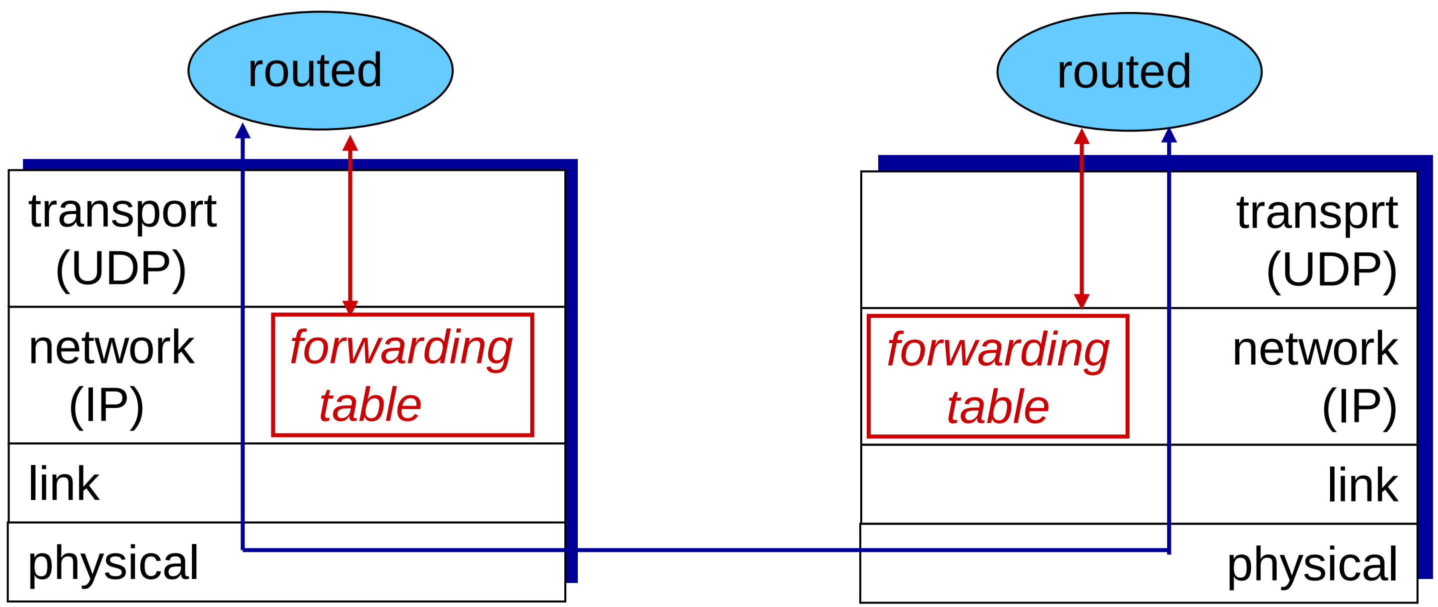
RIP: link failure, recovery

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)

RIP table processing

- RIP routing tables managed by *application-level* process called route-d (daemon) in UNIX
- advertisements sent in UDP packets, port 520, periodically repeated



OSPF (Open Shortest Path First)

- "open": publicly available
  - rfc 2328
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - Shortest-path tree to all subnets locally computed by router using Dijkstra's algorithm
  - How to set link weight is up to the administrator

OSPF (Open Shortest Path First)

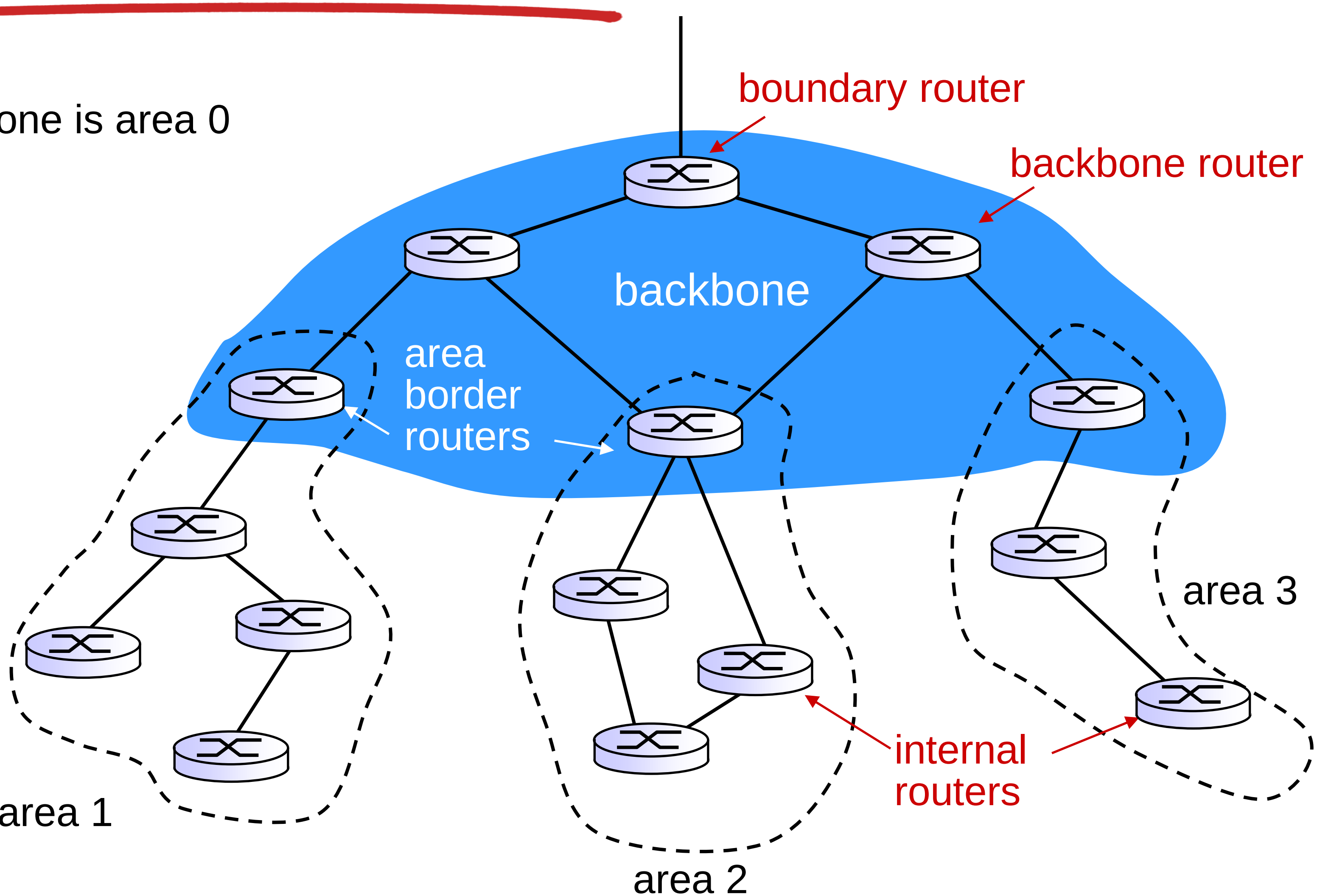
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
  - Upper layer protocol 89
- A router broadcast link-state information when
  - A link's state changes
  - Periodically (e.g., 30 min)
- Check links with HELLO messages
- *IS-IS routing* protocol: nearly identical to OSPF

OSPF “advanced” features (not in RIP)

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
  - Append MD5 hash with a shared secret key
- 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 ToS; high for real time ToS)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Hierarchical OSPF

Backbone is area 0



Hierarchical OSPF

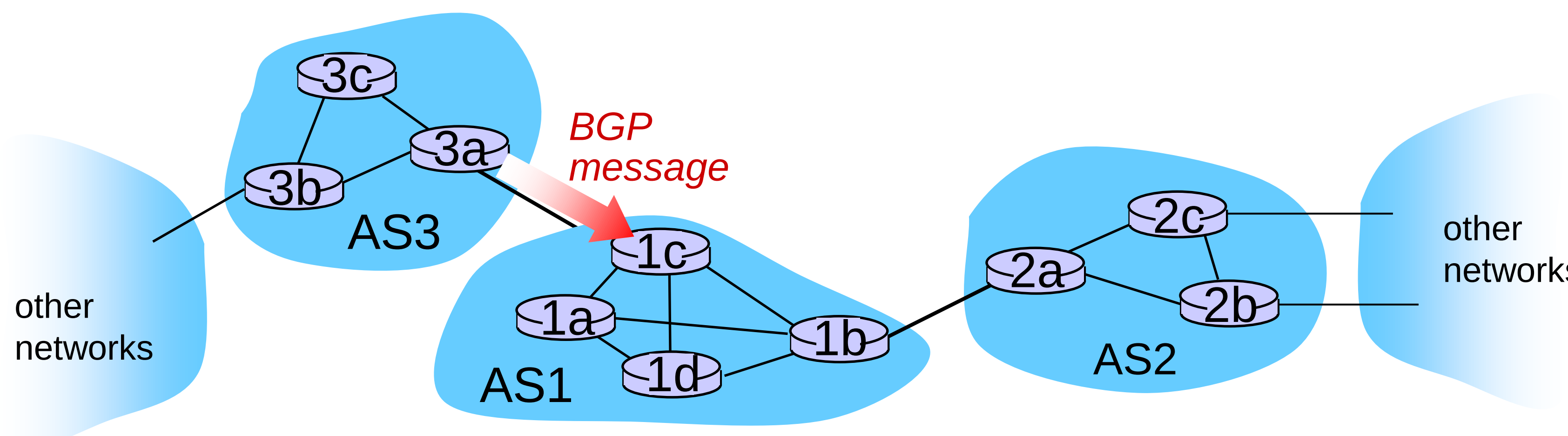
- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.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

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - obtain subnet reachability information from neighboring AS's: eBGP
  - propagate reachability information to all AS-internal routers: iBGP
  - determine "good" routes to other networks based on reachability information and policy.
- Basically, allows subnet to advertise its existence to rest of Internet: *"I am here"*

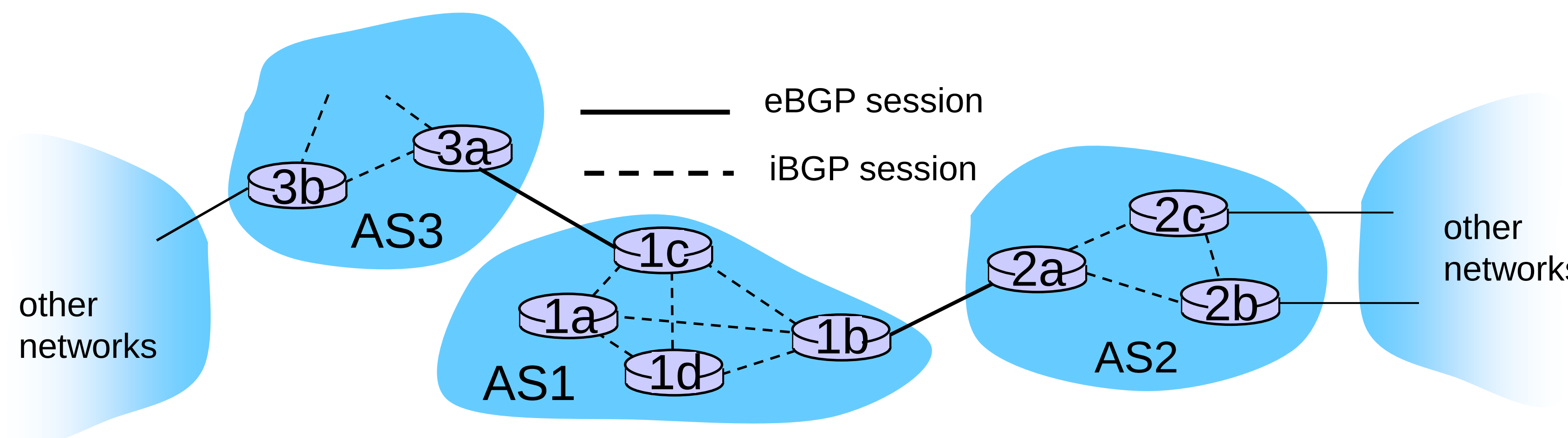
BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections, port 179
- when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP sessions to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



BGP basics: distributing path information

- In BGP, destinations are CIDR prefixes
- Example:
  - AS2 has four subnets: 138.16.64/24, 138.16.65/24, 138.16.66/24, and 138.16.67/24
  - The BGP router in AS2 will aggregate them into a single prefix 138.16.64/22 to advertise to AS1.
- An other example:
  - AS2 has only the first three subnets, the fourth subnet is in AS3
  - AS3 advertise 138.16.67/24
  - AS2 still advertise 138.16.64/22
  - Longest-prefix match in BGP routing

AS number

- In BGP, an AS has a globally unique autonomous number (ASN)
  - Assigned by ICANN
- Sub AS
  - carries only traffic for which it is a source or destination
  - Not necessarily has an ASN

- ASN lookup: http://asn.cymru.com/

```
[Querying v4.whois.cymru.com]
[v4.whois.cymru.com]
AS      | IP               | AS Name
4538    | 222.195.68.249   | ERX-CERNET-BKB China Education and Research Network Center, CN
```

Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - NEXT-HOP: the IP address of the router interface that begins the AS PATH

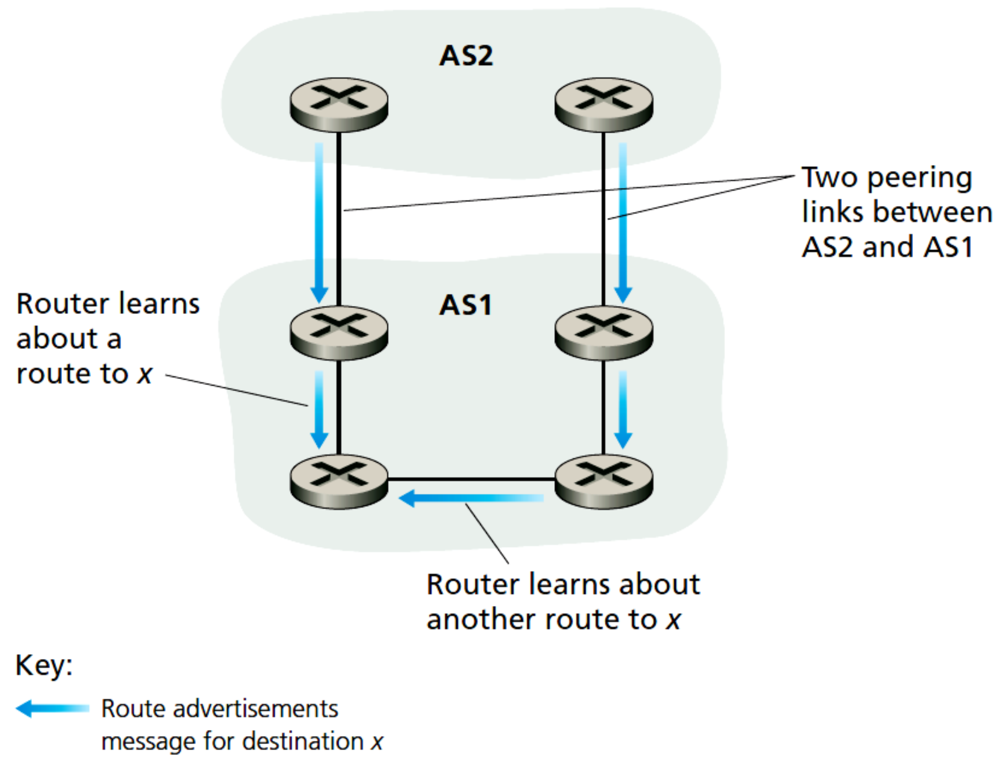
- Example:

138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

201.44.13.125 is the IP address of the first BGP router encountered in AS3

Path attributes and BGP routes

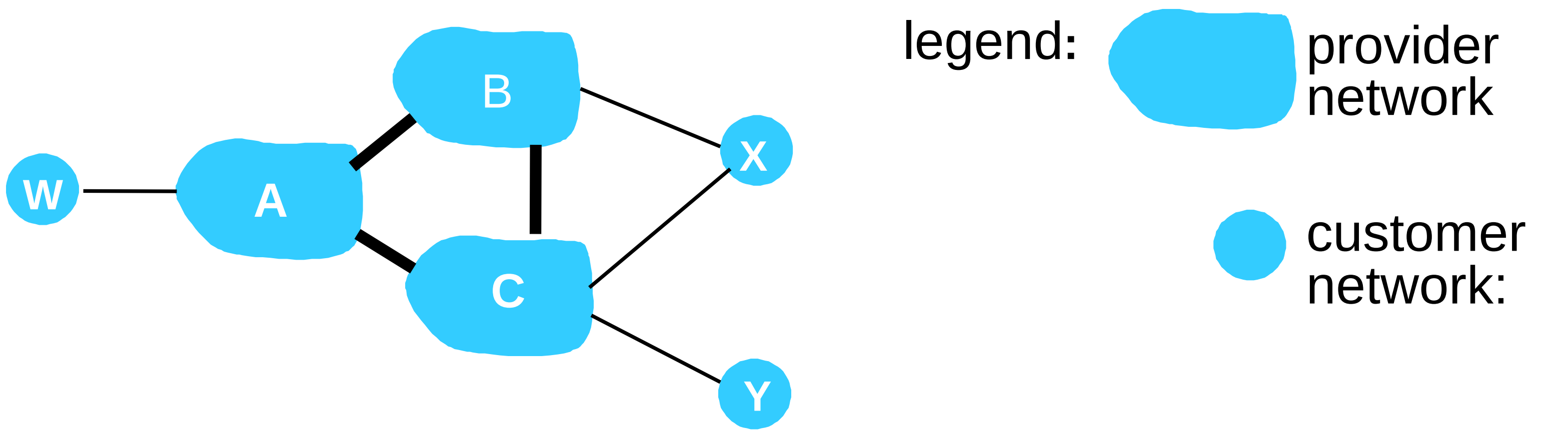
- Two routes to x
  - Same AS path
  - Different NEXT-HOP
- The router can use the intra-AS routing algorithm to perform the hot potato policy



Path attributes and BGP routes

- gateway router receiving route advertisement uses import policy to accept/decline
  - e.g., never route through AS x
  - *policy-based* routing

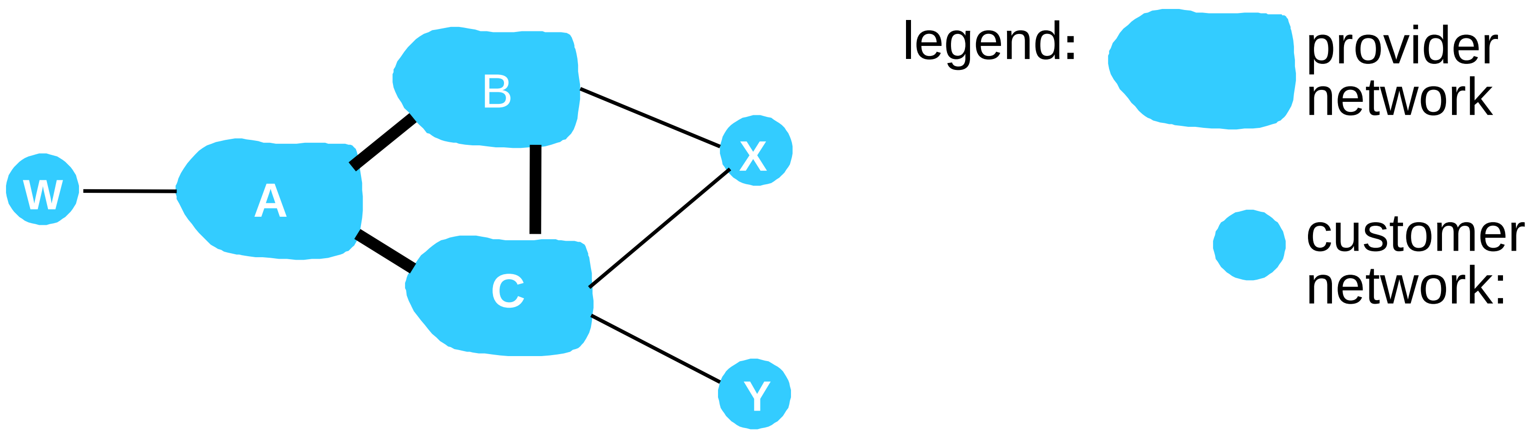
BGP: achieving policy via advertisements



Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

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

BGP: achieving policy via advertisements



Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C:
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
  - C does not learn about CBAw path
- C will route CAw (not using B) to get to w

BGP route selection

- router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria
- These rules are applied *sequentially*

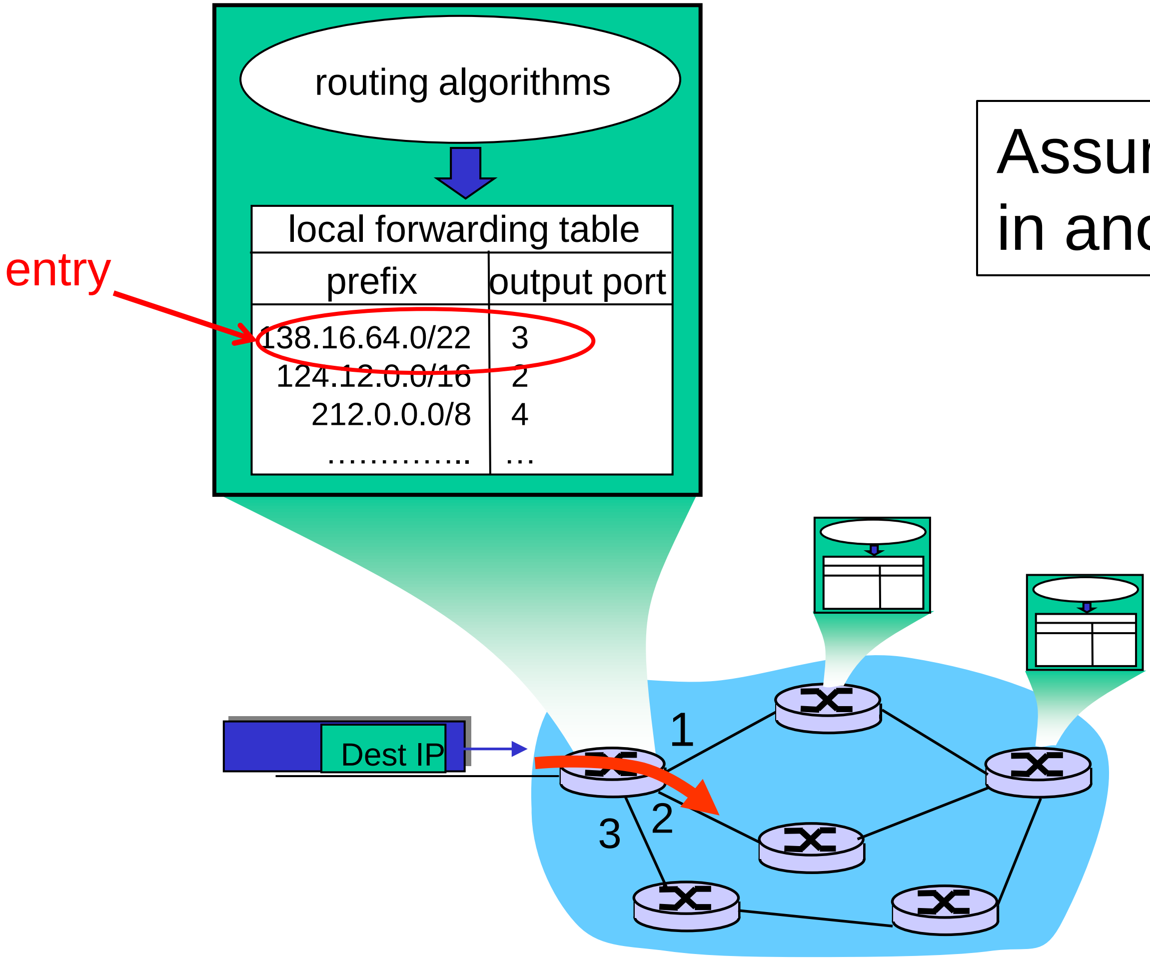
BGP messages

- BGP messages exchanged between peers over TCP connection
- 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

Putting it Altogether: *How Does an Entry Get Into a Router's Forwarding Table?*

- Answer is complicated!
- Ties together hierarchical routing (Section 4.5.3) with BGP (4.6.3) and OSPF (4.6.2).
- Provides nice overview of BGP!

How does entry get in forwarding table?



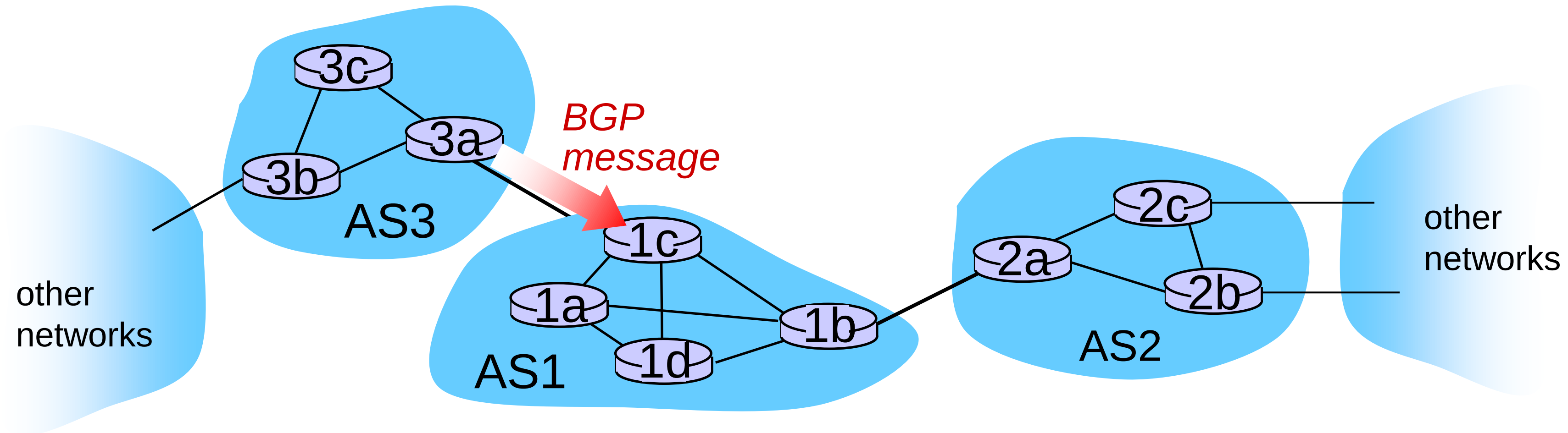
Assume prefix is in another AS.

How does entry get in forwarding table?

High-level overview

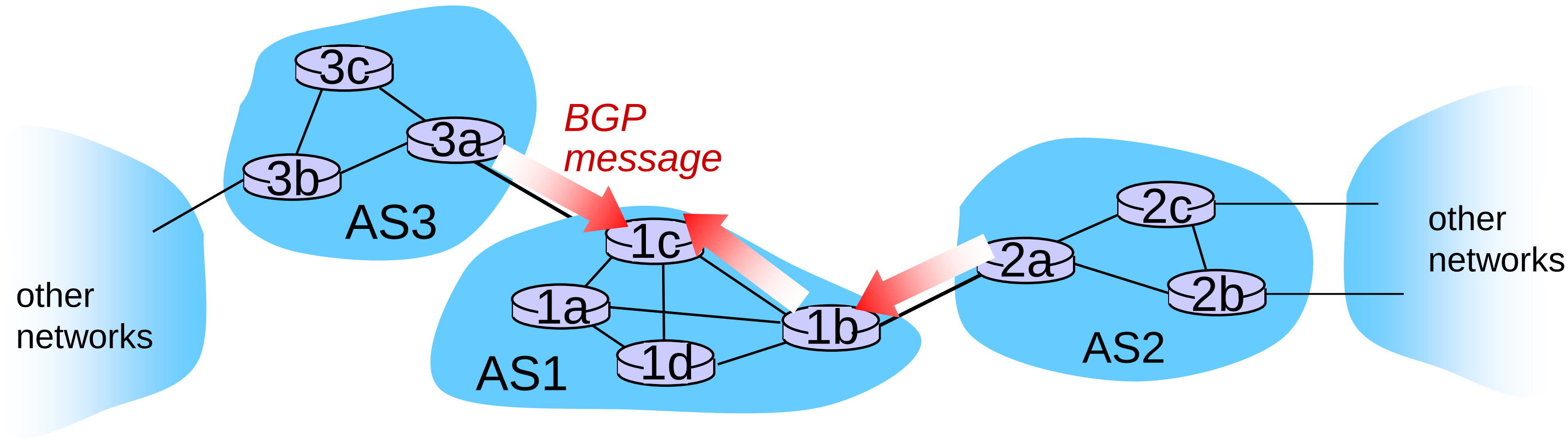
1. Router becomes aware of prefix
2. Router determines output port for prefix
3. Router enters prefix-port in forwarding table

Router becomes aware of prefix



- BGP message contains "routes"
- "route" is a prefix and attributes: AS-PATH, NEXT-HOP,…
- Example: route:
  - Prefix:138.16.64.0/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

Router may receive multiple routes



- Router may receive multiple routes for <u>same</u> prefix
- Has to select one route

Select best BGP route to prefix

- Router selects route based on shortest AS-PATH

- Example:

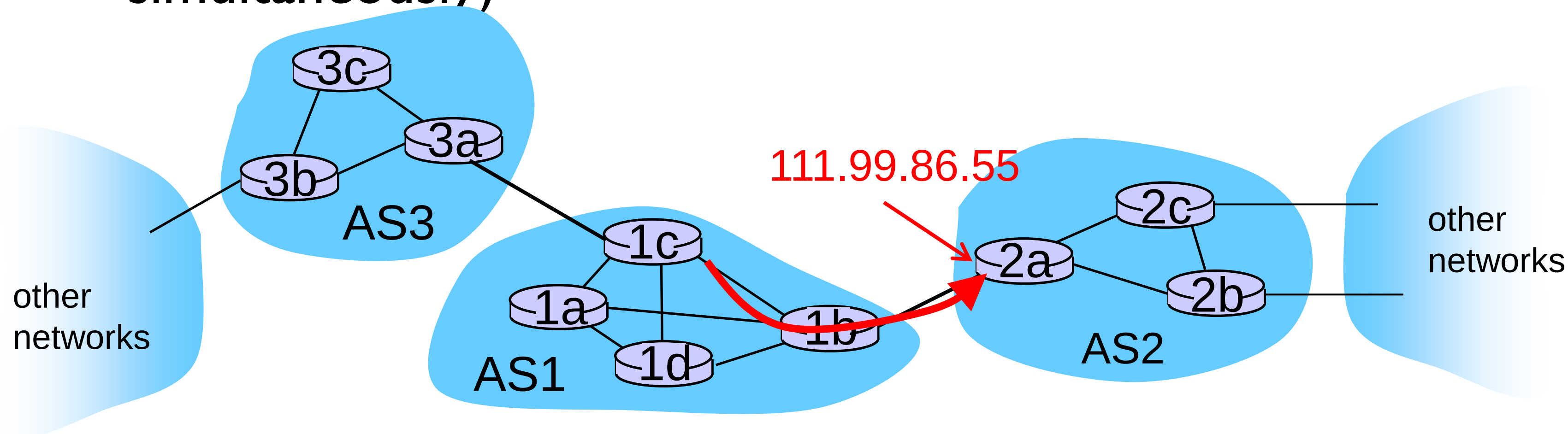
  select

  - AS2 AS17 to 138.16.64.0/22
  - AS3 AS131 AS201 to 138.16.64.0/22

- What if there is a tie? We'll come back to that!

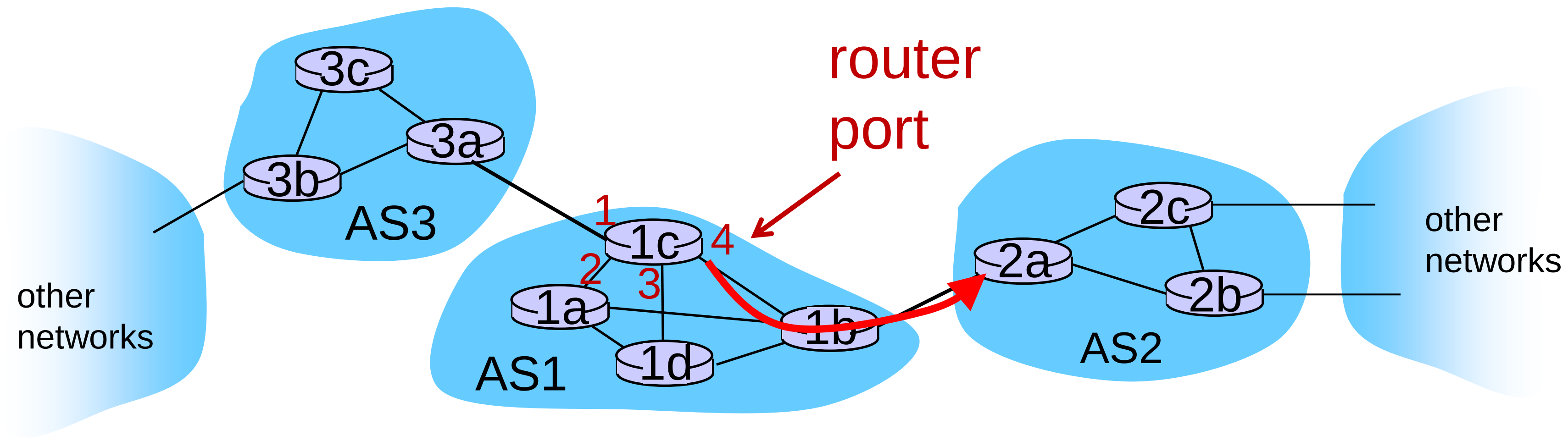
Find best intra-route to BGP route

- Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- Example:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- Router uses OSPF to find shortest path from 1c to 111.99.86.55 (this address belongs to AS1 and AS2 simultaneously)



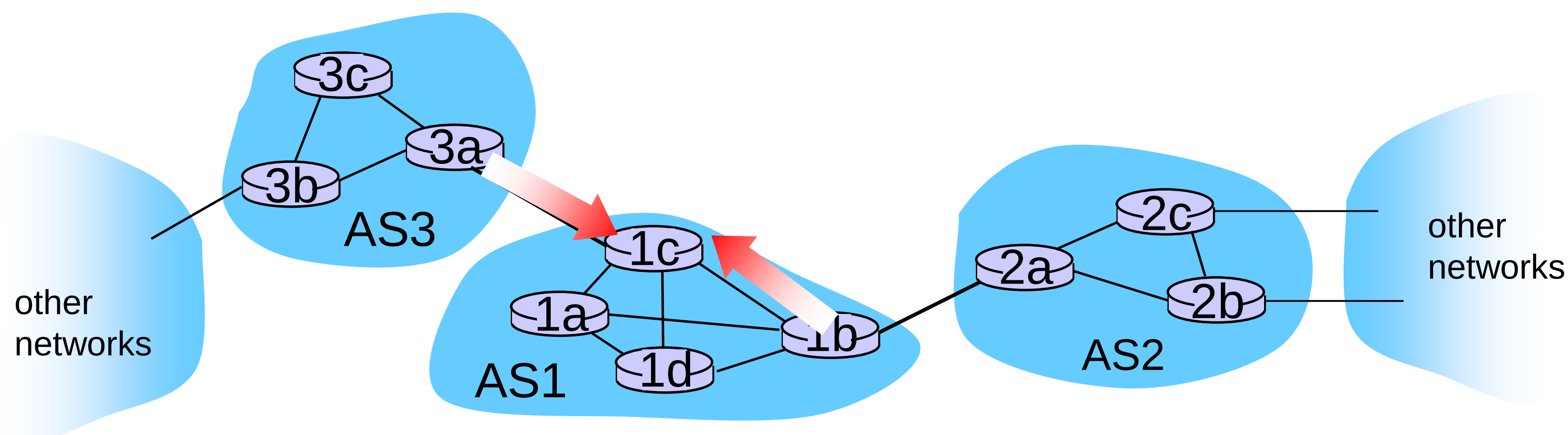
Router identifies port for route

- Identifies port along the OSPF shortest path
- Adds prefix-port entry to its forwarding table:
  - (138.16.64/22 , port 4)



Hot Potato Routing

- Suppose there two or more best inter-routes.
- Then choose route with closest NEXT-HOP
  - Use OSPF to determine which gateway is closest
  - Q: From 1c, chose AS3 AS131 or AS2 AS17?
  - A: route AS3 AS131 since it is closer

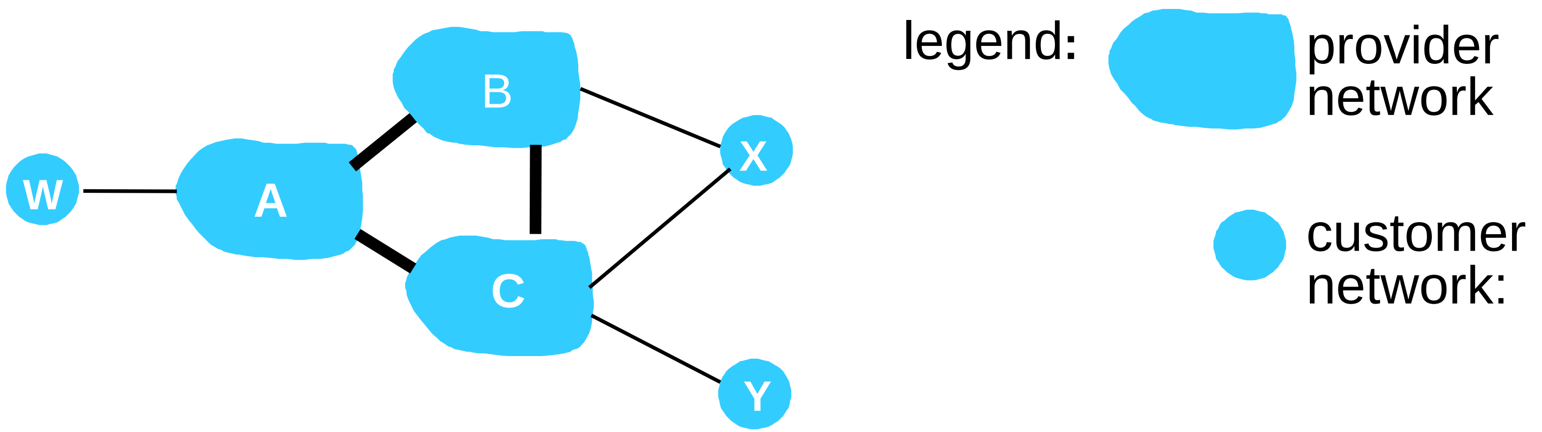


How does entry get in forwarding table?

Summary

1. Router becomes aware of prefix
   - via BGP route advertisements from other routers
2. Determine router output port for prefix
   - Use BGP route selection to find best inter-AS route
   - Use OSPF to find best intra-AS route leading to best inter-AS route
   - Router identifies router port for that best route
3. Enter prefix-port entry in forwarding table

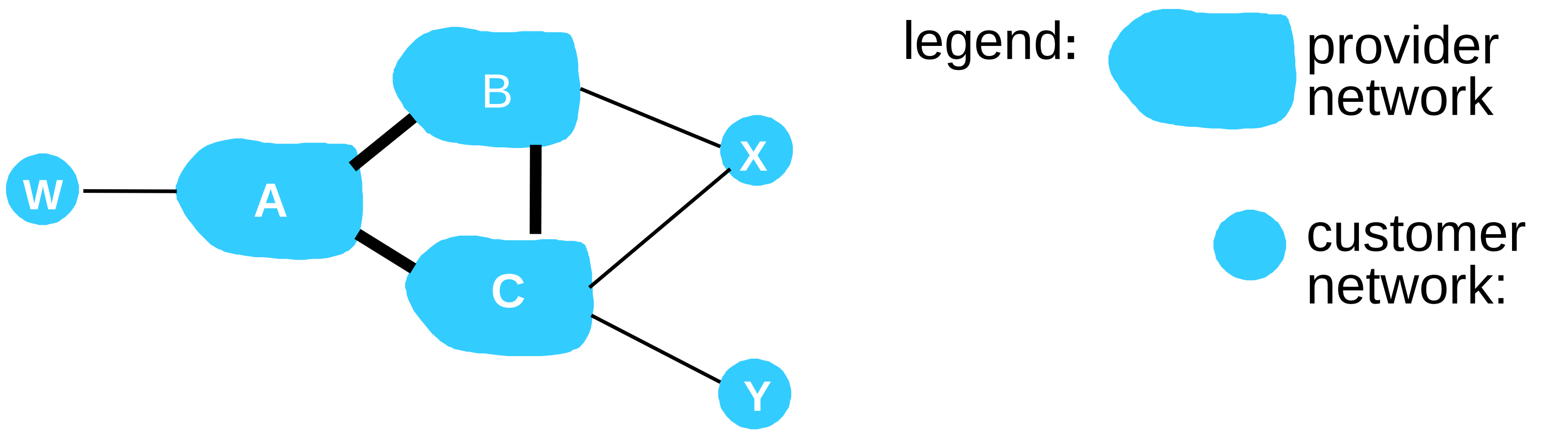
BGP routing policy



X and Y are sub networks, carrying traffics only to or from its own subnets.

- 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 route from B via X to C
  - .. so X will not advertise to B a route to C
  - B does not know the path BXC

BGP routing policy (2)



- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - 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!

Why different Intra-, 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

IP Hijacking

- How
  - An AS announces that it originates a prefix that it does not actually originate.
  - An AS announces a more specific prefix than what may

Chinese ISP hijacks the Internet

Posted by Andree Toonk – April 8, 2010 – Hijack – 25 Comments

This morning many BGPmon.net users received an alert regarding a possible prefix hijack by a Chinese network. AS23724 is one of the Data Centers operated by China Telecom, China's largest ISP. Normally AS23724 CHINANET-IDC-BJ-AP IDC, China Telecommunications Corporation only originates about 40 prefixes, however today for about 15 minutes they originated about ~37,000 unique prefixes that are not assigned to them. This is what we typically call a prefix hijack.

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

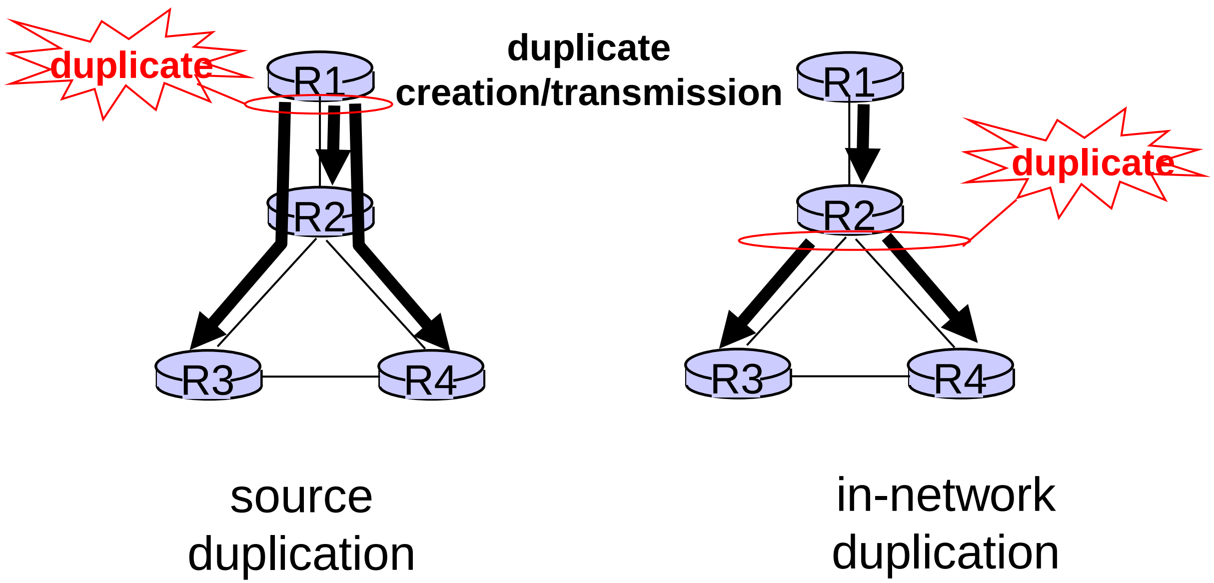
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Broadcast routing

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

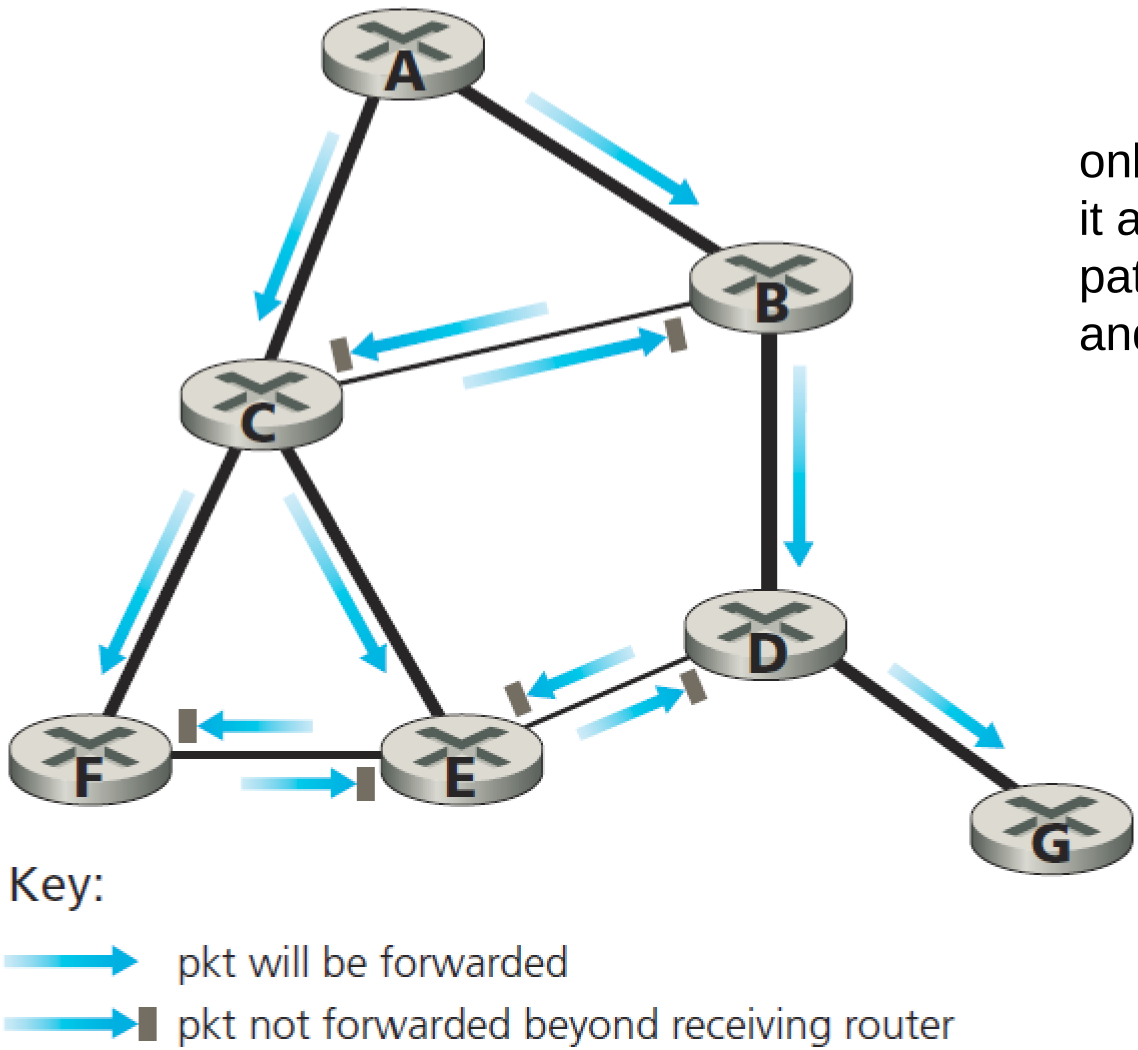


- source duplication: how does source determine recipient addresses?

In-network duplication

- *Uncontrolled 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 broadacsted (src. addr. + seq. num)
  - 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

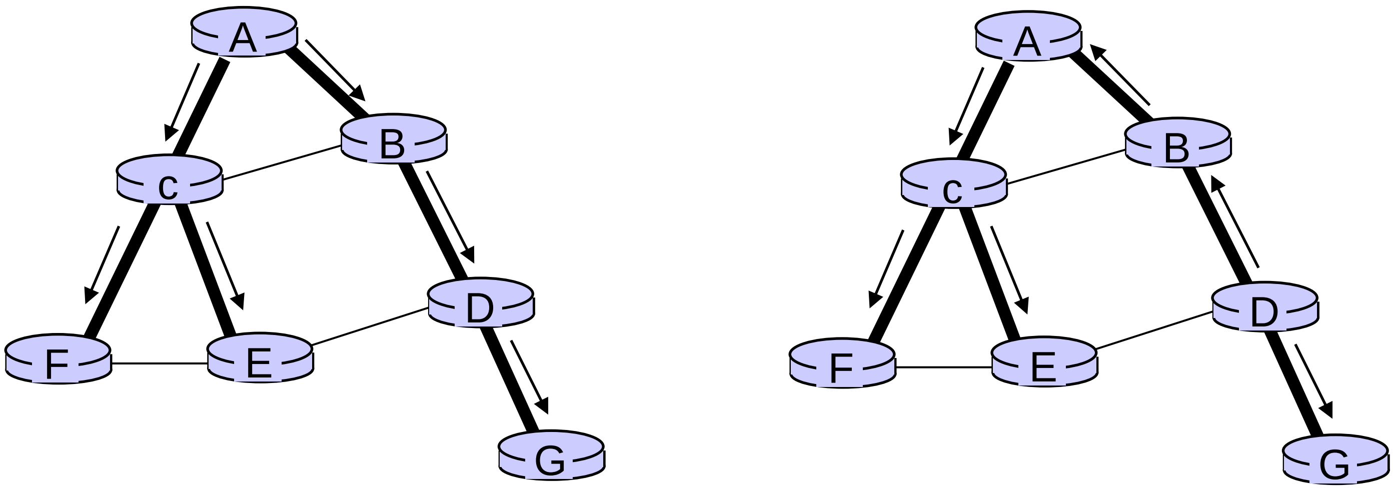
Reverse path forwarding



only forward packet if it arrived on shortest path between node and source

Spanning tree

- first construct a spanning tree
- nodes then forward/make copies only along spanning tree



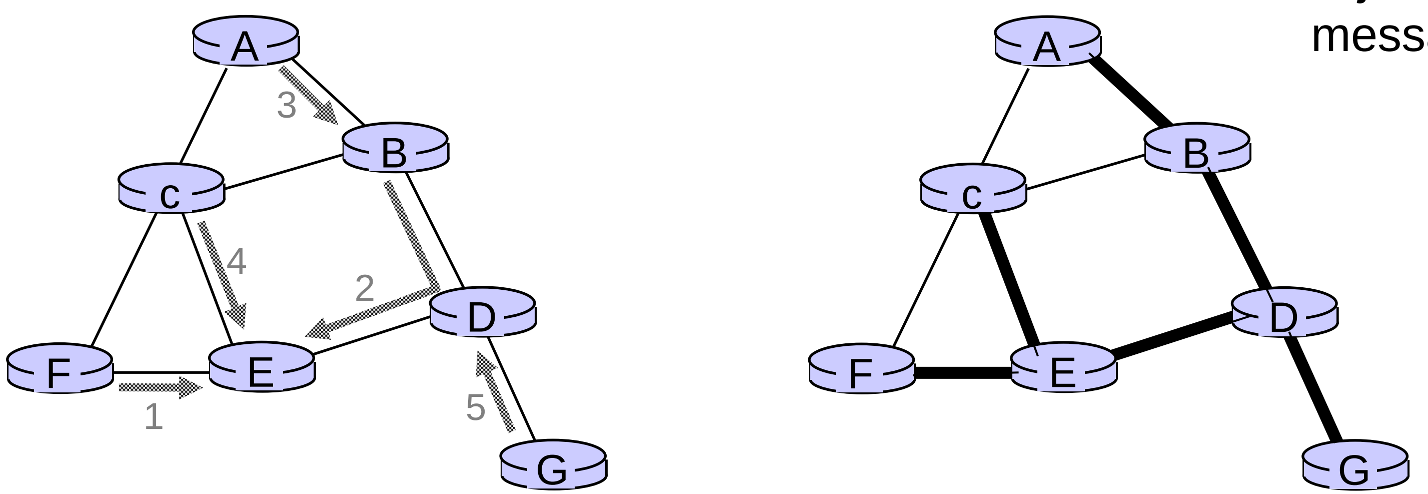
(a) broadcast initiated at A

(b) broadcast initiated at D

Spanning tree: creation

- Define a center node
- each node sends unicast join message to center node
  - message forwarded until it arrives at a node already belonging to spanning tree

D joins by B's message

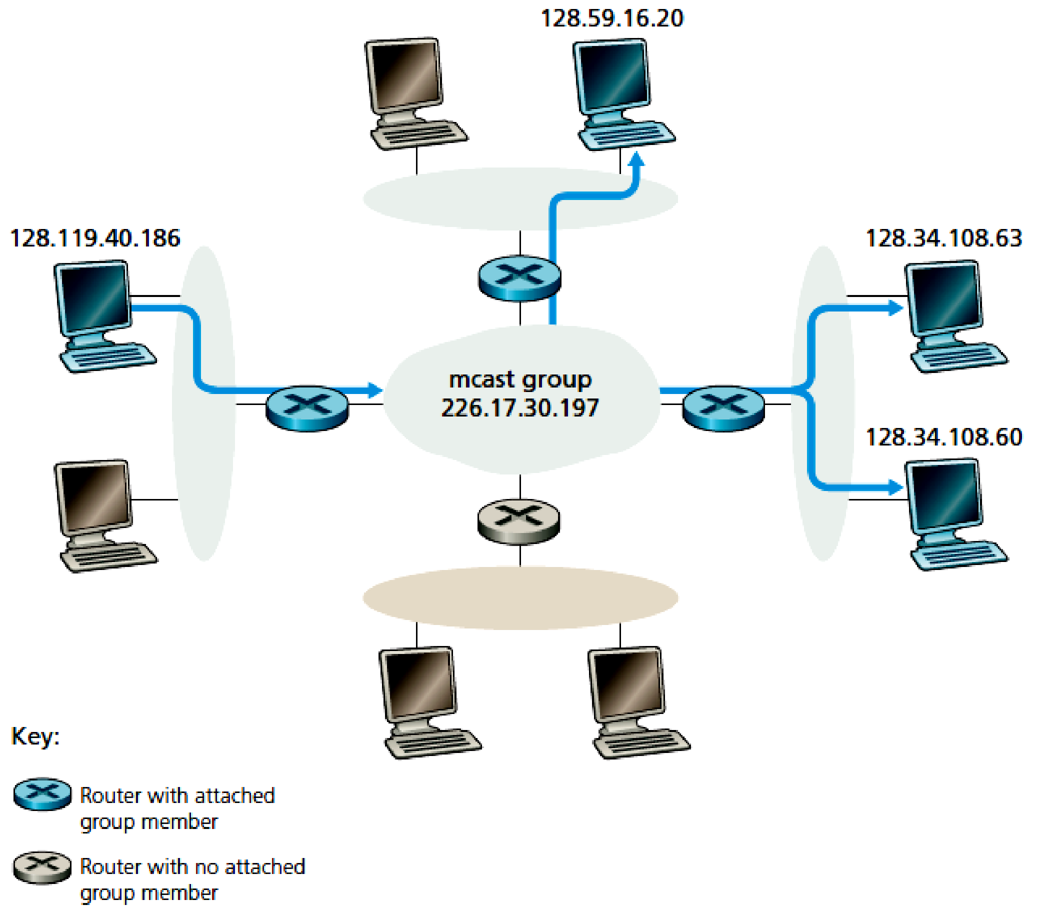


(a) stepwise construction of spanning tree (center: E)

(b) constructed spanning tree

Multicast Service

- Deliver packets to only a *subset of* network nodes.
- Two problems:
  - How to identify the receivers?
  - How to address them?
- Solution:
  - Use a class D multicast IP address to represent a group of receivers



  - 224.0.0.0 to 239.255.255.255

IGMP

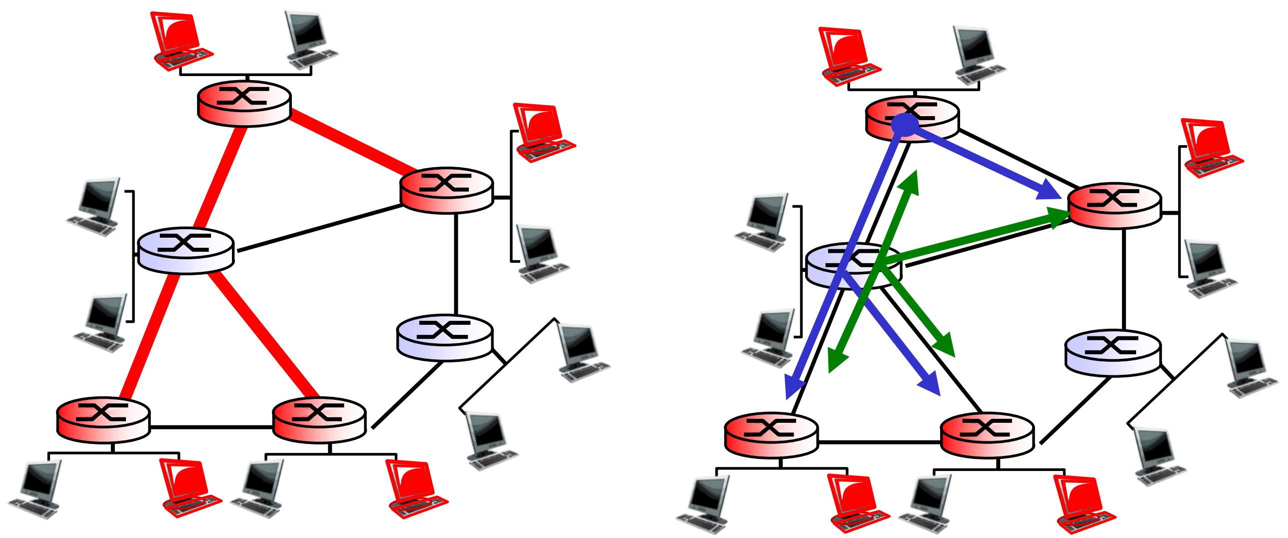
- How to manage a group of receivers?
- IGMPv3 protocol (rfc 3376)
  - Between a host and its directly attached router
  - Inform the router an application running on the host wants to join a specific multicast group
- Three message types:
  - membership_query: router to host, determine all multicast groups joined by the hosts
  - membership_report: host to router, respond the query
  - leave_group: optional. When not used, resort to soft state protocol mechanism
    - State is removed via a timeout event instead of explicitly refreshed.

Multicast routing: problem statement

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

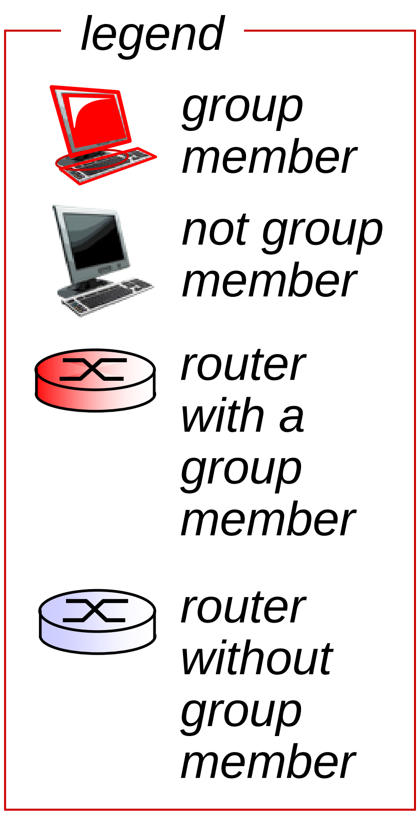
tree: not all paths between routers used

- *shared-tree:* same tree used by all group members
- *source-based:* different tree from each sender to rcvrs



shared tree

source-based trees

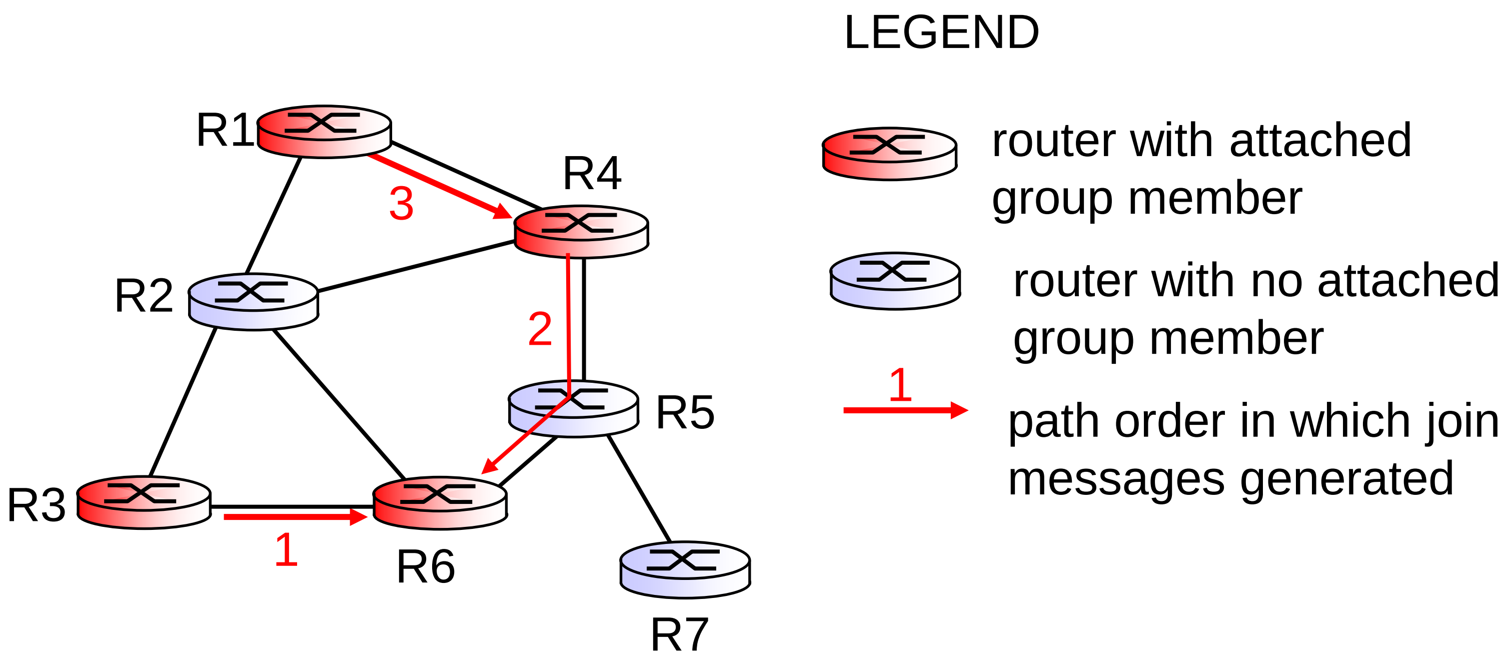


Constructing group-shared tree

- *Center-based approach*
- one router identified as "*center*" of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Center-based trees: example

suppose R6 chosen as center:



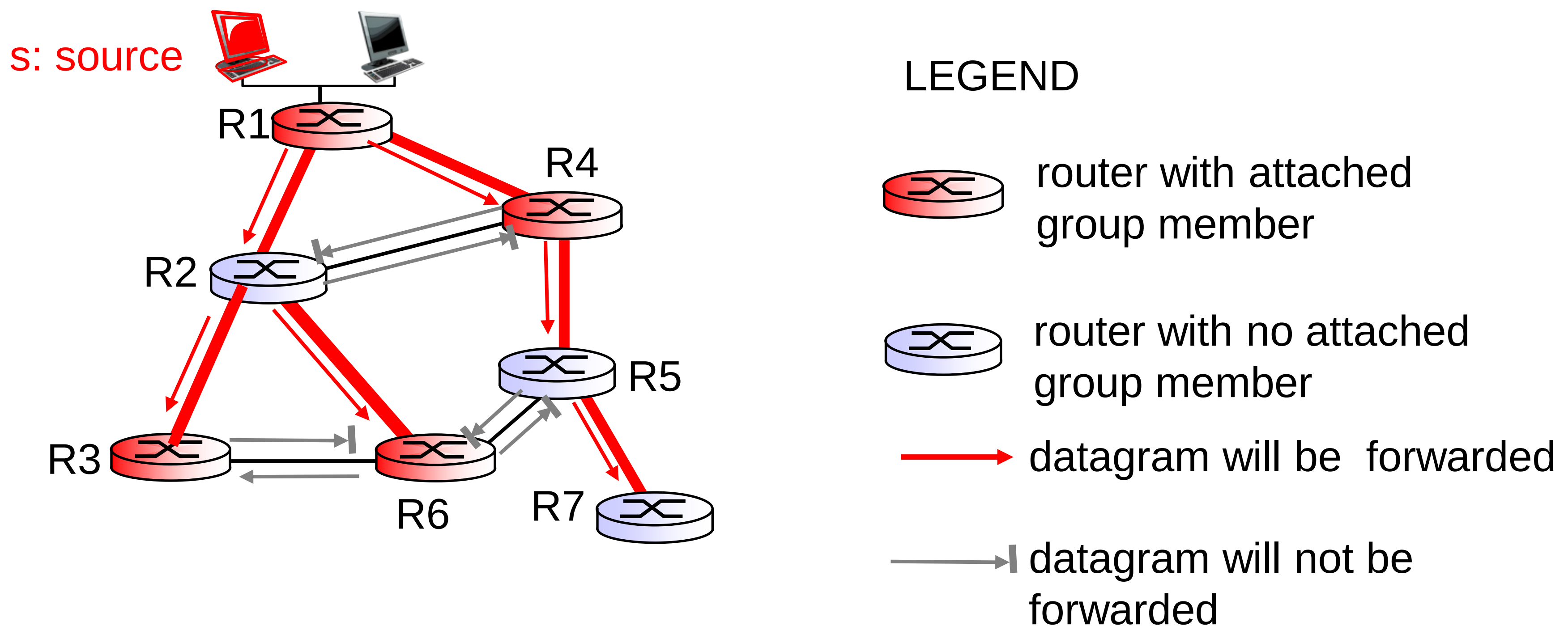
Constructing source-based tree

RFP (reverse path forwarding) algorithm

- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

> ***if*** (mcast datagram received on incoming link on shortest path back to center)
> ***then*** flood datagram onto all outgoing links
> ***else*** ignore datagram

Reverse path forwarding: example

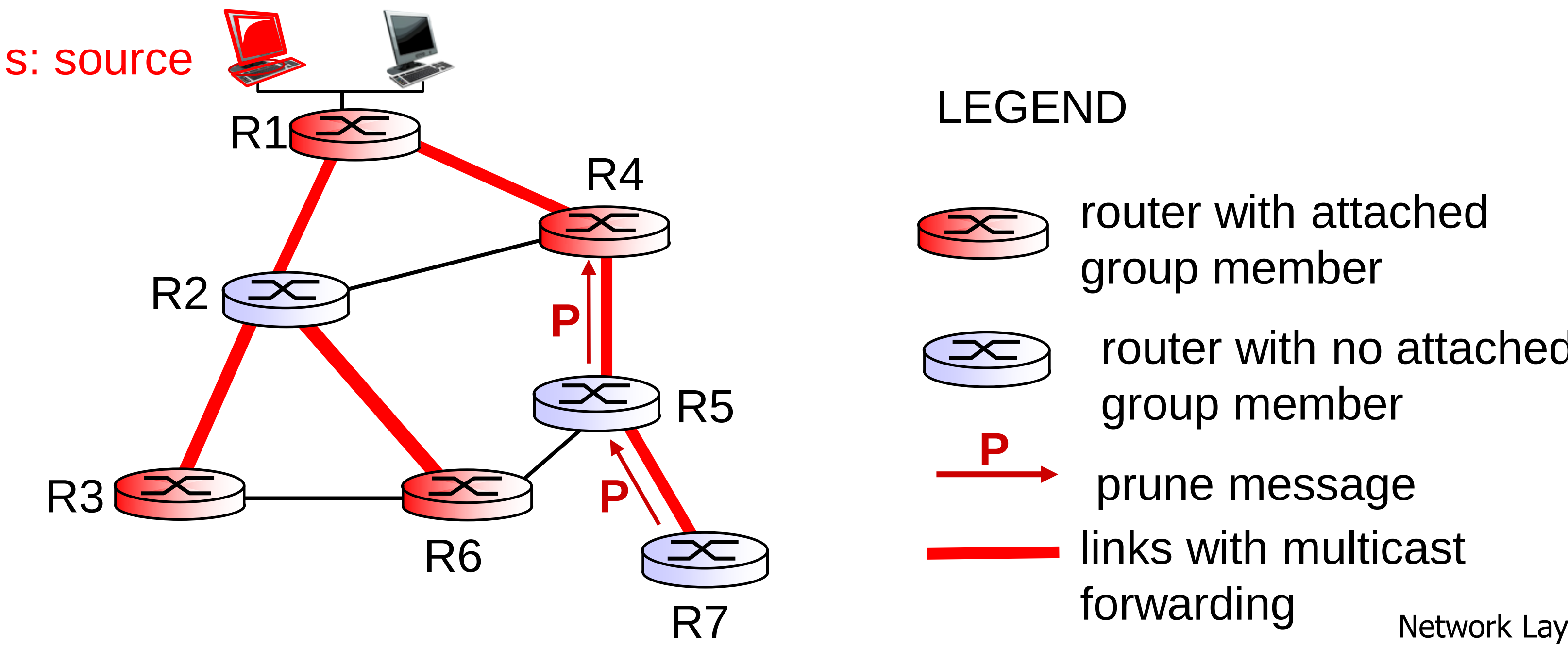


- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse path forwarding: pruning

- forwarding tree contains subtrees with no mcast group members (e.g. R7)
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members
- Pruning
  - A multicast router that receives multicast packets and has no attached hosts joined to that group will send a prune message to its upstream router.
  - If a router receives prune messages from each of its downstream routers, then it can forward a prune message upstream.

RPF Pruning: Example



Internet Multicasting Routing Protocols

- Distance-Vector Multicast Routing Protocol (DVMRP) (rfc 1075)
  - source-based trees with reverse path forwarding and pruning
- Protocol-Independent Multicast (PIM) routing protocol (rfc 3973)
  - Dese mode, similar to DVMRP
  - Sparse mode, center-based approach to set up multicast tree
- Not widely deployed on the Internet
  - Political and economical reasons

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