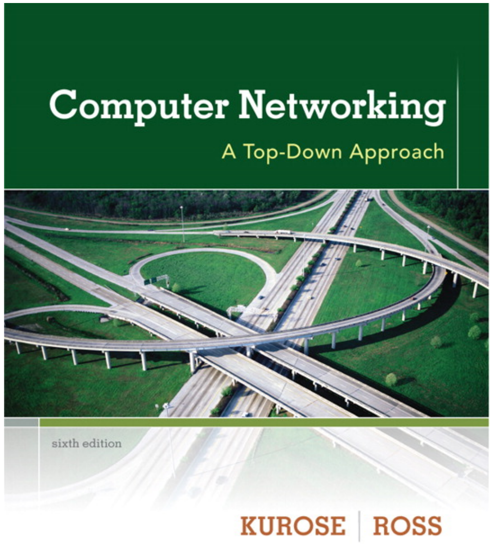
Unit 6
Network Layer

Bibliography: Kurose 12, Chapter 4



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

Slides adapted from JFK/KWR slides

Unit 6: network layer

chapter goals:

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - Virtual Circuit and Datagram Networks
  - IP: Internet Protocol
  - Routing (path selection)

Unit 6: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IP fragmentation (Labs)
- IPv4 addressing
- DHCP (Labs)
- ICMP (Labs)

4.4 routing algorithms

- distance vector
- link state
- hierarchical routing

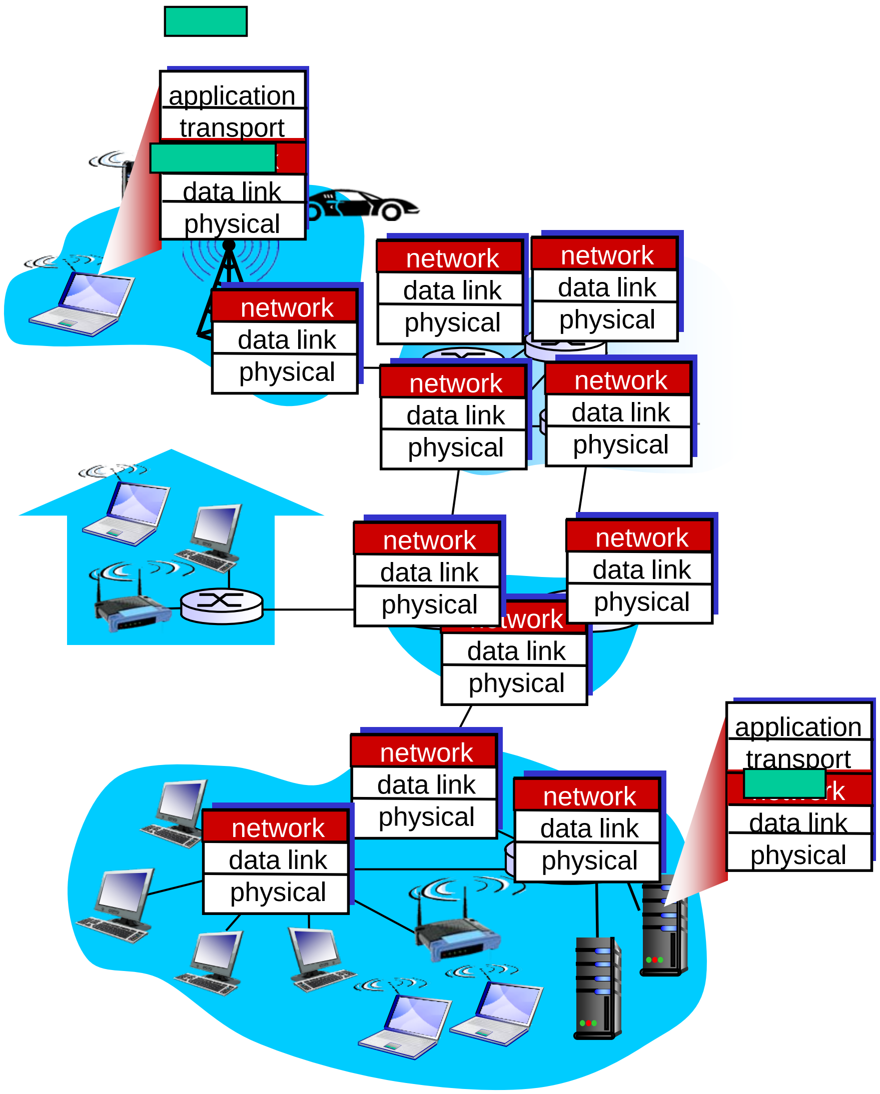
4.5 routing in the Internet

- RIP
- OSPF
- BGP

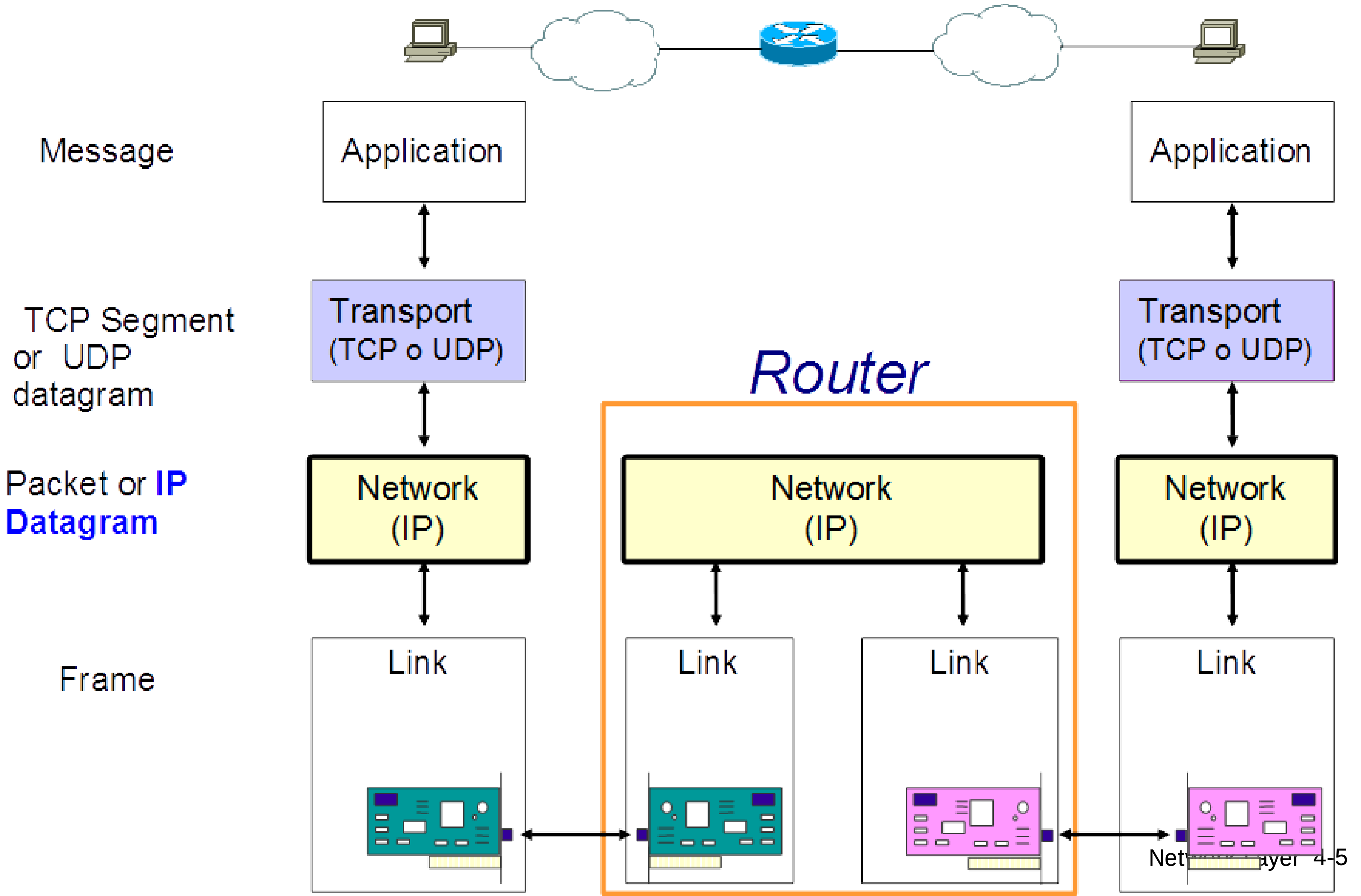
4.6 IPv6

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



Communication and protocol stack



Message

TCP Segment or UDP datagram

Packet or **IP Datagram**

Frame

Application

Transport (TCP o UDP)

Network (IP)

Link

Router

Network (IP)

Link

Link

Application

Transport (TCP o UDP)

Network (IP)

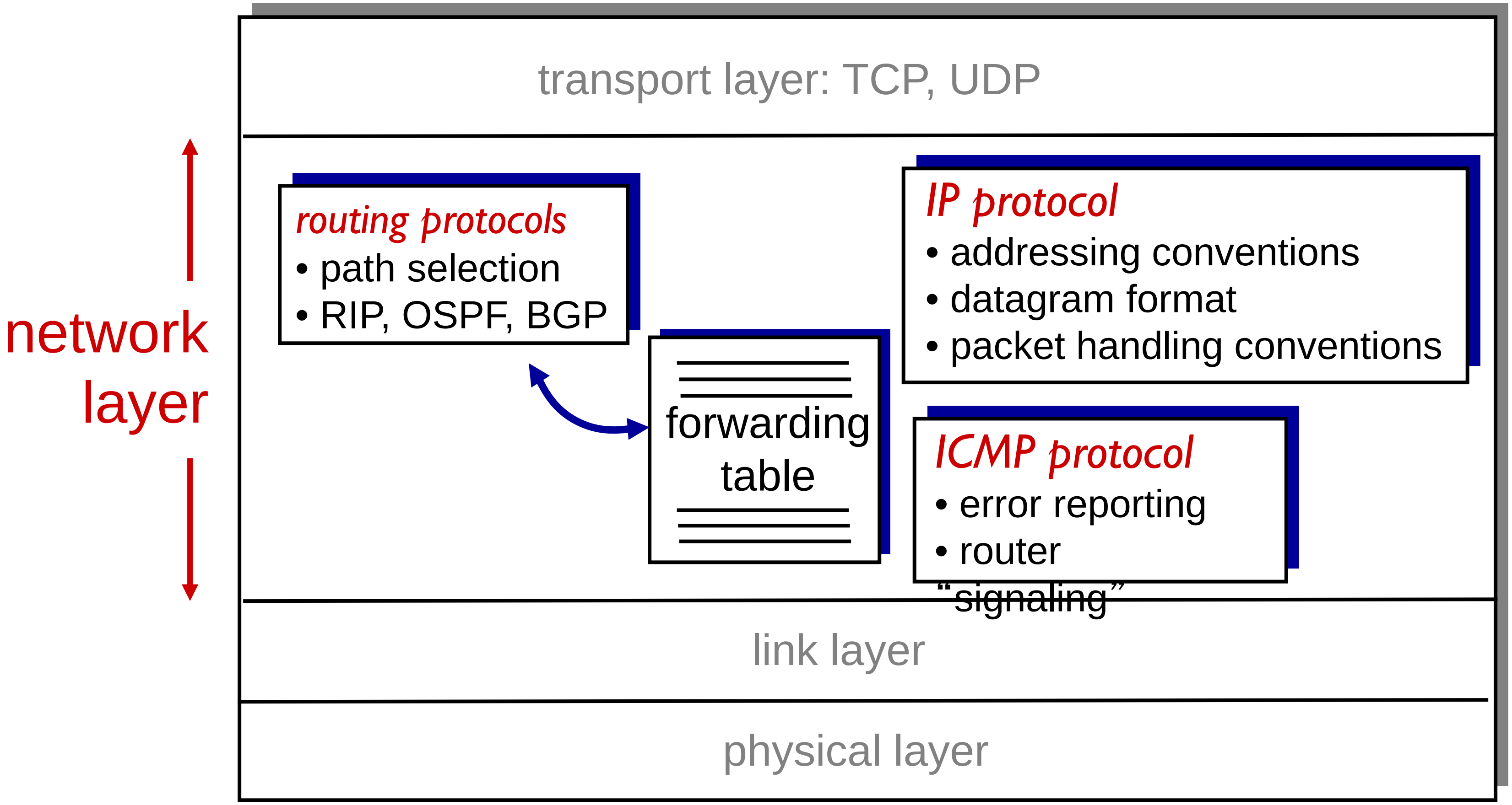
Link

Network Layer

- To carry the packages through the network you need:
  - Identify with addresses of the devices that intervene in the communication (IP addresses)
  - Choose a route in the network that allows you to reach the destination (routing)
- IP is in charge of both problems

The Internet network layer

host, router network layer functions:



transport layer: TCP, UDP

network layer

routing protocols
- path selection
- RIP, OSPF, BGP

forwarding table

IP protocol
- addressing conventions
- datagram format
- packet handling conventions

ICMP protocol
- error reporting
- router "signaling"

link layer

physical layer

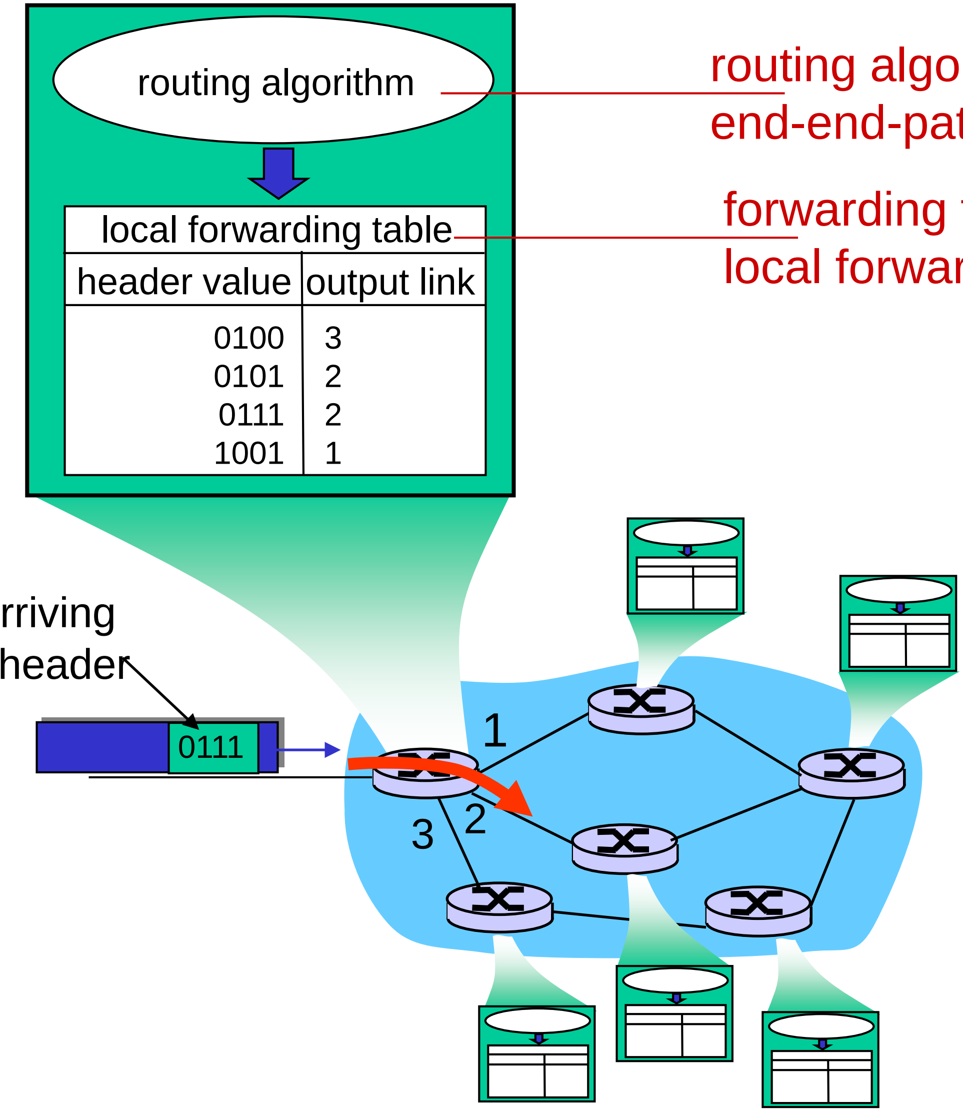
Two key network-layer functions

- *forwarding:*
  - move packets from router' s input to appropriate router output
  - router local action
- *routing:*
  - determine route taken by packets from source to destination
    - *routing algorithms*
  - network wide process

analogy:

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

Interplay between routing and forwarding



local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

value in arriving packet’s header

Network service model

- The network service model defines the characteristics of end-to-end transport of packets between sending and receiving end systems.
- Possible services that the network layer could provide:
  - for individual datagrams:
    - guaranteed delivery
    - guaranteed delivery with less than a fixed delay
  - for a flow of datagrams:
    - in-order datagram delivery
    - guaranteed minimum bandwidth to flow
    - restrictions on changes in inter-packet spacing

Network service model

- The Internet's network layer provides a single service, known as **best-effort service**.
  - timing between packets is not guaranteed to be preserved,
  - packets are not guaranteed to be received in the order in which they were sent,
  - nor is the eventual delivery of transmitted packets guaranteed.
  - Given this definition, a network that delivered *no* packets to the destination would satisfy the definition of best-effort delivery service.

Network layer service models:

- Different network architectures offer different services

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
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

Unit 6: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol
- datagram format
- IP fragmentation (Labs)
- IPv4 addressing
- DHCP (Labs)
- ICMP (Labs)

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

4.5 routing in the Internet
- RIP
- OSPF
- BGP

4.6 IPv6

Connection, connection-less service

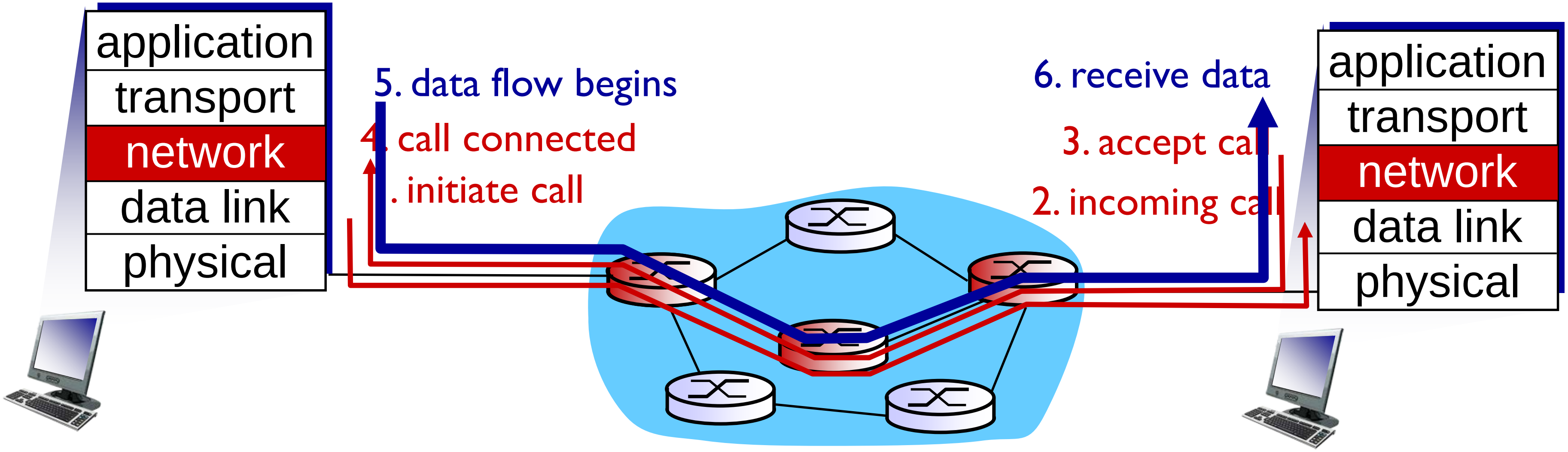
- *datagram* network provides network-layer *connectionless* service
  - Like UDP
  - IP is based on datagram service
- *virtual-circuit* network provides network-layer *connection* service
  - Like TCP
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service:* host-to-host
  - *no choice:* network provides one or the other
  - *implementation:* in network core

Virtual Circuit Model

- Three phases:
  1. Connection establishment, circuit set up
     - Path is chosen, circuit information stored in routers
  2. Data transfer, circuit is used
     - Packets are forwarded along the path
  3. Connection teardown, circuit s deleted
     - Circuit is removed from routers
- 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)

Virtual circuits: signaling protocols

- used in ATM, frame-relay, X.25
- not used in today's Internet



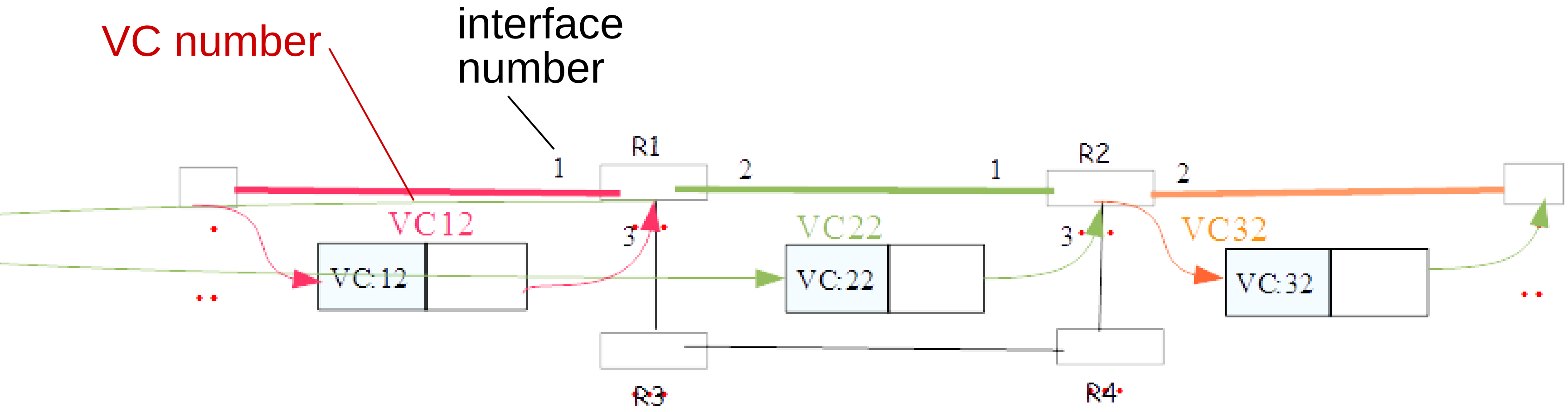
VC implementation

a VC consists of:

1. *path* from source to destination
2. *VC numbers*, one number for each link along path
   - VC numbers don't have any global meaning, only unique for a link
3. *entries in forwarding tables* in routers along path

- packet belonging to VC carries VC number (rather than destination address)
- VC number can be changed on each link.
  - new VC number comes from forwarding table

VC forwarding table



VC Forwarding Table of R1

Input interface	Input VC Number	Output Interface	Output VC Number
1	12	2	22

VC Forwarding Table of R2

Input Interface	Input VC Number	Output Interface	Output VC Number
1	22	2	32

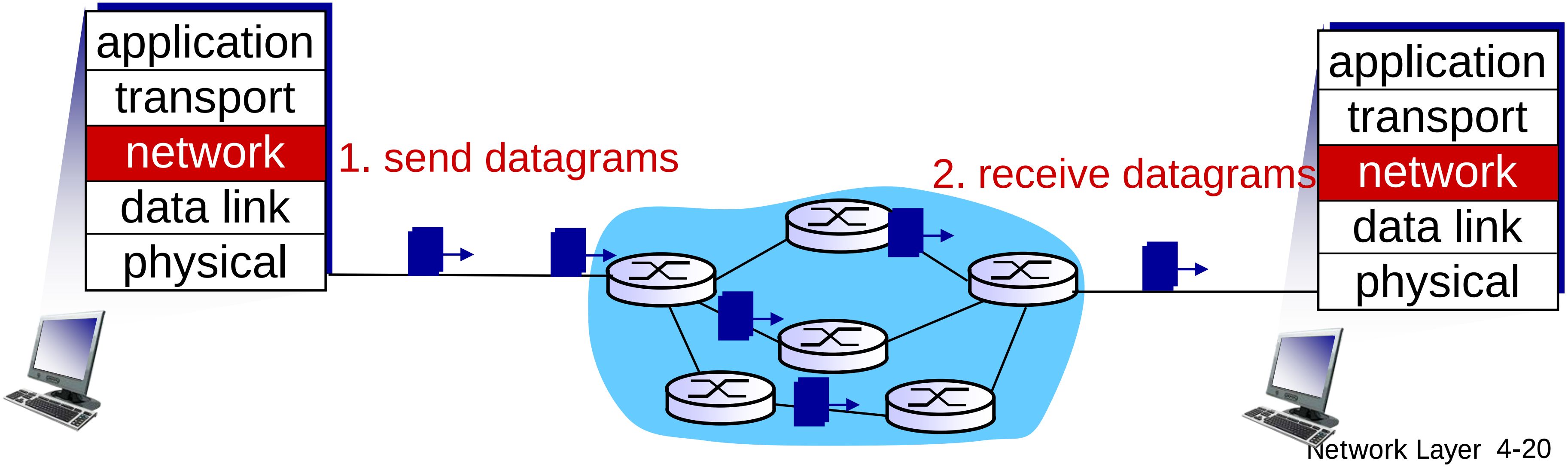
VC routers maintain connection state information!

Advantages and Disadvantages

- At least 1 RTT delay before sending data
  - Source host have to send the connection request to the destination host and wait this the connection acknowledgment
- Less header overhead
  - The connection request contains the complete address of the destination host (long and unique on the network), but the data packets only the VC identifier (short and unique in the link)
- Routers must store status information about VCs
- If a link fails, the connection falls and it must be re-established
- Buffer space reservation in routers to store packages if necessary

Datagram networks

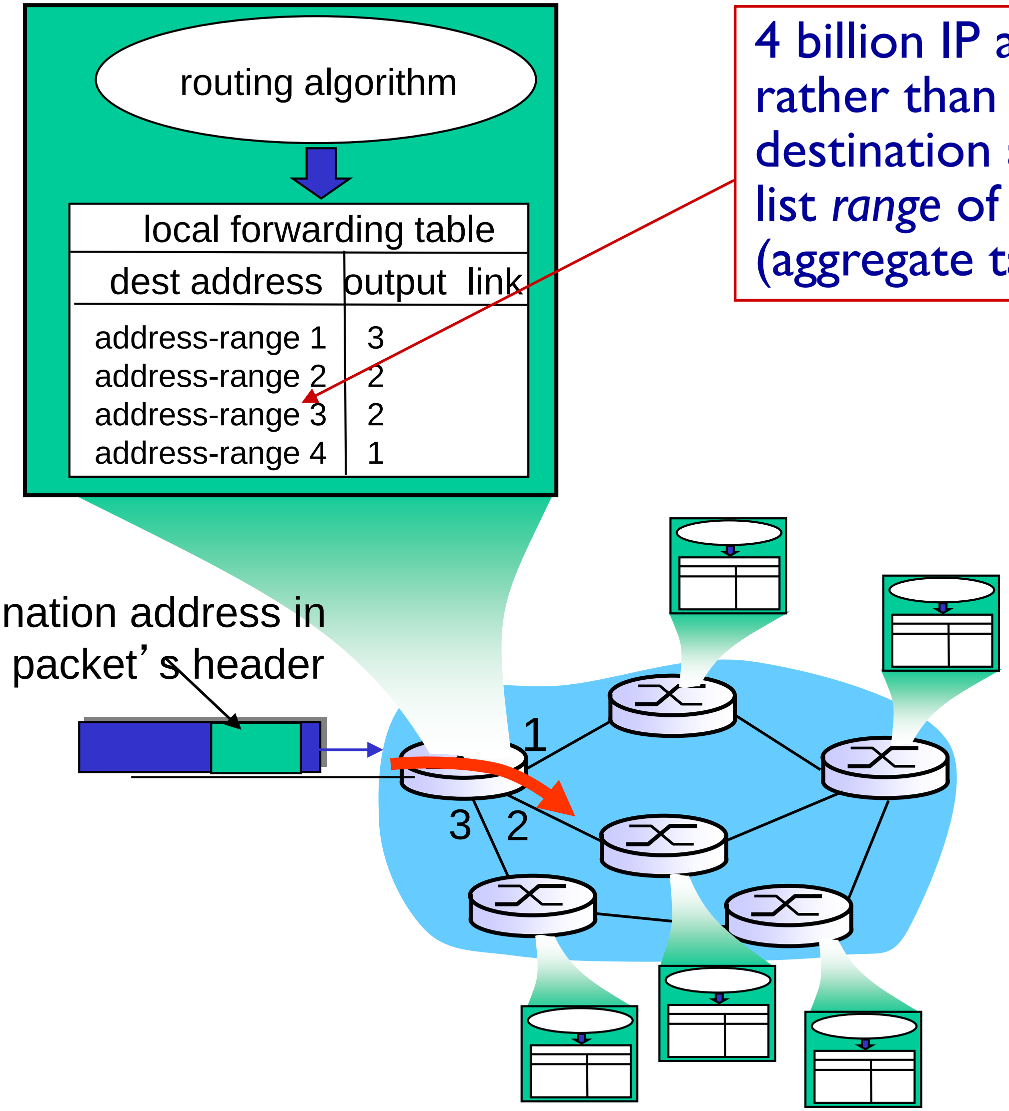
- Internet is a Datagram network
- no call setup at network layer (best-effort service)
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address
  - Packages between the same source and destination can follow different routes



Datagrams vs Virtual Circuits

Issue	Datagrams	Virtual Circuits
Setup phase	Not needed	Required
Router state	Per destination	Per connection
Addresses	Packet carries full address	Packet carries short labels (VCs id.)
Routing	Per packet	Per circuit
Quality of service	Difficult to add	Easier to add

Datagram forwarding table



routing algorithm

local forwarding table	
dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

IP destination address in arriving packet’s header

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 to **11001000 00010111 00010111 11111111**	0
11001000 00010111 00011000 00000000 to **11001000 00010111 00011000 11111111**	1
11001000 00010111 00011001 00000000 to **11001000 00010111 00011111 11111111**	2
otherwise	3

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

Longest prefix matching

longest prefix matching

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

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

examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

Unit 6: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing

4.4 routing algorithms

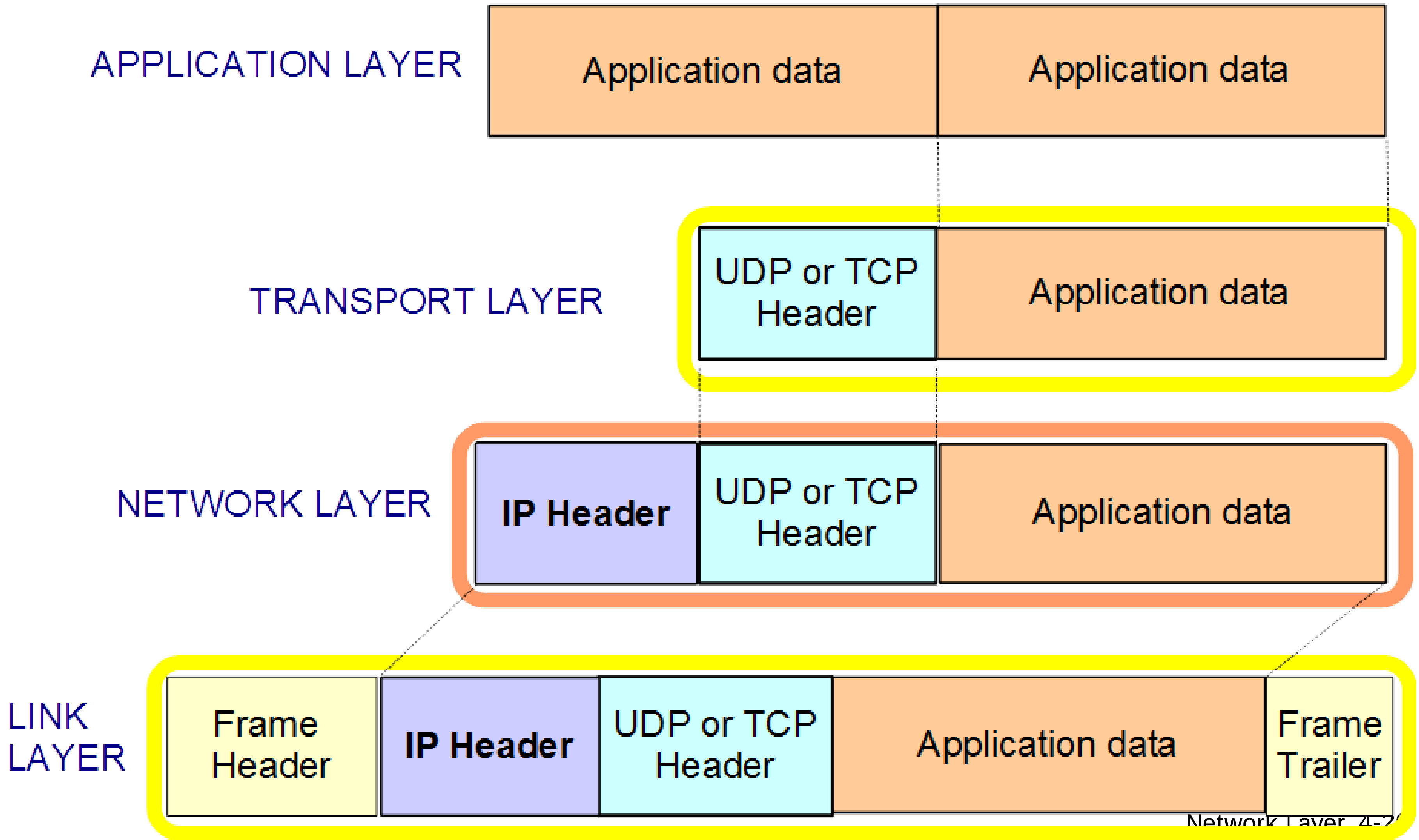
- distance vector
- link state
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

4.6 IPv6

Data Encapsulation



APPLICATION LAYER

Application data | Application data

TRANSPORT LAYER

UDP or TCP Header | Application data

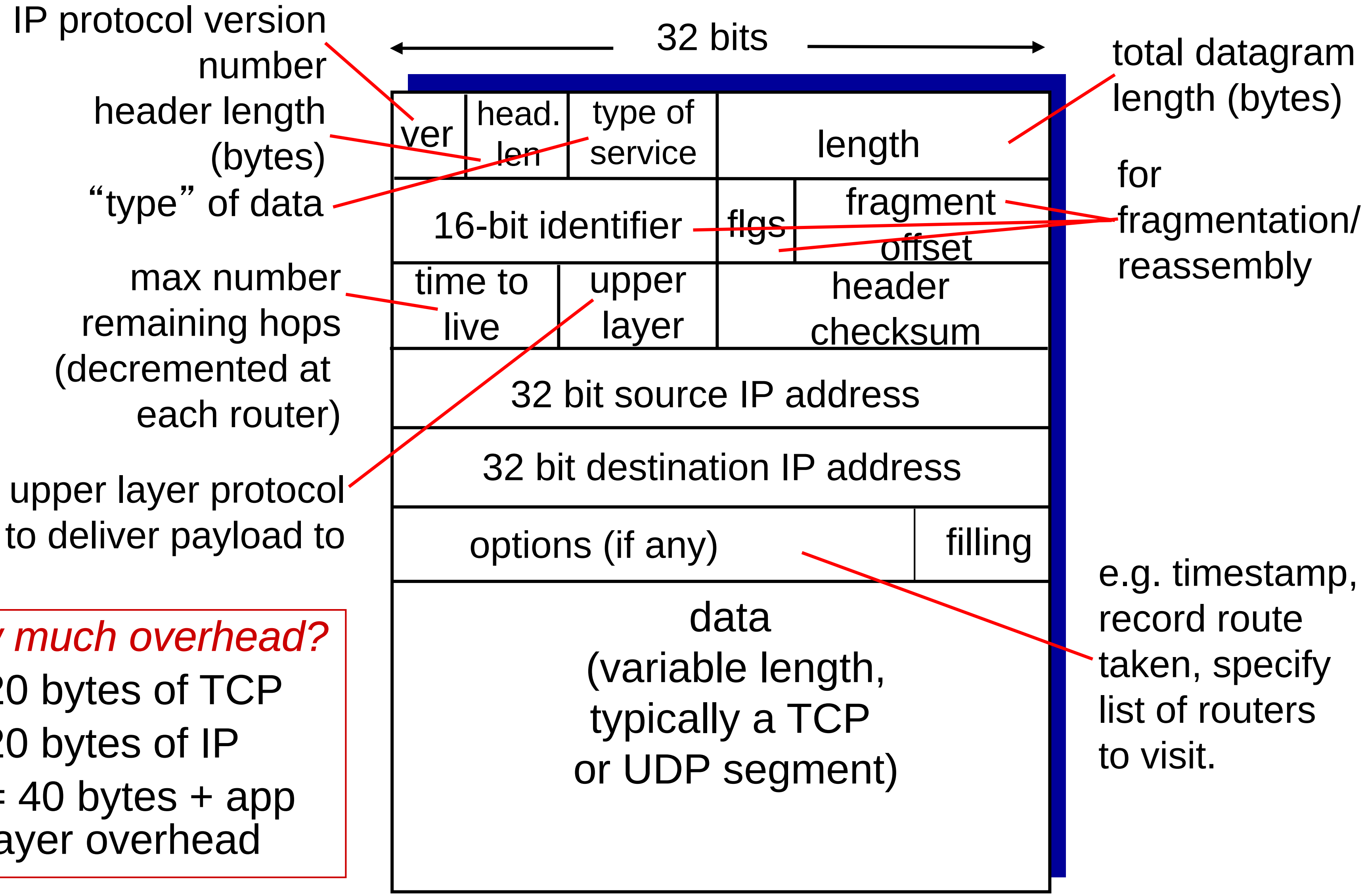
NETWORK LAYER

IP Header | UDP or TCP Header | Application data

LINK LAYER

Frame Header | IP Header | UDP or TCP Header | Application data | Frame Trailer

IP datagram format



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

32 bits

ver | head. len | type of service | length

16-bit identifier | flgs | fragment offset

time to live | upper layer | header checksum

32 bit source IP address

32 bit destination IP address

options (if any) | filling

data (variable length, typically a TCP or UDP segment)

total datagram length (bytes)

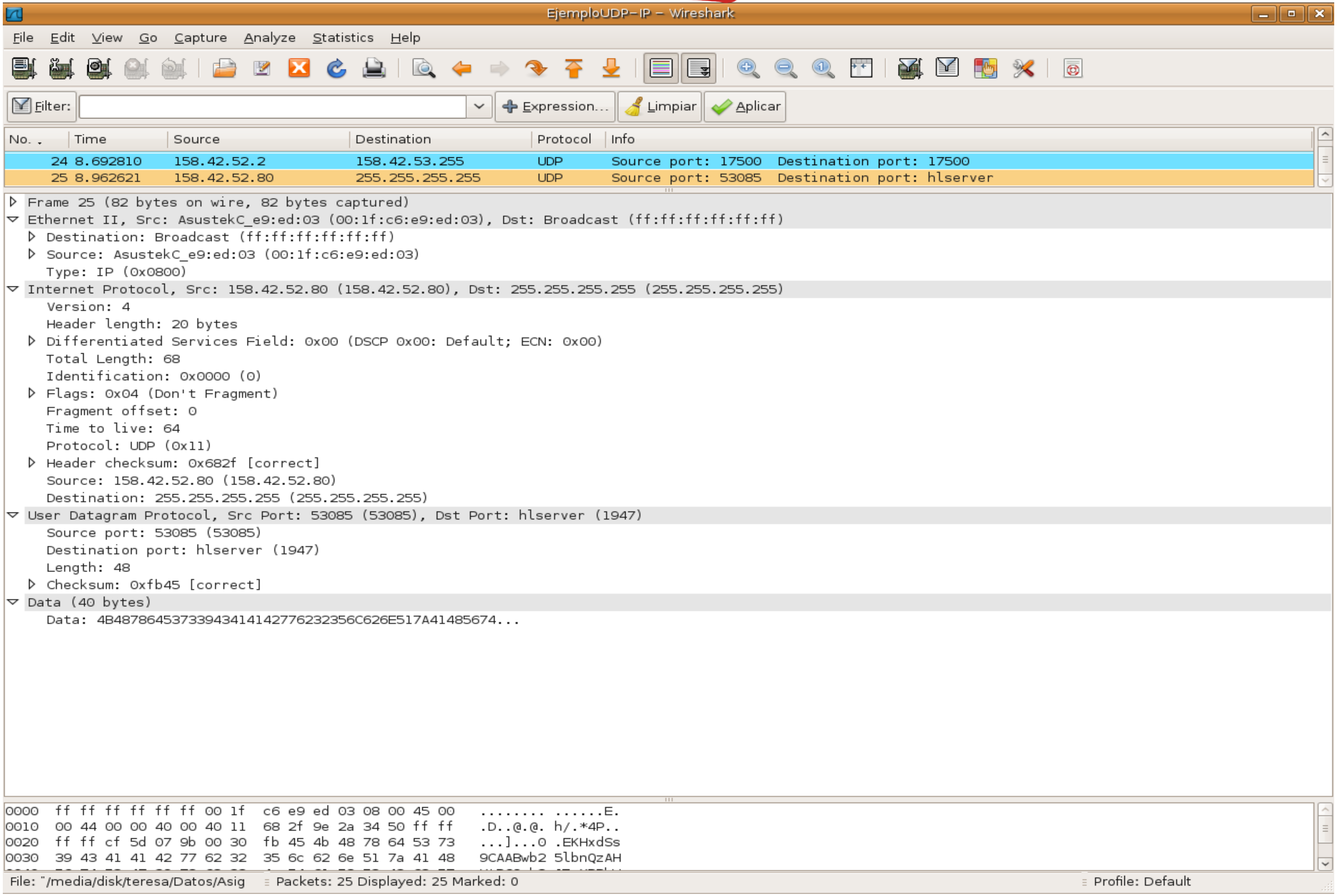
for fragmentation/ reassembly

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 datagram format



```
No. .   Time        Source          Destination        Protocol  Info
   24 8.692810    158.42.52.2     158.42.53.255      UDP       Source port: 17500  Destination port: 17500
   25 8.962621    158.42.52.80    255.255.255.255    UDP       Source port: 53085  Destination port: hlserver
```

```
▷ Frame 25 (82 bytes on wire, 82 bytes captured)
▽ Ethernet II, Src: AsustekC_e9:ed:03 (00:1f:c6:e9:ed:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Source: AsustekC_e9:ed:03 (00:1f:c6:e9:ed:03)
    Type: IP (0x0800)
▽ Internet Protocol, Src: 158.42.52.80 (158.42.52.80), Dst: 255.255.255.255 (255.255.255.255)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 68
    Identification: 0x0000 (0)
  ▷ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (0x11)
  ▷ Header checksum: 0x682f [correct]
    Source: 158.42.52.80 (158.42.52.80)
    Destination: 255.255.255.255 (255.255.255.255)
▽ User Datagram Protocol, Src Port: 53085 (53085), Dst Port: hlserver (1947)
    Source port: 53085 (53085)
    Destination port: hlserver (1947)
    Length: 48
  ▷ Checksum: 0xfb45 [correct]
▽ Data (40 bytes)
    Data: 4B48786453733943414142776232356C626E517A41485674...
```

```
0000  ff ff ff ff ff ff 00 1f  c6 e9 ed 03 08 00 45 00   ........ ......E.
0010  00 44 00 00 40 00 40 11  68 2f 9e 2a 34 50 ff ff   .D..@.@. h/.*4P..
0020  ff ff cf 5d 07 9b 00 30  fb 45 4b 48 78 64 53 73   ...]...0 .EKHxdSs
0030  39 43 41 41 42 77 62 32  35 6c 62 6e 51 7a 41 48   9CAABwb2 5lbnQzAH
```

```
File: "/media/disk/teresa/Datos/Asig    Packets: 25 Displayed: 25 Marked: 0    Profile: Default
```

IPv4 datagram fields

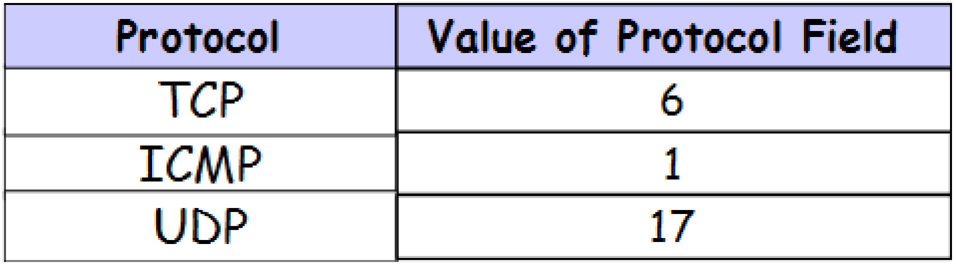
- Version number (4 bits).
  - By looking at the version number, the router can determine how to interpret the remainder of the IP datagram.
- Header length (4 bits).
  - Expressed in words of 32 bits (minimum value = 5)
  - To determine where in the IP datagram the data actually begins.
    - Because an IPv4 datagram can contain a variable number of options
- Datagram length (16 bits).
  - Total length of the IP datagram (header plus data), measured in bytes.
  - Maximum length = 65.535 bytes

IPv4 datagram fields

- Type of service (TOS) (8 bits).
  - 3 bits for the priority (ignored), 4 bits for the type of service and 1 bit to zero.
  - The 4-bit type of service allows the user to request the desired conditions (only one bit to 1):
    - *Minimizing delays* 1000
    - *Maximize productivity* 0100
    - *Maximize reliability* 0010
    - *Minimize cost* 0001
  - These values can be helpful in routing decisions ... but is not guaranteed the type of service requested.
  - RFC 1349

IPv4 datagram fields

- Time-to-live (TTL).
  - To ensure that datagrams do not circulate forever (due to, for example, a long-lived routing loop) in the network.
  - This field is decremented by one each time the datagram is processed by a router. If the TTL field reaches 0, the datagram must be dropped.
- Protocol.
  - Used only when an IP datagram reaches its final destination to indicate the specific protocol to which the data portion of this IP datagram should be passed.



Protocol	Value of Protocol Field
TCP	6
ICMP	1
UDP	17

IPv4 datagram fields

- Header checksum.
  - The checksum must be recomputed and stored again at each router, as the TTL field, and possibly the options field as well, may change.
- Options.
  - Allow an IP header to be extended.
  - Header options were meant to be used rarely.
  - Lets specify: source routing (using a chosen route) timestamp, record route (RR), etc.
  - It has variable length (that determines the length of the filling field)

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

4.4 routing algorithms

- distance vector
- link state
- hierarchical routing

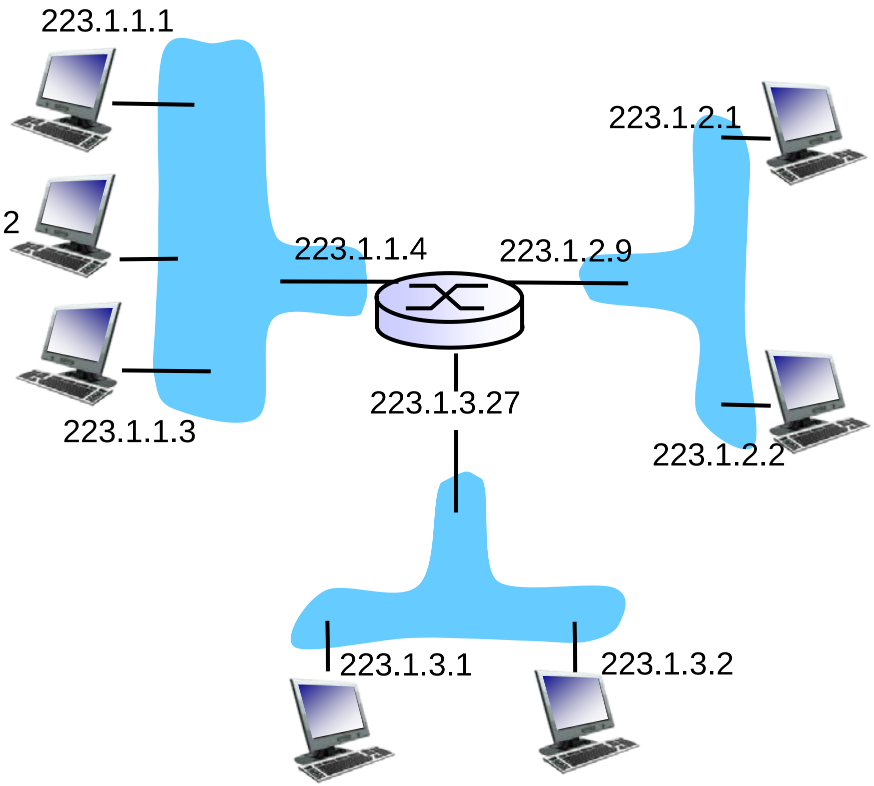
4.5 routing in the Internet

- RIP
- OSPF
- BGP

4.6 IPv6

IP addressing: introduction

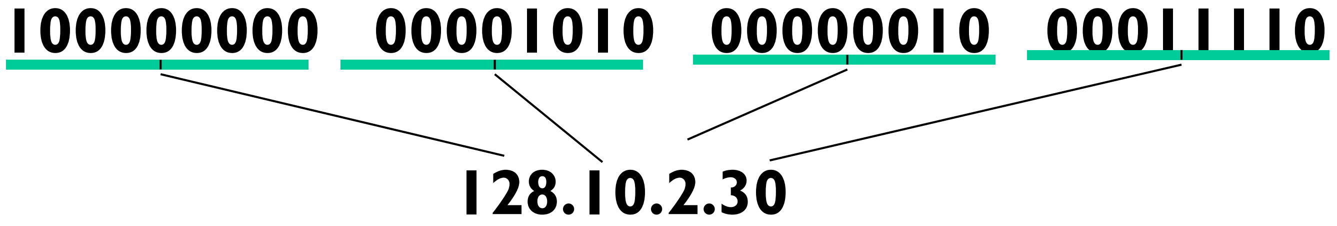
- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - routers typically have multiple interfaces
  - host typically has one active interface (e.g., wired Ethernet, wireless 802.11)
- *one IP address associated with each interface*



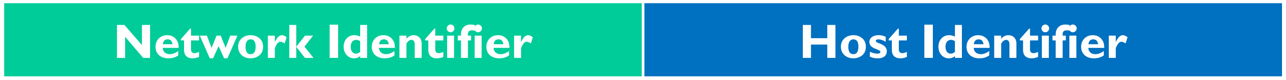
$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addresses v4

- IP addresses v4 are represented as four decimal numbers obtained from the four bytes that make up the IP address (n1.n2.n3.n4)



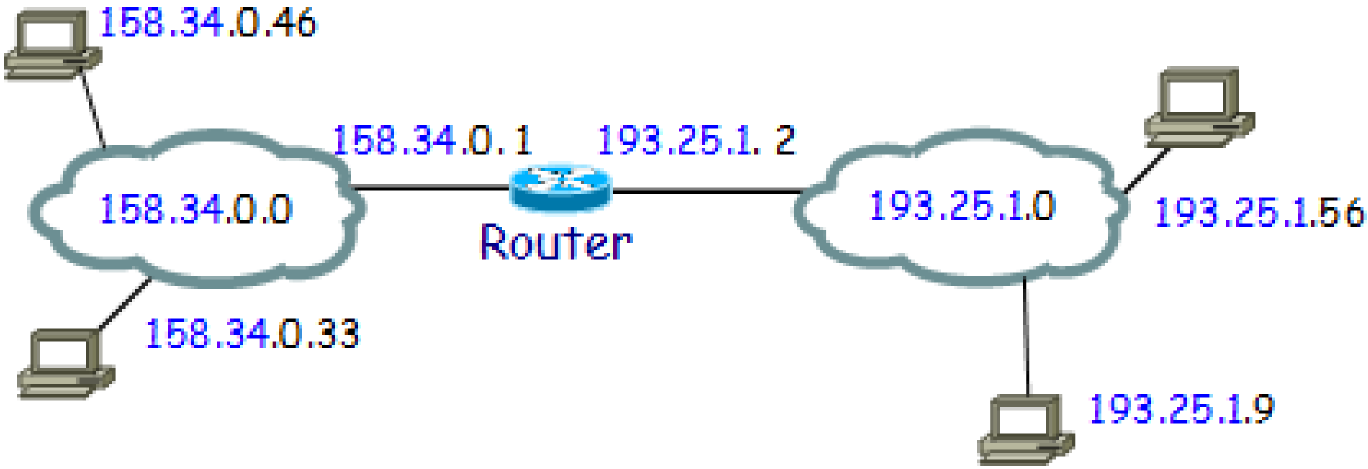
- Each IP address has two fields:



  - All systems (hosts and routers) connected to the same network share the same IP network Identifier (network ID or network prefix)

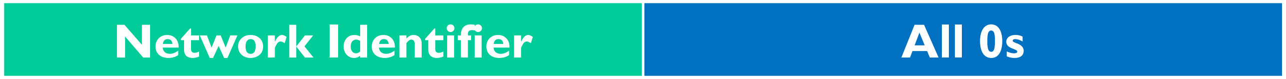
Routers and IP addressing

- Each router has at least two IP addresses:
  - A router connects multiple networks (at least two)
  - Each IP address has a different network identifier



Special IP Addresses

❖ Network IP Address

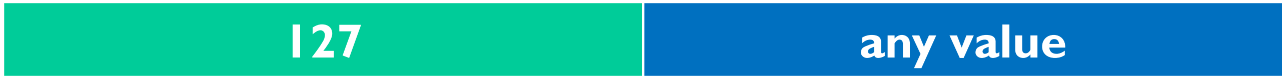


Network Identifier | All 0s

- Example: 158.34.0.0 identifies the 158.34 network
- Refers to that entire network, not to an specific device on that network
- This address **never** can be source or destination address in an IP datagram

Special IP Addresses

Loopback Address

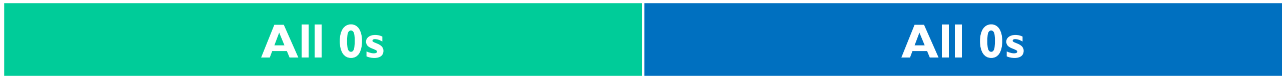


127	any value

- Example: 127.0.0.1
- The loopback address has no hardware associated with it, and it is not physically connected to a network.
- The loopback enables a user to test network applications without being connected to the network.

Special IP Addresses

- **This host Address**

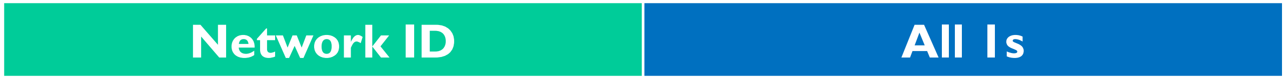


All 0s All 0s

- Example: 0.0.0.0
- The address of the host (which sends the datagram)
- It is used as source address when the host obtains its IP address automatically through the network
  - When DHCP protocol is used

Special IP Addresses

- **Directed Broadcast Addresses**

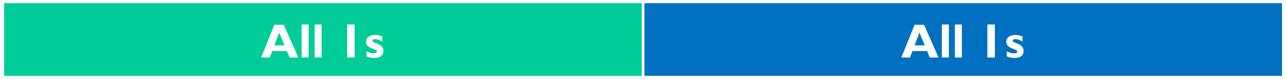


Network ID | All 1s

  - Example: 158.42.255.255
  - To send a copy of a packet to all computers on an IP network
  - It sends a single copy of the package over the Internet

Special IP Addresses

- **Limited Broadcast Addresses**


All 1s

All 1s

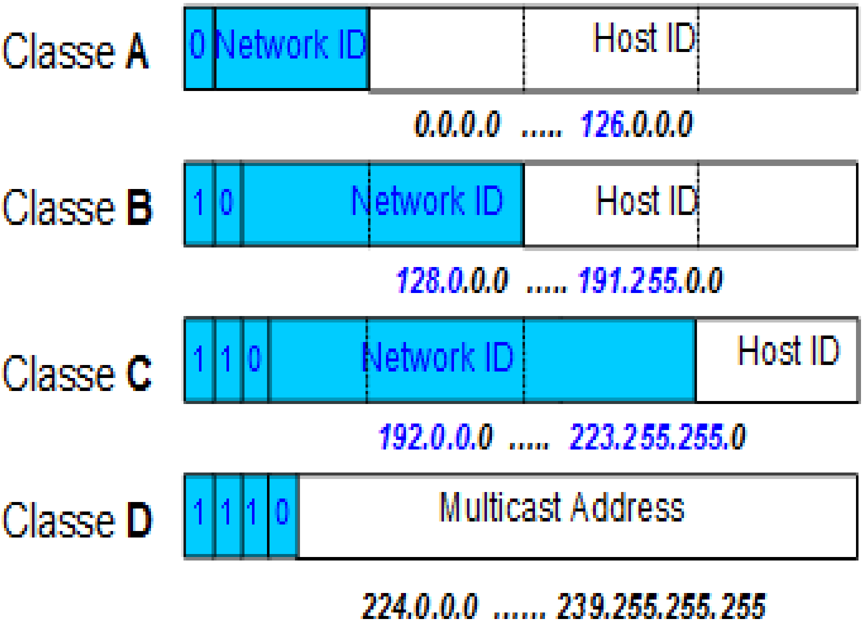
- Example: 255.255.255.255
- To send a copy of a packet to all computers on the network to which is connected the sending host.
- Used as destination address when host starts, and still does not know its IP address.
  - DHCP protocol uses it

Types of Internet Addressing

- Two types of Internet addressing, depending on how it determines the length of the network prefix:
  - Classful IP addressing:
    - The network ID portion can take only the predefined number of bits 8, 16, or 24.
  - Classless IP addressing (CIDR = Classless Inter-Domain Routing)
    - Any number of bits can be assigned to the network ID.
    - It requires a network mask to know the number of bits that identify the network prefix.
    - Example: 158.42.0.0/16

Types of Internet Addressing

- Classful IP addressing:
  - The network ID portion can take only the predefined number of bits 8, 16, or 24.



Network ID: 8 bits

Network ID: 16 bits

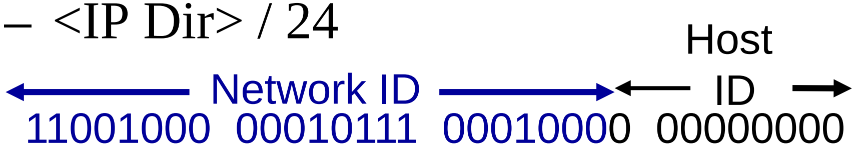
Network ID: 24 bits

Classful IP addressing

- Classful IP addressing problem:
  - The inflexibility of the class system accelerated IPv4 address pool exhaustion.
    - Example:
      - An organization that needs 258 hosts would get a Class B license, even though it would have far fewer than 65,534 hosts.
      - This resulted in most of the block of addresses allocated going unused.

CIDR. Classless Inter-Domain Routing

- Address format: a.b.c.d/x
- The network prefix is specify using a netmask
- The network mask:
  - Lets you know which bits belong to network identifier and which to host identifier
  - It has the same size as IP address (32 bits) and its bits to1identify the network prefix.
  - Two equivalent ways to express it:
    - Example for 24-bit netmask:
      - 255.255.255.0
      - <IP Dir> / 24



Network ID

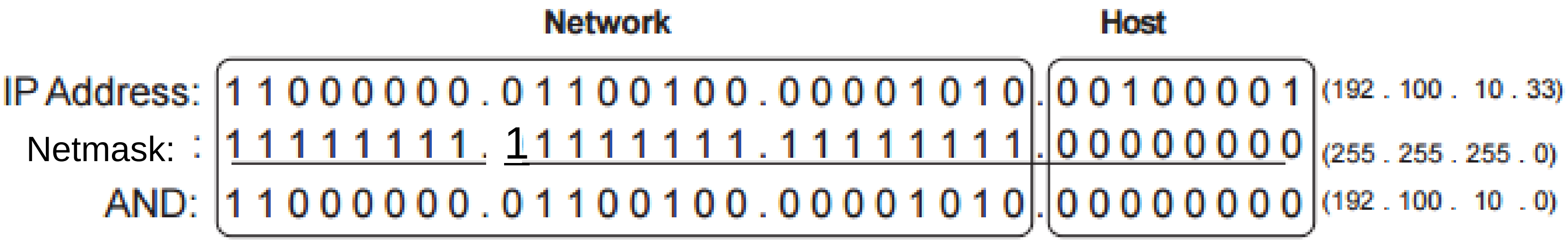
Host ID

11001000 00010111 00010000 00000000

200.23.16.0/23

ANDING with netmasks

- In order to get the network address you must AND the IP address with the netmask in binary



	Network	Host	
IP Address:	1 1 0 0 0 0 0 0 . 0 1 1 0 0 1 0 0 . 0 0 0 0 1 0 1 0 .	0 0 1 0 0 0 0 1	(192 . 100 . 10 . 33)
Netmask: :	1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 .	0 0 0 0 0 0 0 0	(255 . 255 . 255 . 0)
AND:	1 1 0 0 0 0 0 0 . 0 1 1 0 0 1 0 0 . 0 0 0 0 1 0 1 0 .	0 0 0 0 0 0 0 0	(192 . 100 . 10 . 0)

Remember …

ANDING Equations:

1 AND 1 = 1

1 AND 0 = 0

0 AND 1 = 0

0 AND 0 = 0

Private IP Addresses (RFC 1918)

- Addresses within private address space will only be unique within the Private Network (within an enterprise/organization).
  - The address space can thus be used by many Private Networks.
  - Hosts that do not require access to hosts in other enterprises or the Internet at large can use private addresses.
- Routers don't route private addresses out into the Internet.
- Private address ranges
  - 192.168.0.0/16: 192.168.0.0 - 192.168.255.255 (65,536 IP addresses)
  - 172.16.0.0/12 : 172.16.0.0 - 172.31.255.255 (1,048,576 IP addresses)
  - 10.0.0.0/8: 10.0.0.0 - 10.255.255.255 (16,777,216 IP addresses)

Subnets and Route Aggregation

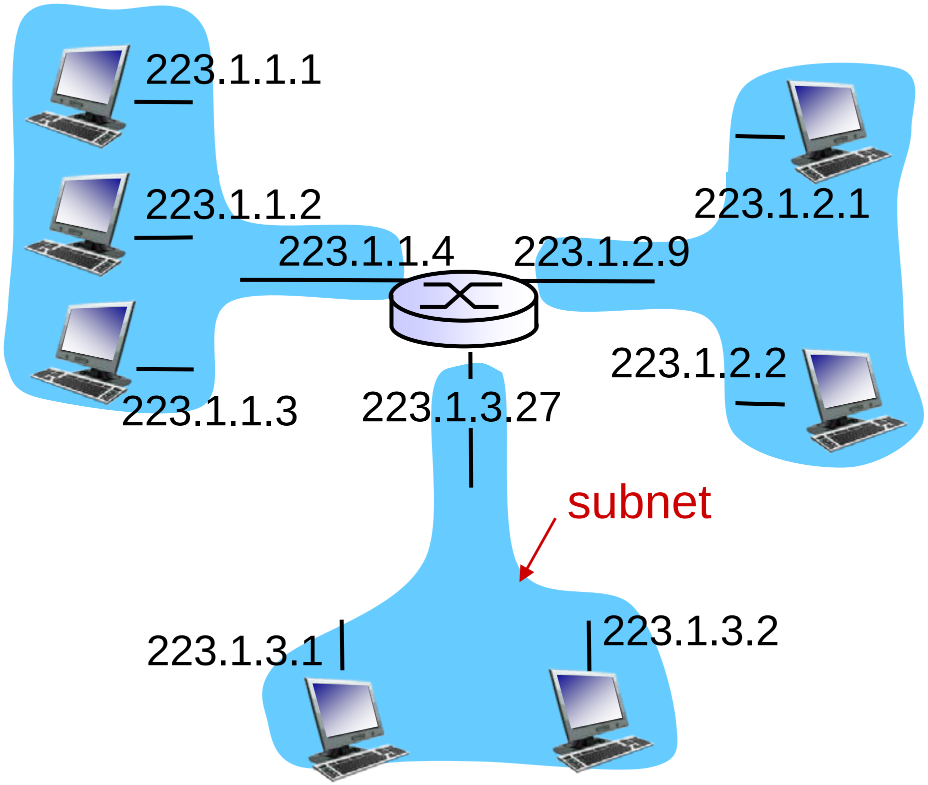
- Why Subnets?
  - To identify individual parts of an organization, ie, divide the network into subnetworks.
  - How? A subnet mask borrows bits from the host portion of the address to create a subnetwork address between the network and host portions of an IP address.
    - Some bits of the host ID will become bits of the network ID
- Why Route Aggregation?
  - To express a set of IP addresses on a single IP address or fewer than the initial set of IP addresses.
  - How? Making that some bits of the network identification become bits of the host ID

Subnets example

- Network 192.228.17.0/24
  - 192.228.17.0 = 11000000.11100100.00010001.00000000
  - Netmask /24:
  - 255.255.255.0 = 11111111.11111111.11111111.00000000
  - Can have up to 254 *hosts*
    - *# Host ID: 8 bits -> 2^8= 256 possible IP Addresses*
    - *256-2 (Network Address, and Broadcast Address) = 254 Host IP Addresses*
- We want to divide it into 8 subnets
  - to address 8 subnets need 3 bits (8 = 2^3)
  - The Host ID gives the 3 bits to identify the subnetwork
  - New (sub)network mask: /27
    - 255.255.255.**224** = 11111111.11111111.11111111.**111**00000
  - # Host ID: 5 bits -> 30 *hosts* each subnetwork (2^5-2)

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*

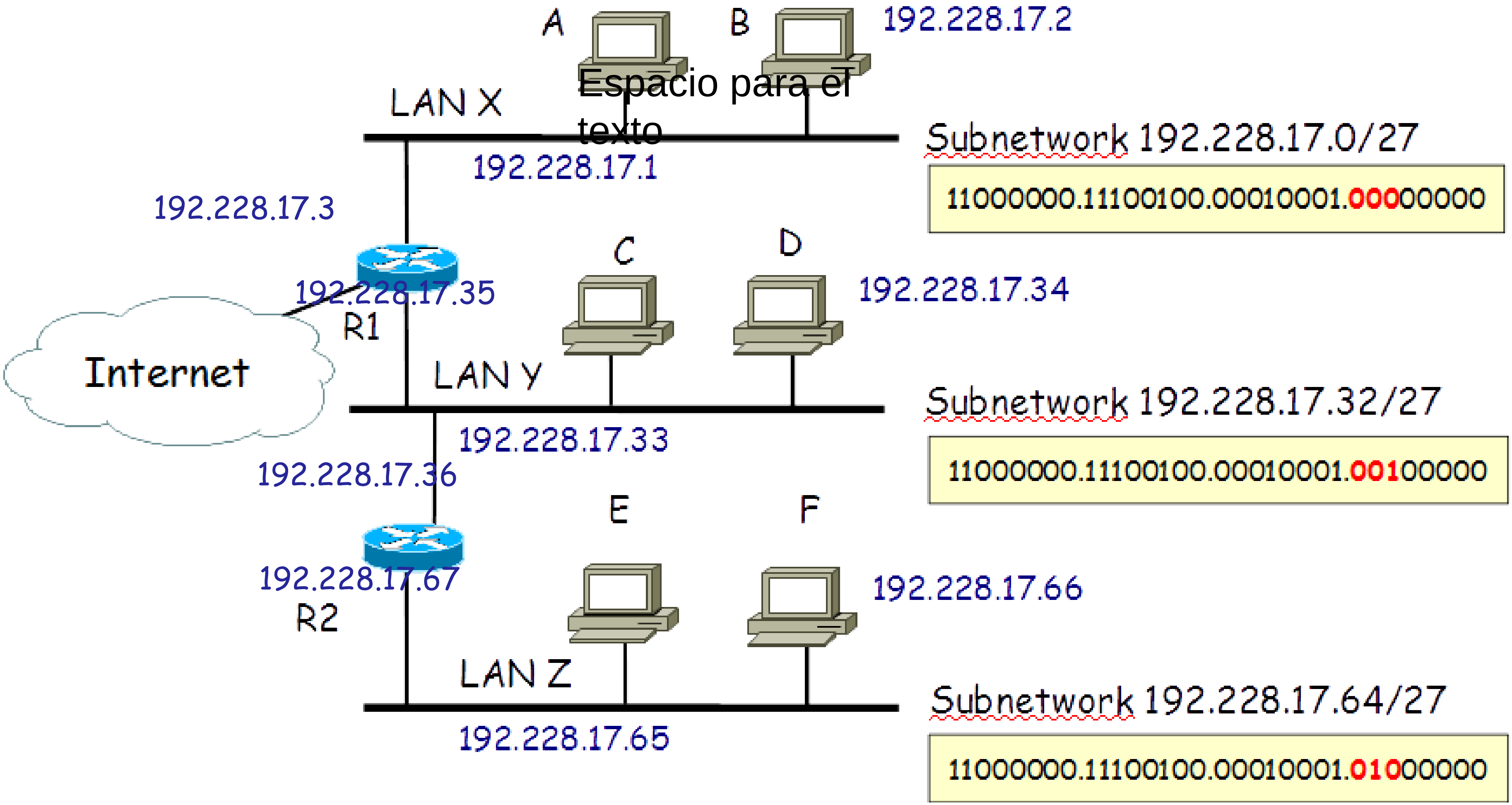


network consisting of 3 subnets

Subnets

Network 192.228.17.0 11000000.11100100.00010001.00000000

Network mask:
255.255.255.224 11111111.11111111.11111111.11100000

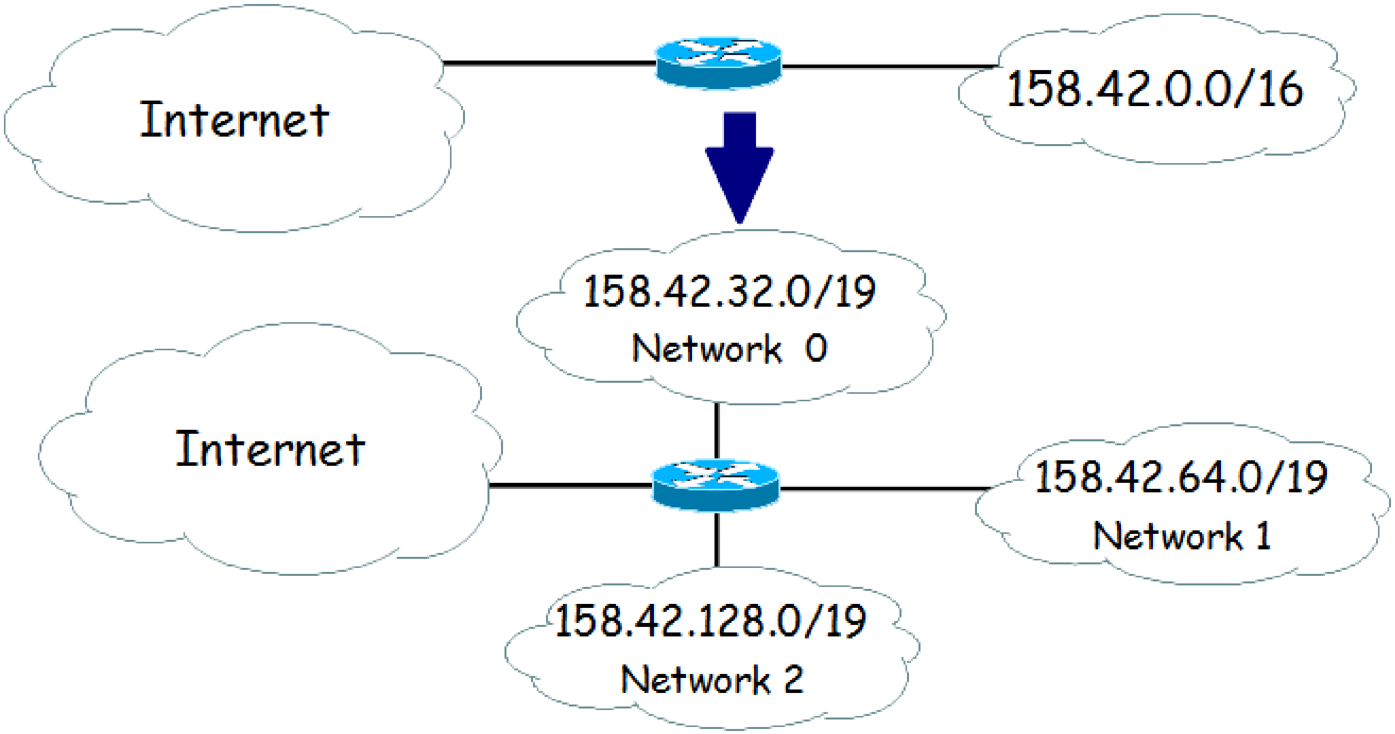


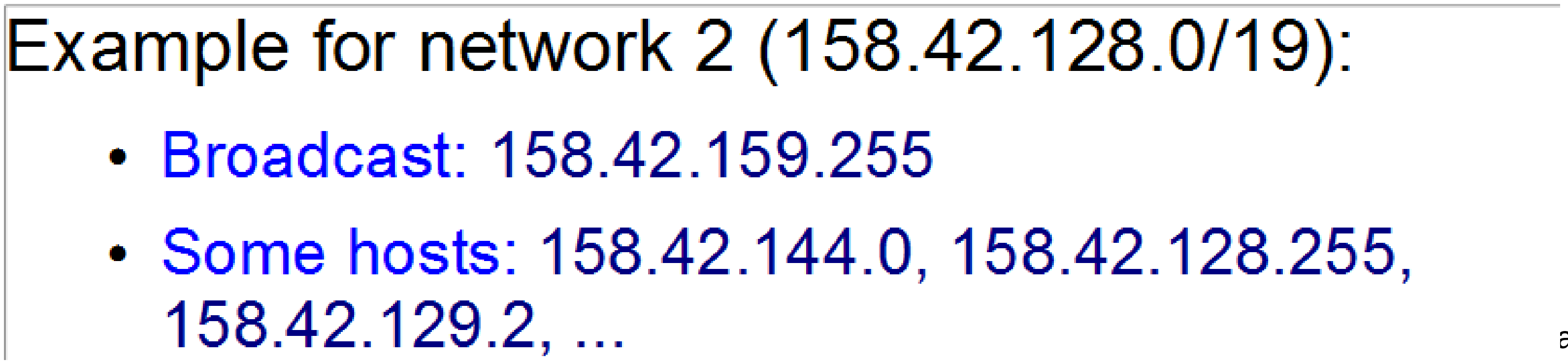
Subnetwork 192.228.17.0/27
11000000.11100100.00010001.00000000

Subnetwork 192.228.17.32/27
11000000.11100100.00010001.00100000

Subnetwork 192.228.17.64/27
11000000.11100100.00010001.01000000

Subnets





Example for network 2 (158.42.128.0/19):

- Broadcast: 158.42.159.255
- Some hosts: 158.42.144.0, 158.42.128.255, 158.42.129.2, ...

Route Aggregation

- The 8 networks from 194.32.136.0/24 to 194.32.143.0/24 have a common prefix of 21 bits:

11000010 00100000 10001000 00000000

11000010 00100000 10001001 00000000

. . .

11000010 00100000 10001111 00000000

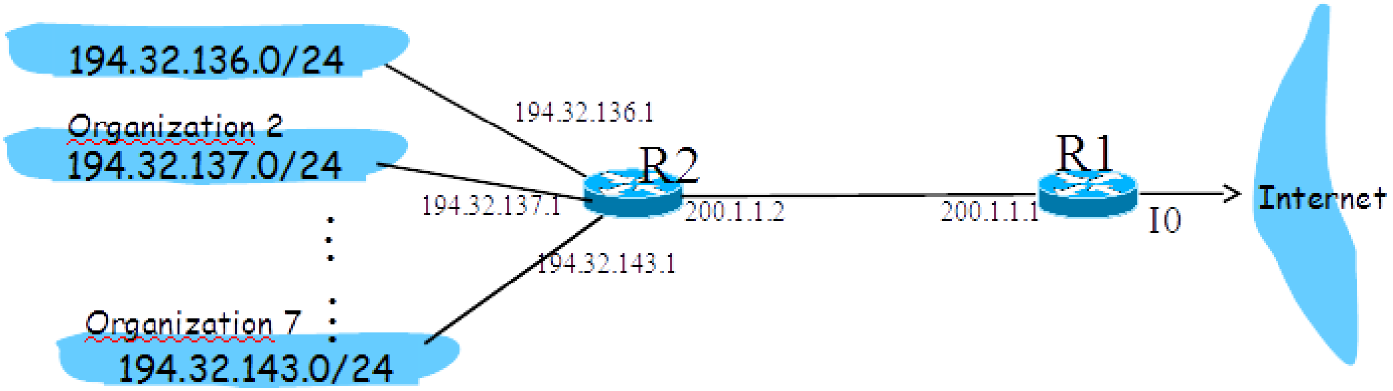
- This contiguous block of addresses can be expressed as a single (super) network:

194.32.136.0/21

The new network mask (supernetwork mask) is:

255.255.248.0

Route Aggregation



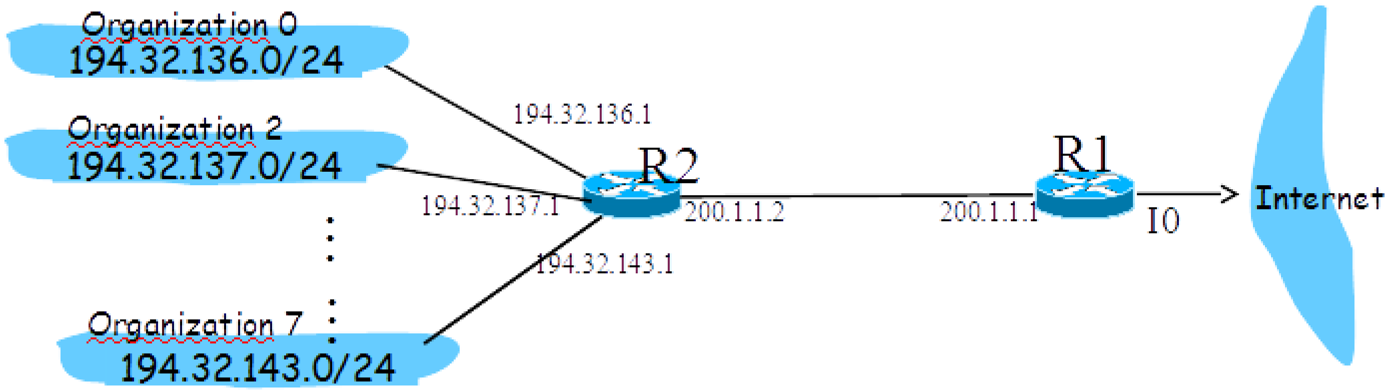
Forwarding Table of R1

Network Destination	Network mask	Rute	Interface
194.32.136.0	255.255.255.0	200.1.1.2	200.1.1.1
194.32.137.0	255.255.255.0	200.1.1.2	200.1.1.1
194.32.138.0	255.255.255.0	200.1.1.2	200.1.1.1
194.32.139.0	255.255.255.0	200.1.1.2	200.1.1.1
194.32.140.0	255.255.255.0	200.1.1.2	200.1.1.1
194.32.141.0	255.255.255.0	200.1.1.2	200.1.1.1
194.32.142.0	255.255.255.0	200.1.1.2	200.1.1.1
194.32.143.0	255.255.255.0	200.1.1.2	200.1.1.1
0.0.0.0	0.0.0.0	Intemet	I0

Forwarding Table of R2

Network Destination	Network mask	Rute	Interface
194.32.136.0	255.255.255.0	0.0.0.0	194.32.136.1
194.32.137.0	255.255.255.0	0.0.0.0	194.32.137.1
194.32.138.0	255.255.255.0	0.0.0.0	194.32.138.1
194.32.139.0	255.255.255.0	0.0.0.0	194.32.139.1
194.32.140.0	255.255.255.0	0.0.0.0	194.32.140.1
194.32.141.0	255.255.255.0	0.0.0.0	194.32.141.1
194.32.142.0	255.255.255.0	0.0.0.0	194.32.142.1
194.32.143.0	255.255.255.0	0.0.0.0	194.32.143.1
0.0.0.0	0.0.0.0	200.1.1.1	200.1.1.2

Route Aggregation



Forwarding Table of R1

Network Destination	Network mask	Rute	Interface
194.32.136.0	255.255.248.0	200.1.1.2	200.1.1.1
0.0.0.0	0.0.0.0	Internet	I0

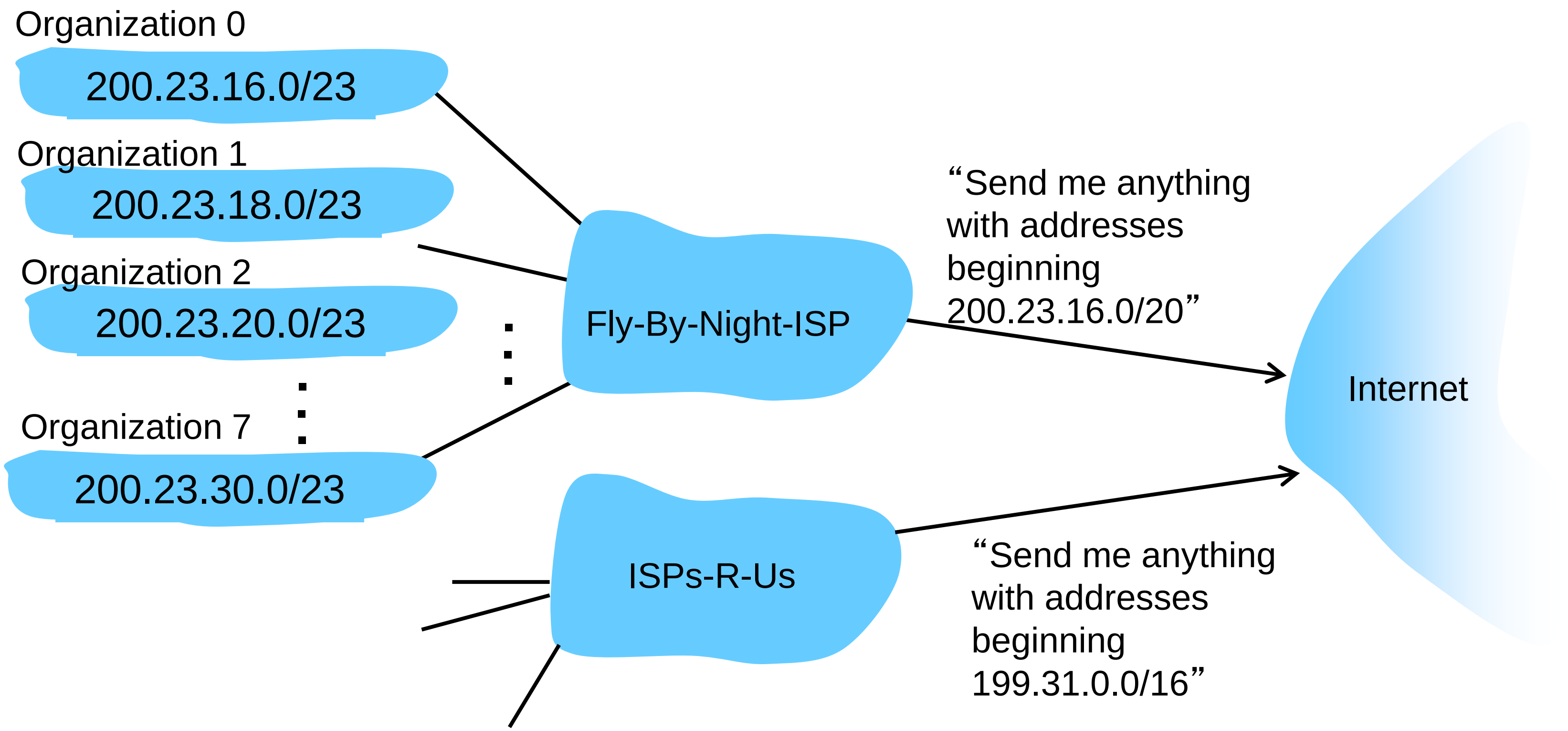
Address Aggregation reduces the size of the forwarding tables!!

Forwarding Table of R2

Network Destination	Network mask	Rute	Interface
194.32.136.0	255.255.255.0	0.0.0.0	194.32.136.1
194.32.137.0	255.255.255.0	0.0.0.0	194.32.137.1
194.32.138.0	255.255.255.0	0.0.0.0	194.32.138.1
194.32.139.0	255.255.255.0	0.0.0.0	194.32.139.1
194.32.140.0	255.255.255.0	0.0.0.0	194.32.140.1
194.32.141.0	255.255.255.0	0.0.0.0	194.32.141.1
194.32.142.0	255.255.255.0	0.0.0.0	194.32.142.1
194.32.143.0	255.255.255.0	0.0.0.0	194.32.143.1
0.0.0.0	0.0.0.0	200.1.1.1	200.1.1.2

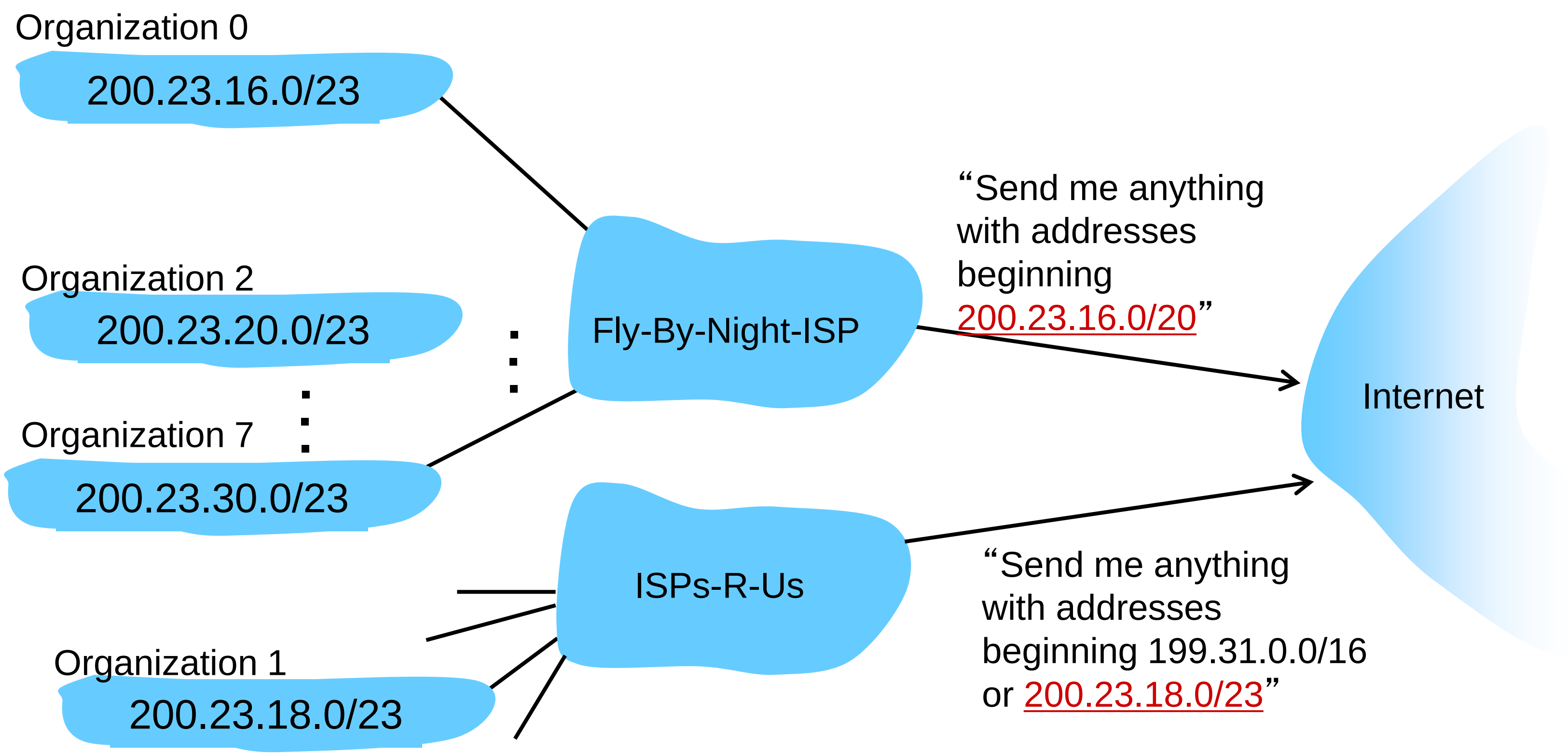
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

IP addresses: how to get one?

- How does host get an IP address?
  - Manually: configured by the system administrator
  - Using the Dynamic Host Configuration Protocol (DHCP, Dynamic Host Configuration Protocol) *

* RFC 2131

IP addresses: how to get one?

Q: how does an organization get a network IP addr?

A: Through an ISP (Movistar, Orange, Vodafone ...)

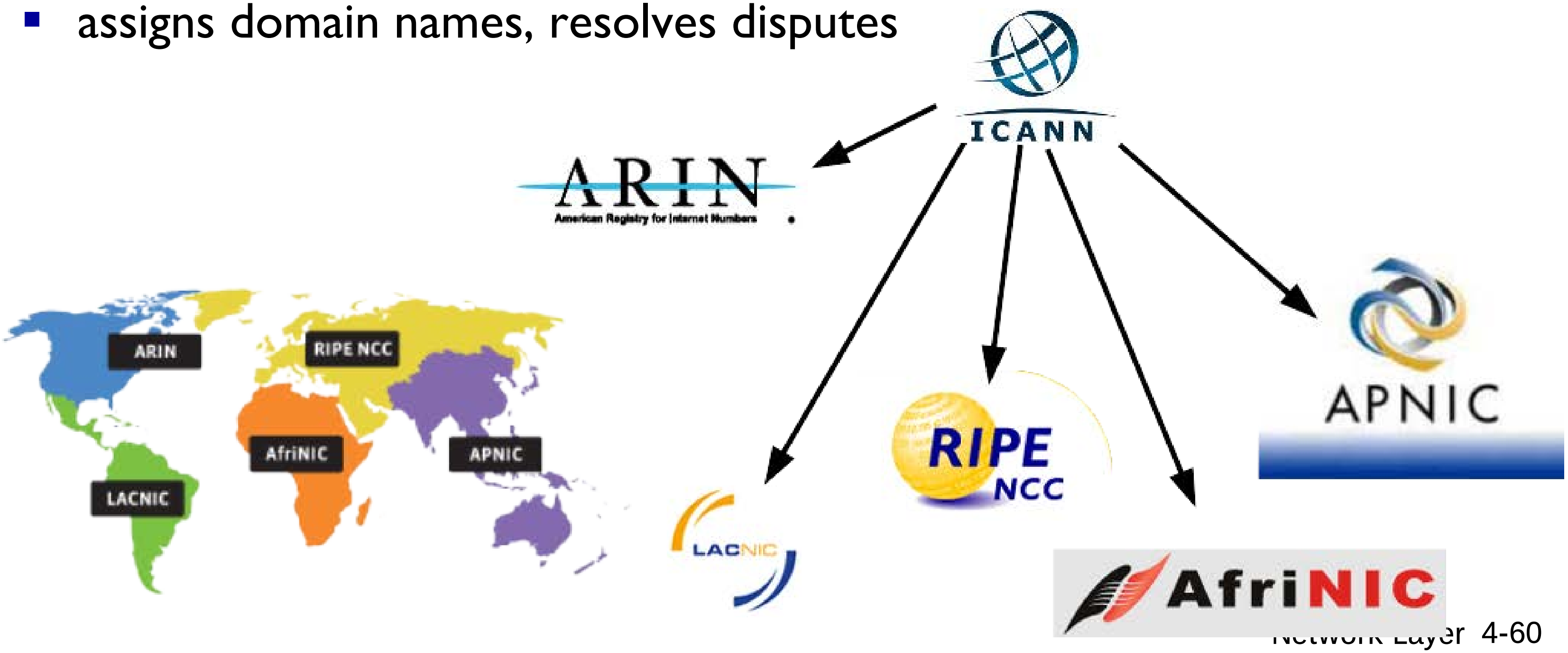
ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000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

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



Forwarding and Routing

- *forwarding:*
  - move packets from router' s input to appropriate router output
  - router local action
- *routing:*
  - determine route taken by packets from source to destination
    - *routing algorithms*
  - network wide process

analogy:

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

Forwarding Tables

- They contain information about the possible destination networks and how to reach them
- Where are they? In *routers* and *hosts*
- How should they be?
  - Compact, with a reduced number of entries to get better performances
  - Only information about destination networks and the next router to reach them.

Forwarding table entries

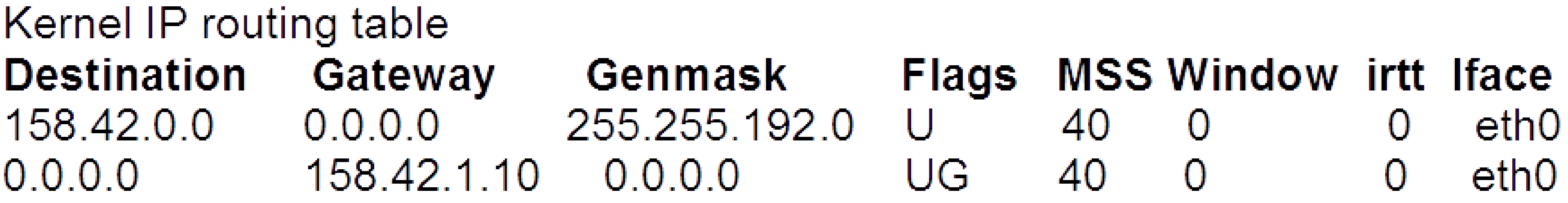
- Information in the forwarding table:

Destination Network	Netmask	Rute (next hop)	Output Interface

- Each entry of the forwarding table indicates the next hop node to reach the destination network
- When an IP datagram arrives :
  - node (router/host) must AND the destination IP address with the netmask in binary and forward the datagram for the forwarding table entry with the *longest* address prefix that matches destination network address.

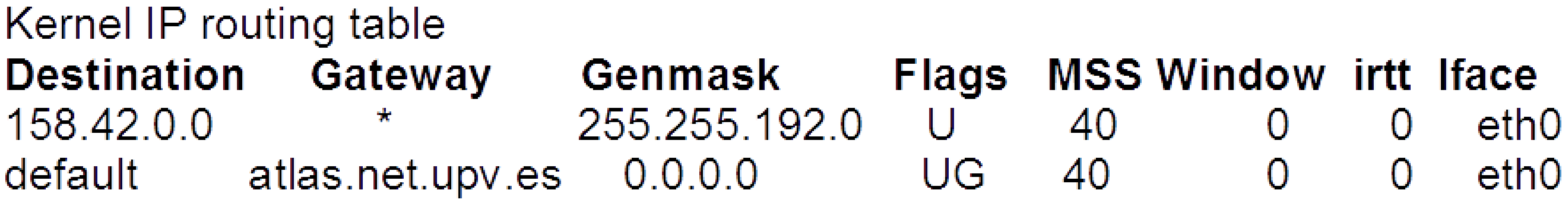
How can we see the forwarding table?

- `netstat -nr`



```
Kernel IP routing table
Destination    Gateway      Genmask        Flags   MSS Window  irtt  Iface
158.42.0.0     0.0.0.0      255.255.192.0  U        40    0        0    eth0
0.0.0.0        158.42.1.10   0.0.0.0       UG       40    0        0    eth0
```

- `netstat -r`



```
Kernel IP routing table
Destination    Gateway          Genmask        Flags   MSS Window  irtt  Iface
158.42.0.0         *            255.255.192.0   U       40         0     0    eth0
default       atlas.net.upv.es    0.0.0.0       UG      40         0     0    eth0
```

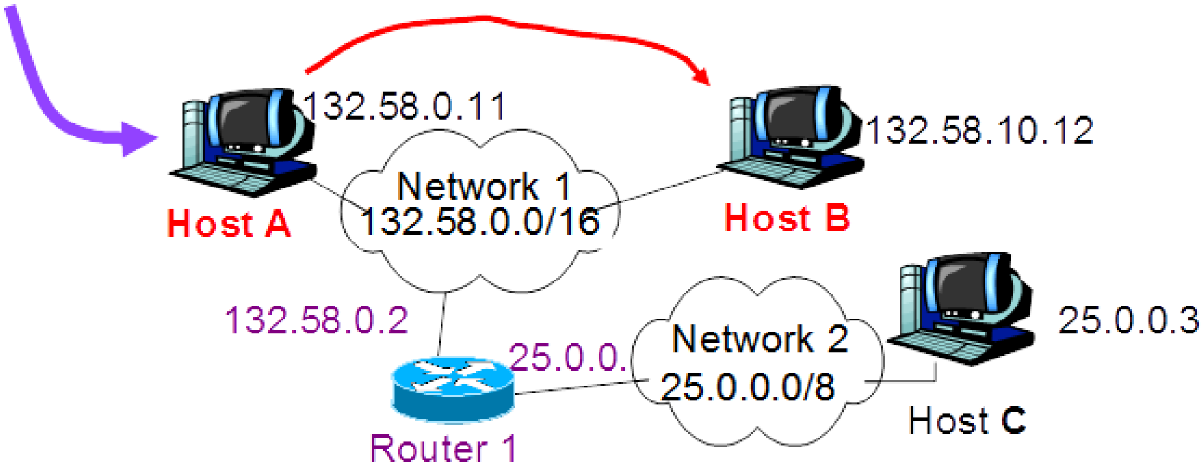
Network – Link layer addressing

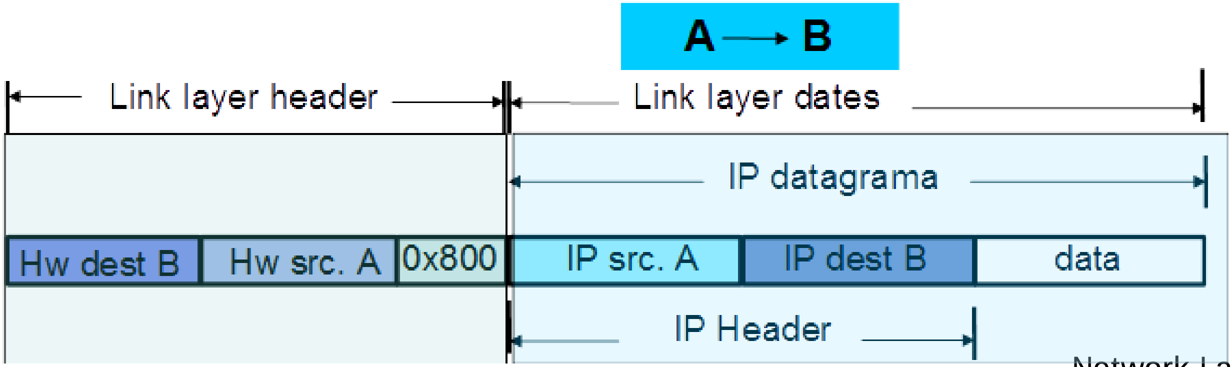
- Computers are identified on the Internet by its IP address
- Destination address in an IP datagram always identify the end user, the consumer of this IP datagram
- Link protocols handle other addresses, physical addresses to identify each node at each end of the link
- We need to use two types of addresses
  - IP addresses in the header of the datagram
  - Physical addresses in the frame header

Forwarding table: same network

Forwarding table of *host* **A**

Dest. Network	Netmask	Rute	Interface
132.58.0.0	255.255.0.0	0.0.0.0	132.58.0.11
0.0.0.0	0.0.0.0	132.58.0.2	132.58.0.11

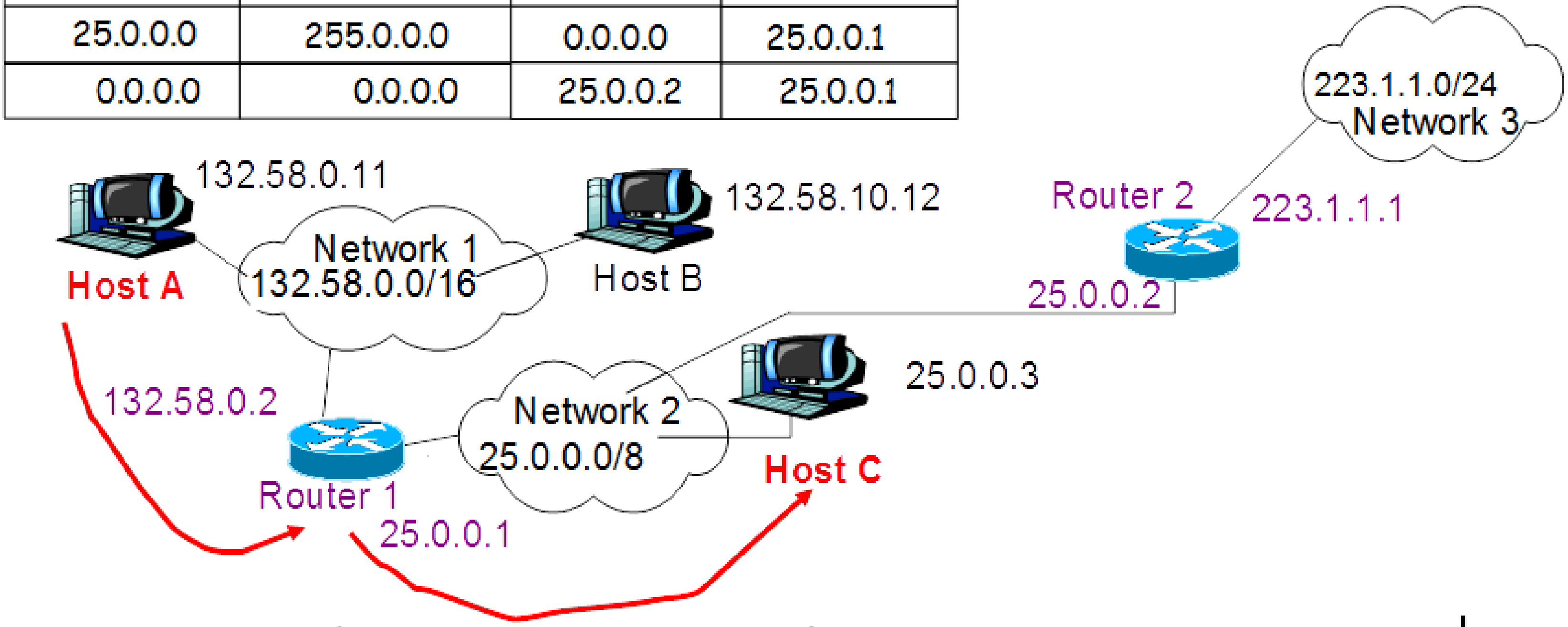




Forwarding table: different network

Forwarding table of *router* 1

Dest. Network	Netmask	Rute	Interface
132.58.0.0	255.255.0.0	0.0.0.0	132.58.0.2
25.0.0.0	255.0.0.0	0.0.0.0	25.0.0.1
0.0.0.0	0.0.0.0	25.0.0.2	25.0.0.1



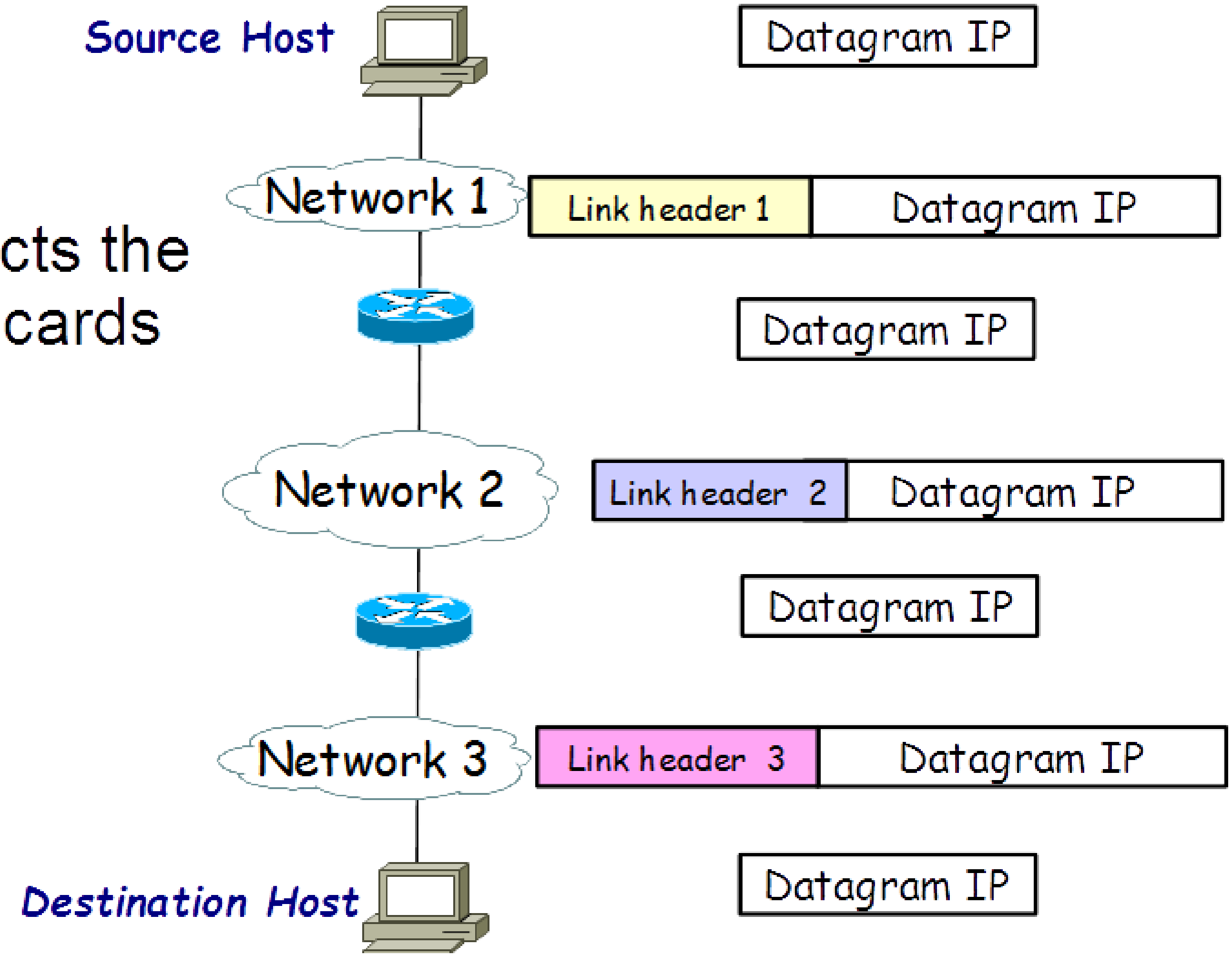
Link layer header ⟶ | Link layer dates

IP header

Hw dst R1	Hw src. A	0x800	IP src. A	IP dest C	data
Hw dst C	Hw src.R1	0x800	IP src. A	IP dest C	data

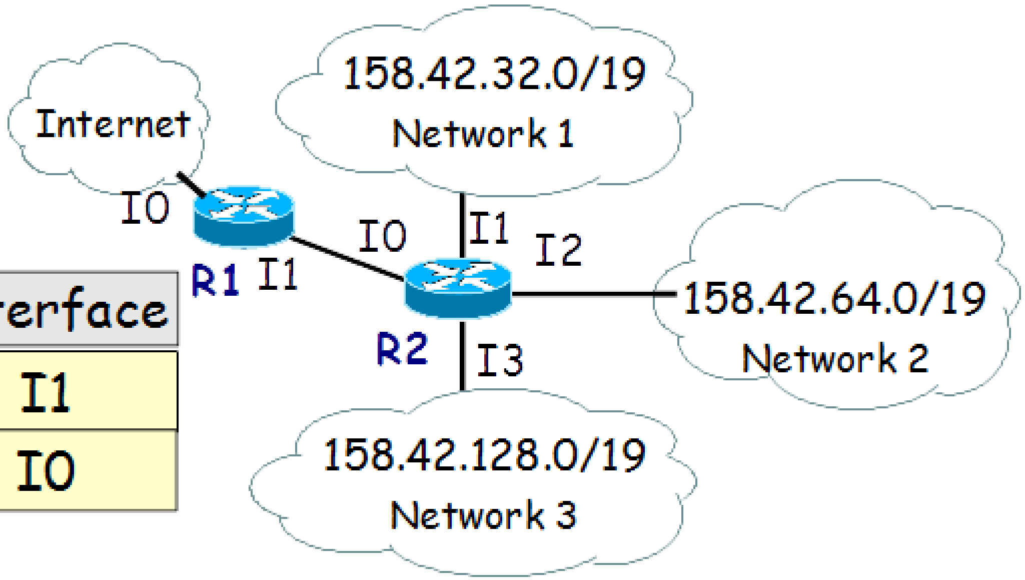
Transmission on Internet

Each router extracts the datagram and discards the frame



Forwarding table

- R1 only sees one network



Dest. Network	Netmask	Rute	Interface
158.42.0.0	255.255.0.0	R2	I1
0.0.0.0	0.0.0.0	Internet	I0

- R2 sees the three networks:

Dest. Network	Netmask	Rute	Interface
158.42.32.0	255.255.224.0	0.0.0.0	I1
158.42.64.0	255.255.224.0	0.0.0.0	I2
158.42.128.0	255.255.224.0	0.0.0.0	I3
0.0.0.0	0.0.0.0	R1	I0

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP

4.4 routing algorithms

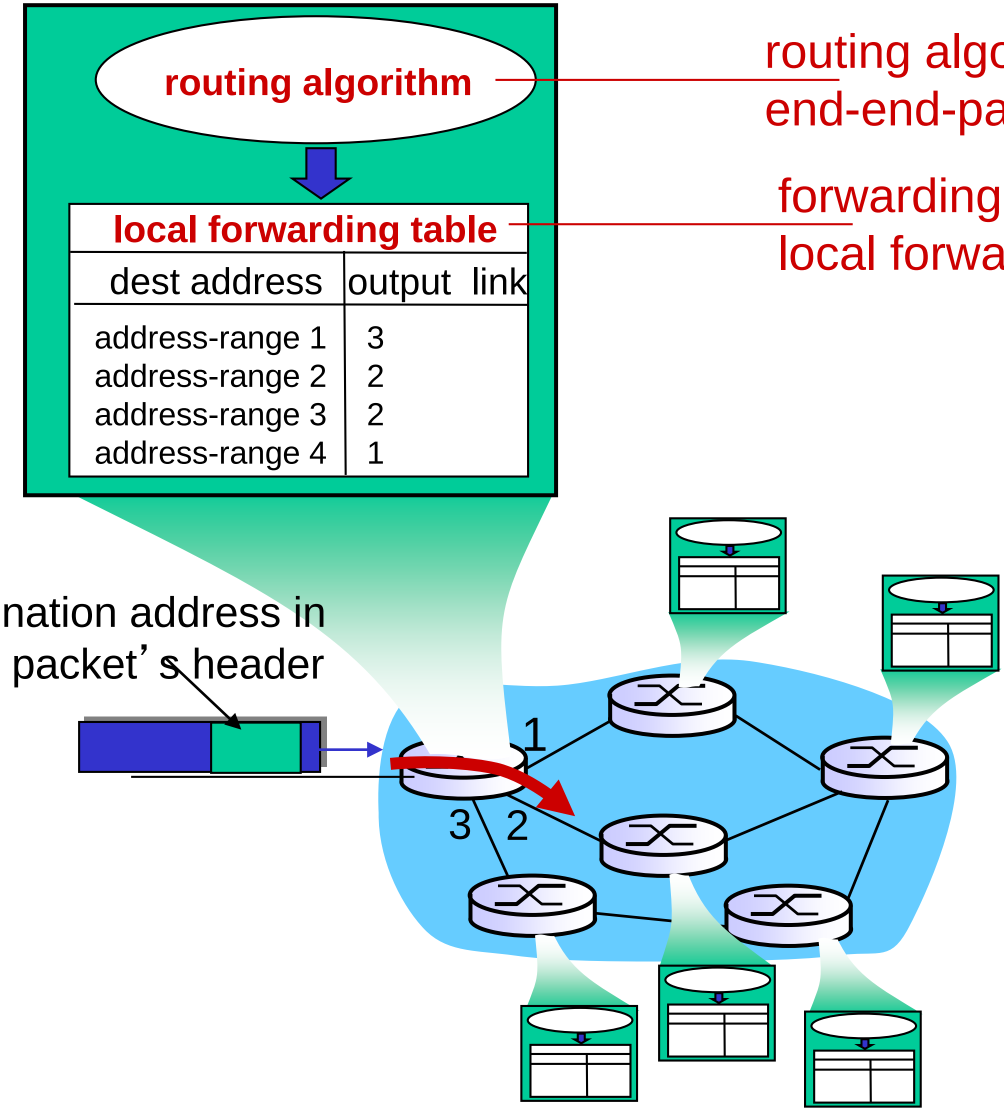
- distance vector
- link state
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

4.6 IPv6

Interplay between routing, forwarding



local forwarding table

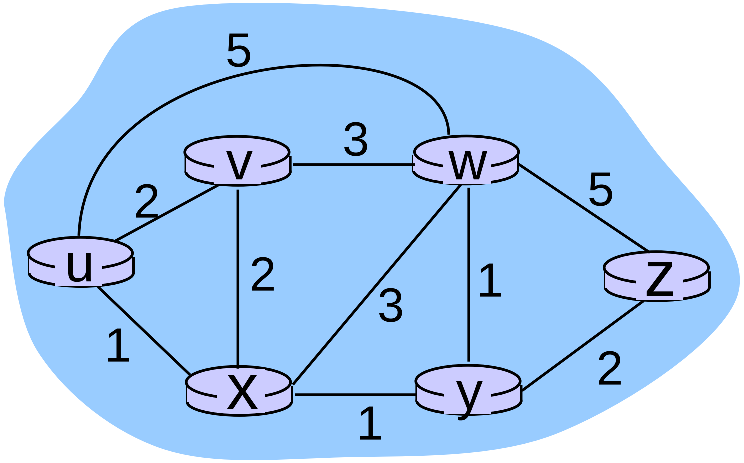
dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

IP destination address in arriving packet’s header

Graph abstraction



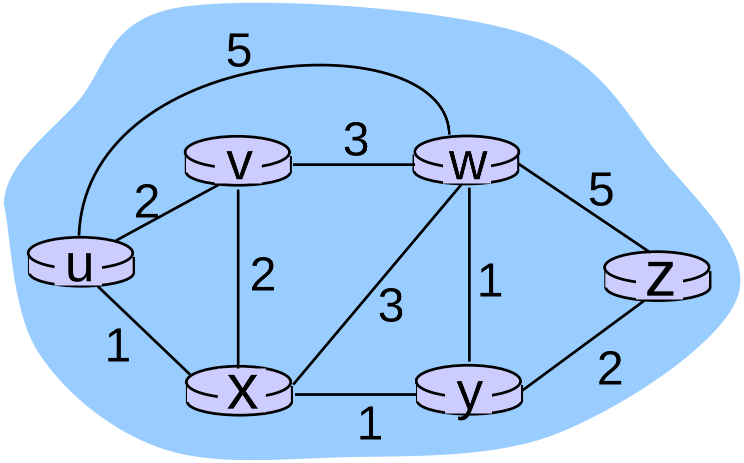
graph: G = (N,E)

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 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" 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 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP

4.4 routing algorithms

- distance vector
- link state
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

4.6 IPv6

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

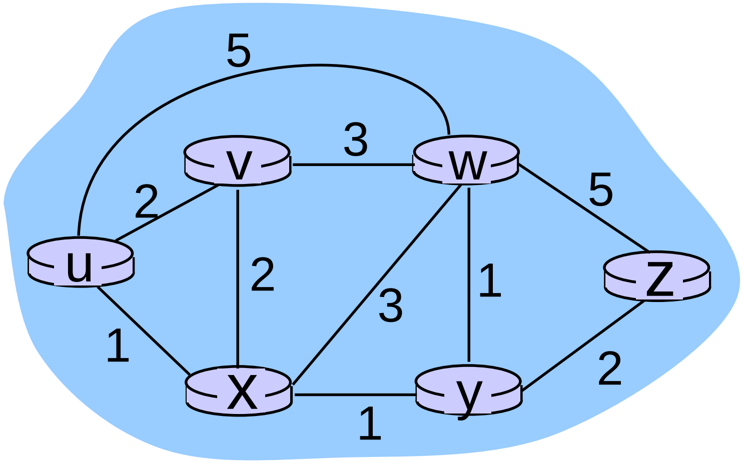
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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

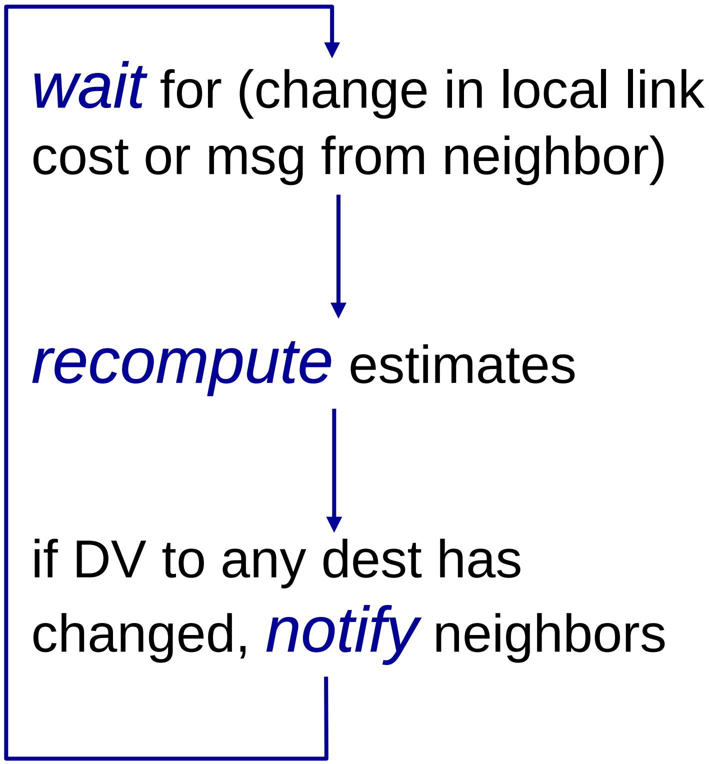
iterative, asynchronous:
each local iteration caused by:

- 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

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

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

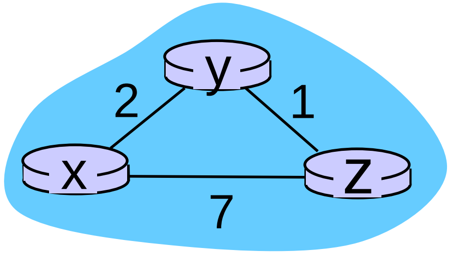
node y table

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

node z table

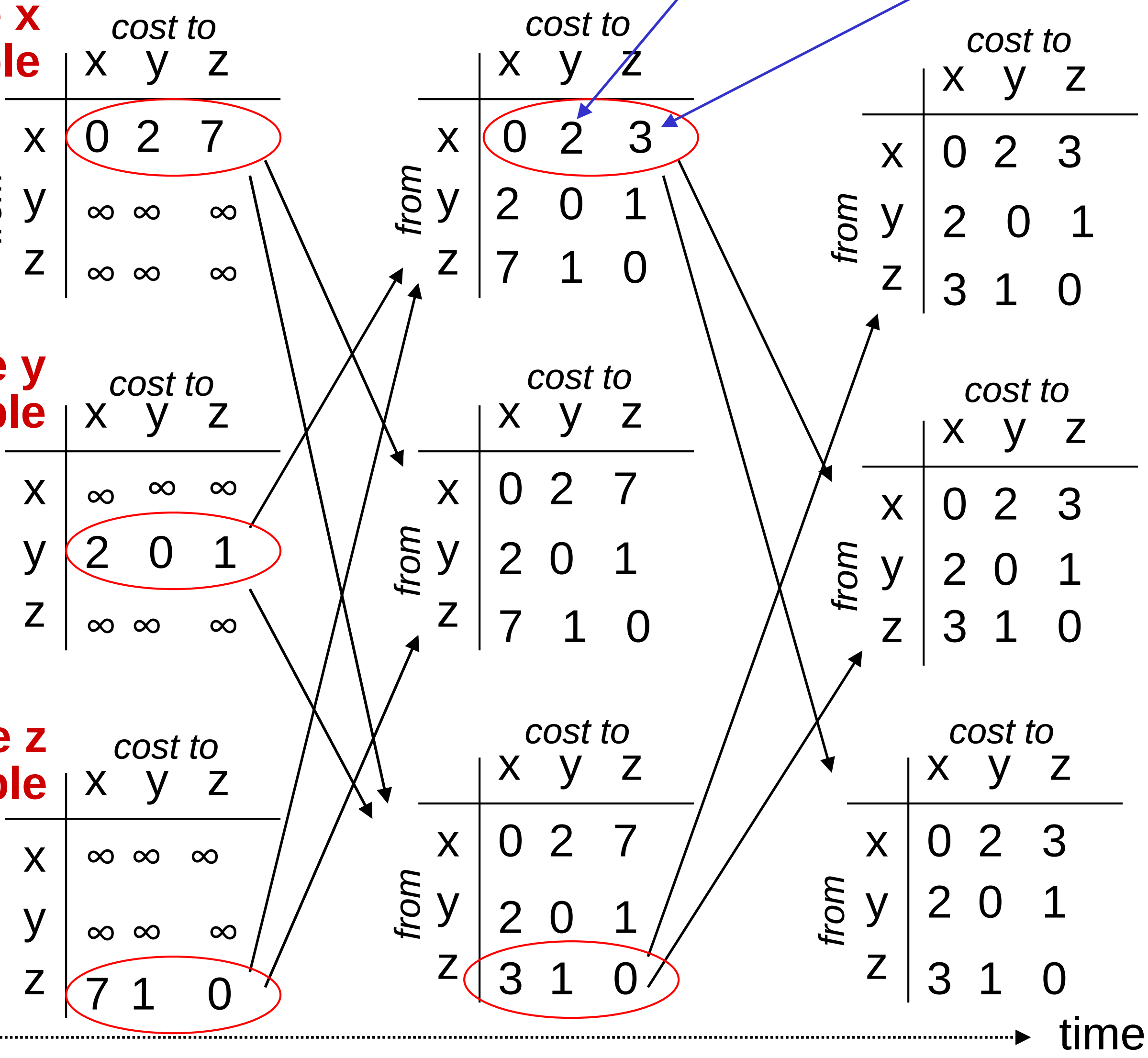
		cost to		
		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

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

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

node y table

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

node z table

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

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

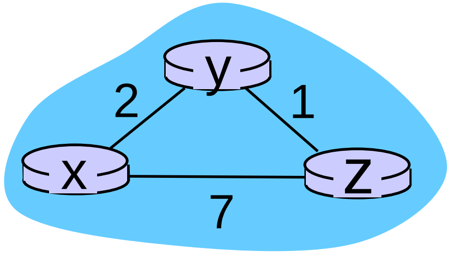
	cost to		
from	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

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

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

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

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



Example of distance table (Node B)

Available information in node B:

- neighbors: C i D
- Distance to its neighbors: $d_B(C) = 1$, $d_B(D) = 1$

Distance from node B to node A

$$d_B(A) = \min\{c(B,C) + d_C(A),\ c(B,D) + d_D(A)\}$$
$$= \min\{1+3,\ 1+1\} = 2$$

Distance Vectors of nodes C and D

C	
A	3
B	1
D	2
E	1

$d_B(C) = 1$

D	
A	1
B	1
C	2
E	1

$d_B(D) = 1$

Distance Table of node B

Cost to destination via (by way of)

Destination

$D_B()$	C	D
A	4	2
C	1	3
D	3	1
E	2	2

Example of forwarding table (Node B)

Cost to destination via (by way of)

	$D_B()$	C	D
Destination	A	4	2
	C	1	3
	D	3	1
	E	2	2

Destination	Next hop
A	D
C	C
D	D
E	C

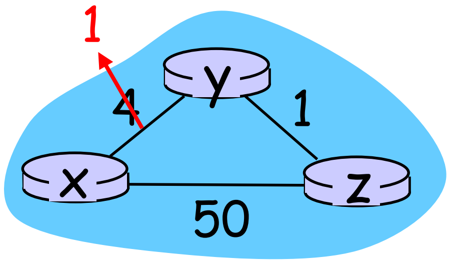
Distance Table → **Forwarding Table**

The neighbor node that provides the least cost path is the next hop in the forwarding table

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



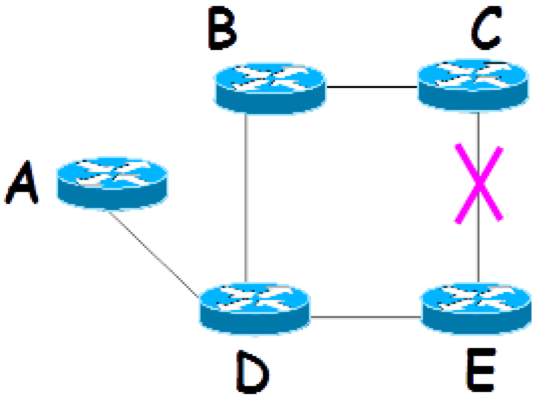
"good news travels fast"

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*.

Example link cost changes



Cost to destination via:

	$D_E()$	C	D
Destination	A	4	[✓ 2]
	B	[✓ 2]	2
	C	[✓ 1]	3
	D	3	[✓ 1]

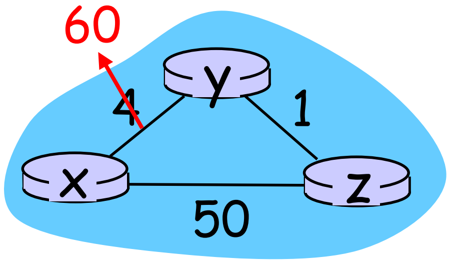
Cost to destination via:

	$D_E()$	C	D
Destination	A	inf	[✓ 2]
	B	inf	[✓ 2]
	C	inf	[✓ 3]
	E	inf	[✓ 1]

Distance vector: link cost changes

link cost changes:

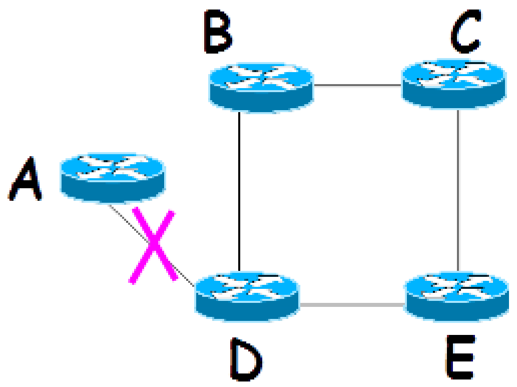
- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text



poisoned reverse:

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

Example link cost changes



Cost to destination via:

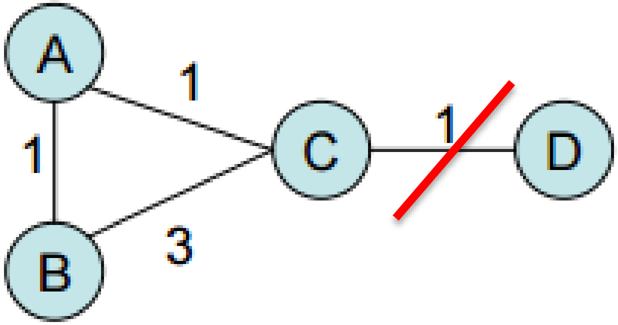
Destination	$D_D()$	A	B	E
	A	inf	3	3
	B	inf	1	3
	C	inf	2	2
	E	inf	3	1

- But B uses D in its rute!!! D – A link -> "count to infinity" problem!
- It creates a routing loop!!
- Solutions:
  - Limiting the diameter of the network: A route is considered unreachable if the hop count exceeds the maximum diameter.
  - Poisoned reverse with Split horizon
    - **Split horizon**: prohibiting a node from advertising a node back onto the interface from which it was learned.
    - **Poisoned reverse**: sets the number of cost to the unconnected node to a number that indicates "infinite"

Distance vector: poisoned reverse

poisoned reverse:

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



- C-D fails, and suppose the original optimal path from B to D is B-A-C-D, which means B will advertise C this optimal path from B's view.
- In this case, even with poisoned reverse, C can pick B as next hop for D.
- A loop is formed again.
- Poisoned reverse cannot prevent routing loops of a larger size of 2

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP

4.4 routing algorithms

- distance vector
- link state
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

4.6 IPv6

A Link-State Routing Algorithm

- Each node knows the distance to its neighbours (link state information)
- Each node distributes link state information to all nodes in the network
  - accomplished via "link state broadcast"
  - all nodes have same info
- With messages received, each node :
  - Builds the network graph
  - Computes least cost paths to all other nodes
  - Builds its own forwarding table

A Link-State Routing Algorithm

- Each router builds a link state package with the following information:
  - source node
  - sequence number
  - List of neighbours and distance
  - Time to Live (TTL)

- When link state packet is built?
  - Periodically
    - The interval can be hours
  - Significant events:
    - change in local link cost
    - Neighbour unreachable
    - Etc...

Link state example

- Link state packets

A	
# Seq	
B	4
E	5

B	
# Seq	
A	4
C	2
F	6

C	
# Seq	
B	2
D	3
E	1

D	
# Seq	
C	3
F	7

E	
# Seq	
A	5
C	1
F	8

F	
# Seq	
B	6
D	7
E	8

A Link-State Routing Algorithm

Notation for Dijkstra algorithm:

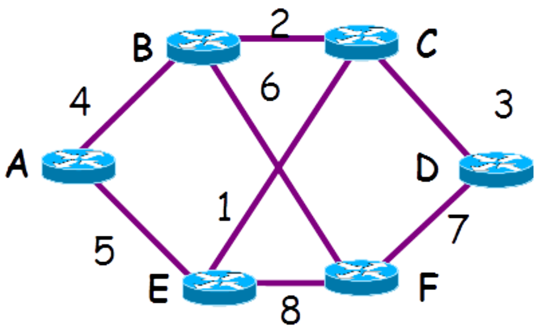
- c(x,y): link cost from node x to y; = ∞ if not direct neighbours
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra’s Algorithm

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

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'
```

Link state example



← Network graph

Source node A runs Dijsktra's Algorithm:

Iteration	N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	4, A	infinite	infinite	5, A	infinite
1	A, B		6, B	infinite	5, A	10, B
2	A, B, E		6, B	infinite		10, B
3	A, B, E, C			9, C		10, B
4	A, B, E, C, D					10, B
5	A, B, E, C, D, F					

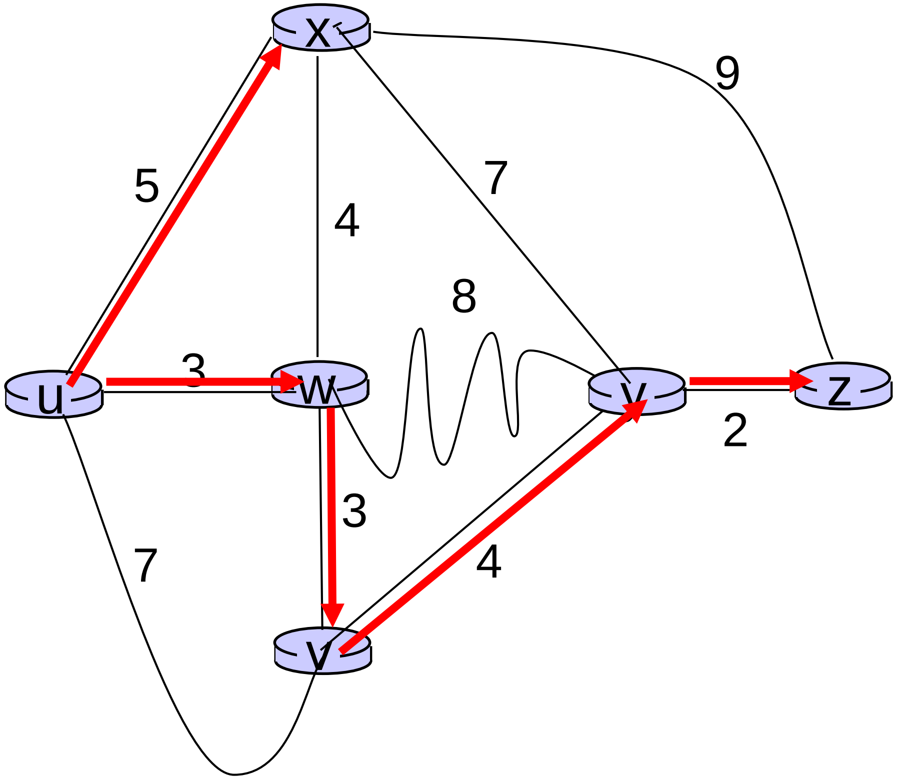
Link state example

Iteration	N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	4, A	infinite	infinite	5, A	infinite
1	A, B		6, B	infinite	5, A	10, B
2	A, B, E		6, B	infinite		10, B
3	A, B, E, C			9, C		10, B
4	A, B, E, C, D					10, B
5	A, B, E, C, D, F					

Forwarding Table

Destination	Next Hop
B	B
C	B
D	B
E	E
F	B

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)

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP

4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

4.6 IPv6

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

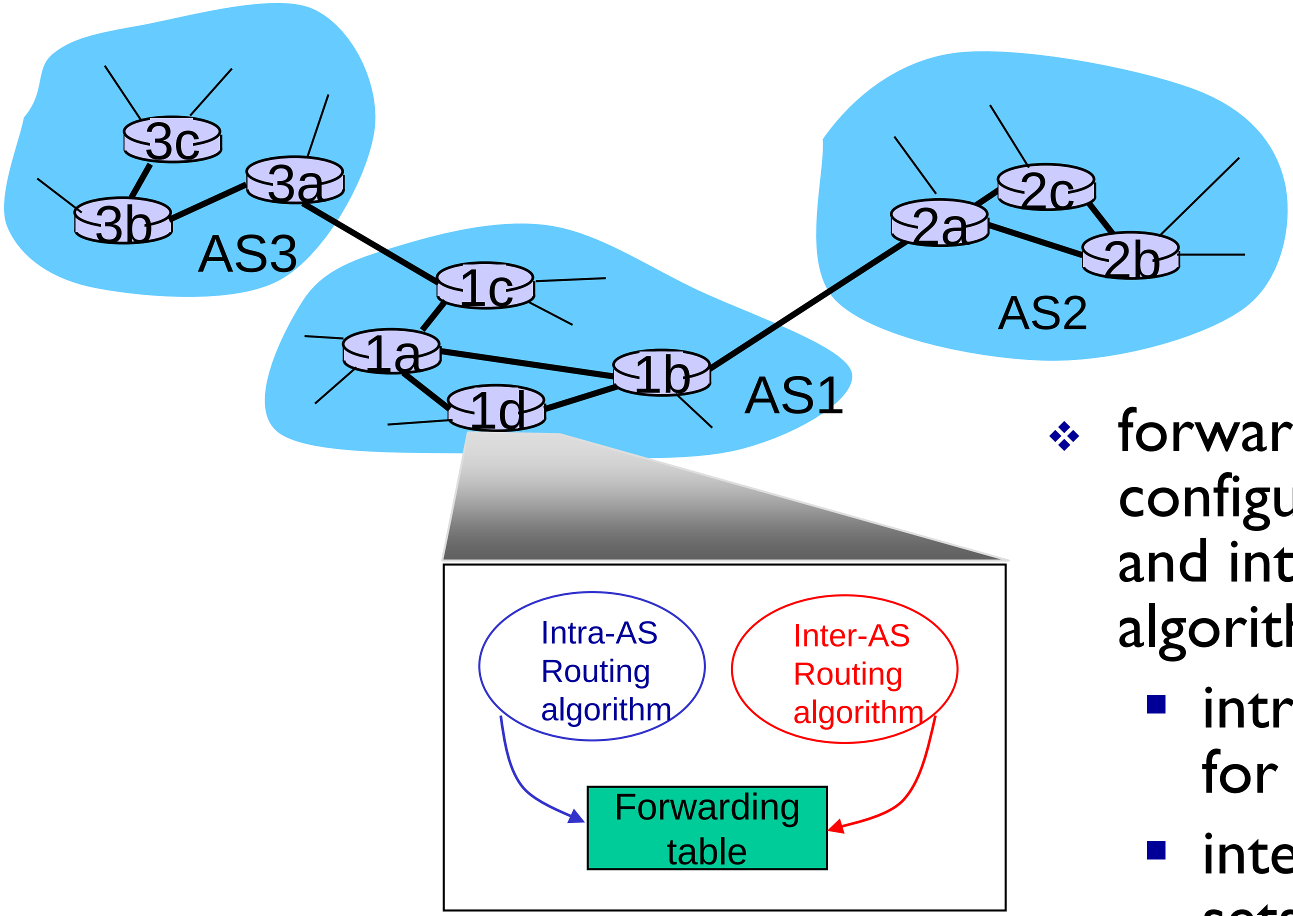
- collect routers into regions, "autonomous systems" (AS)
- Each AS within an ISP
  - ISP may consist of one or more ASes

- 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

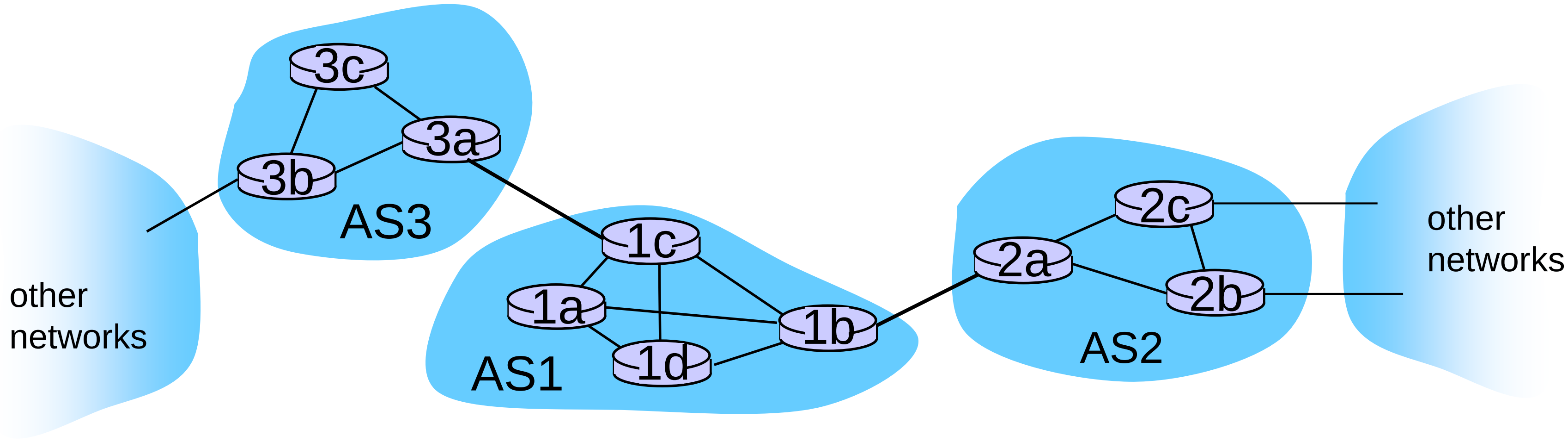
Inter-AS tasks

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

AS1 must:

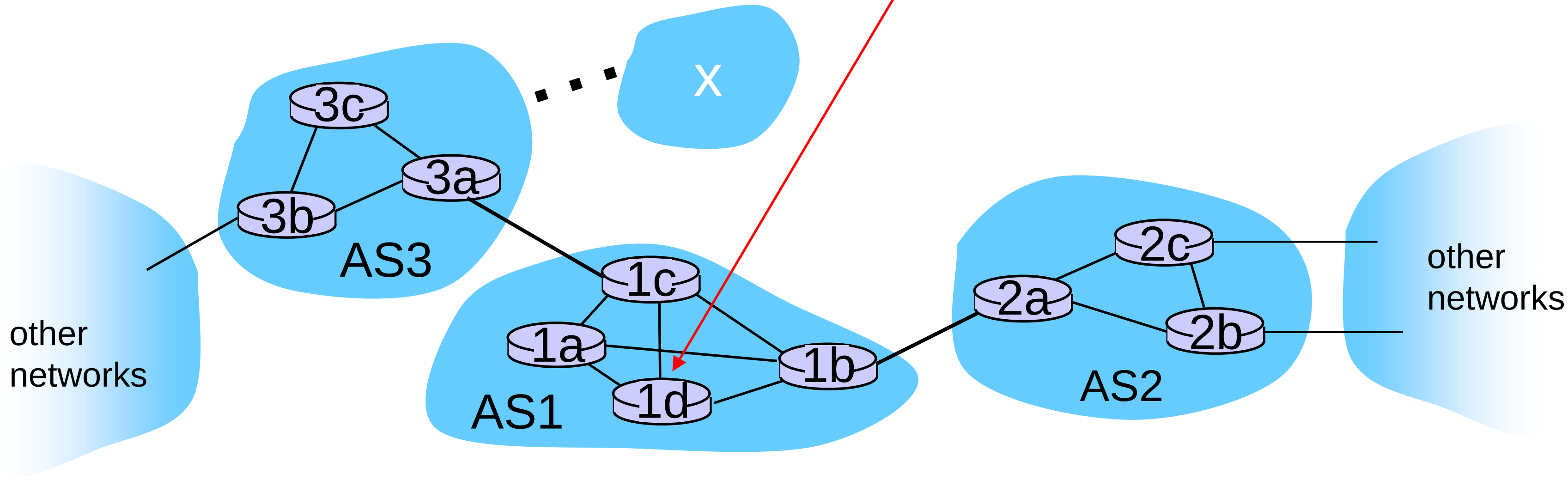
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



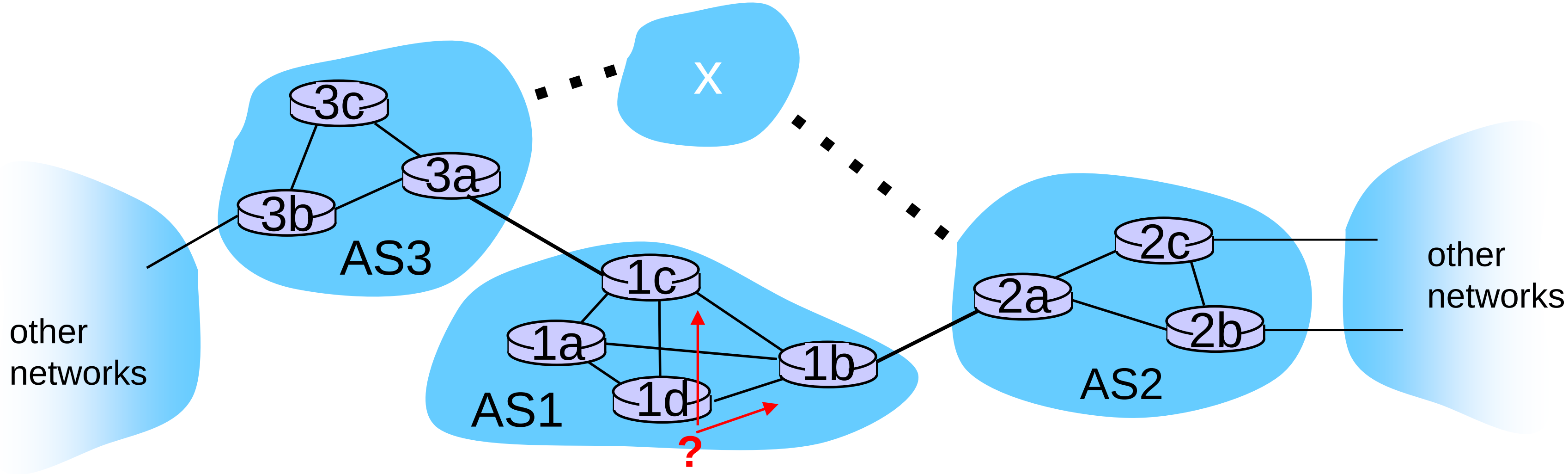
Example: setting forwarding table in router 1d

- 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)*



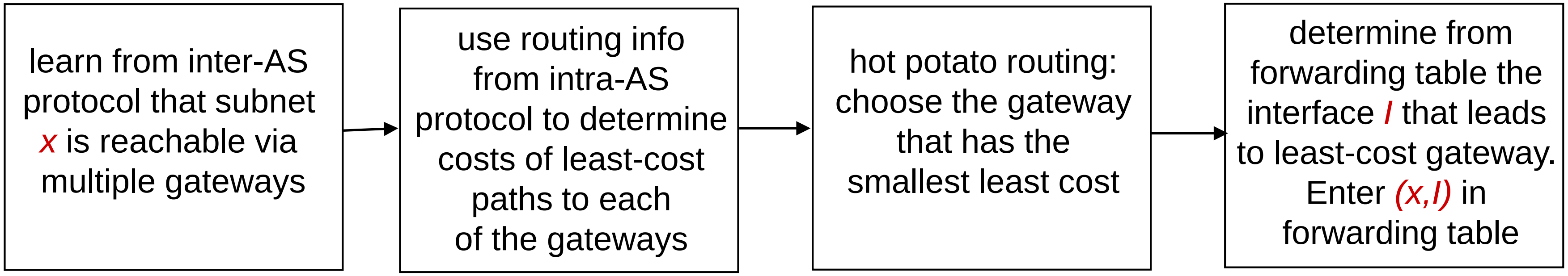
Example: choosing among multiple ASes

- 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.



learn from inter-AS protocol that subnet *x* is reachable via multiple gateways

use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways

hot potato routing: choose the gateway that has the smallest least cost

determine from forwarding table the interface *I* that leads to least-cost gateway. Enter *(x,I)* in forwarding table

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

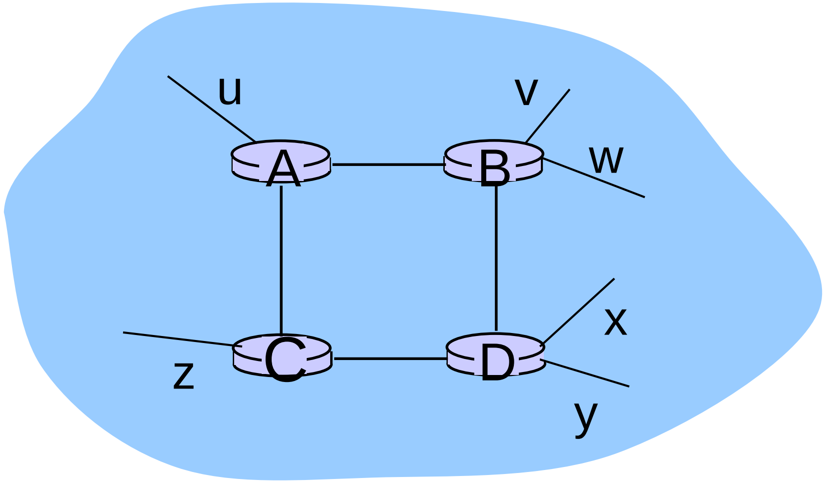
4.6 IPv6

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)

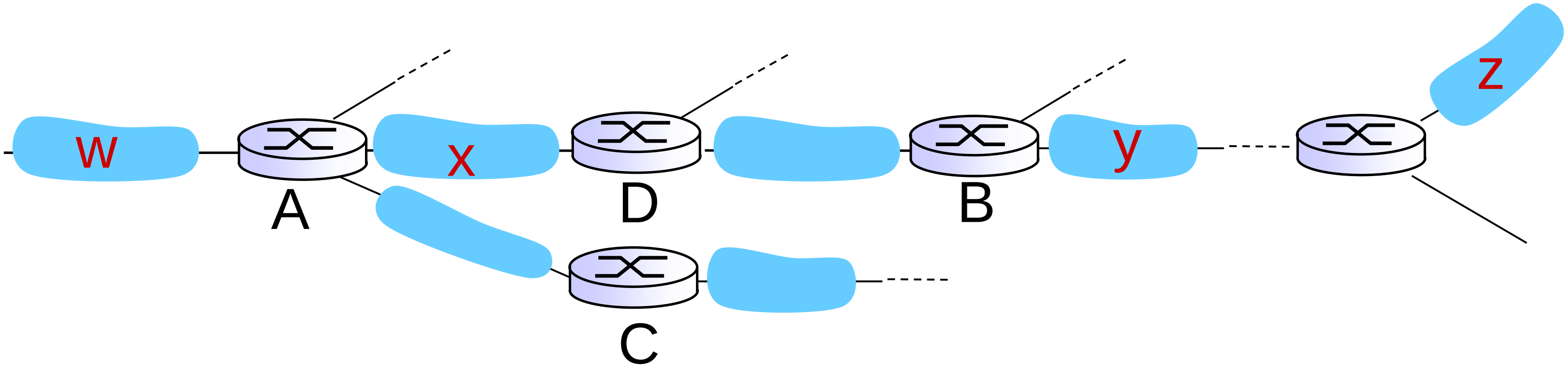
- included in BSD-UNIX distribution in 1982
- Uses UDP (port 520)
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets* *(in IP addressing sense)*



from router A to destination *subnets:*

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

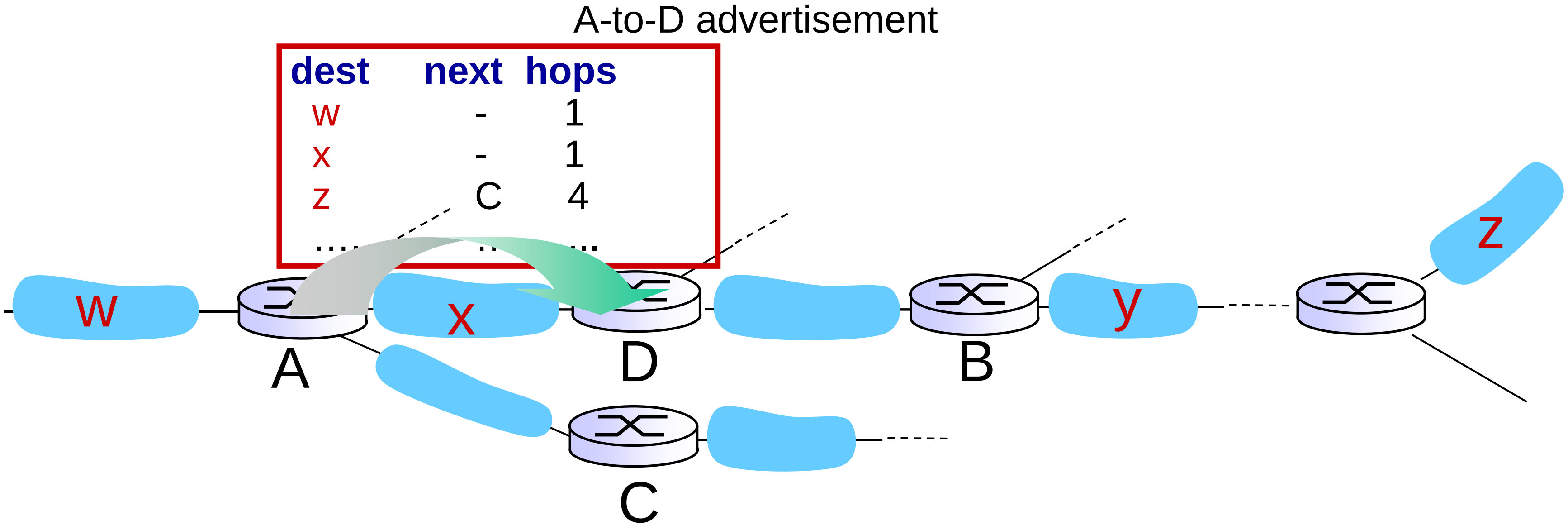
RIP: example



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
….	….	….

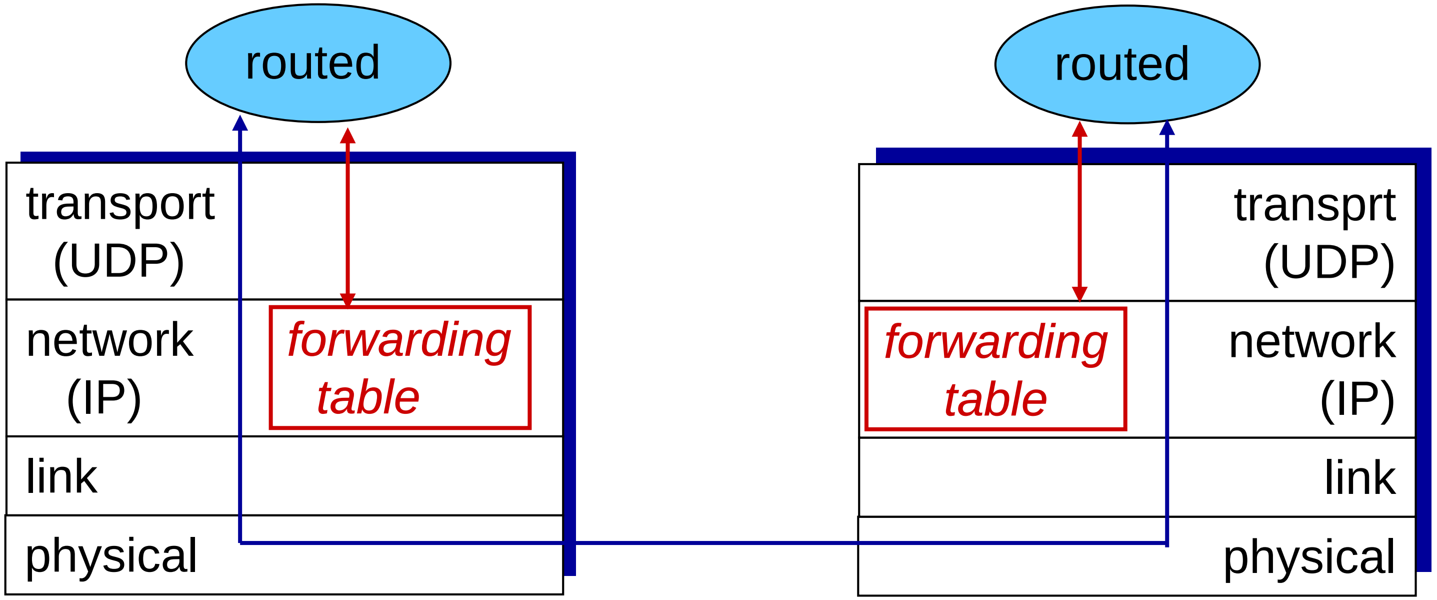
RIP: link failure, recovery

if no advertisement heard after 180 sec --> neighbour/link declared dead

- routes via neighbour invalidated
- new advertisements sent to neighbours
- neighbours 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)
- advertisements sent in UDP packets, periodically repeated



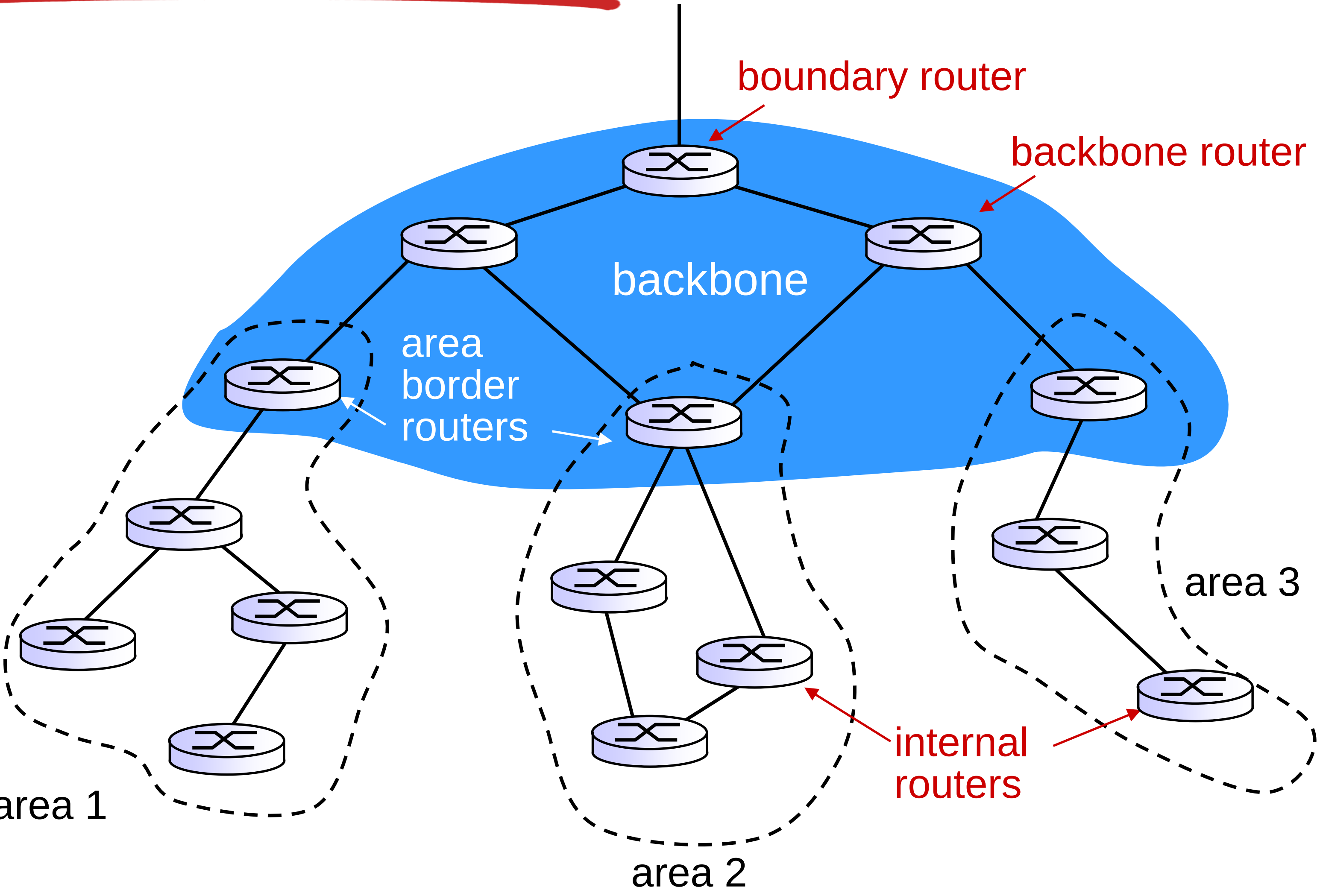
OSPF (Open Shortest Path First)

- "open": publicly available, RFC 2328
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbour
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
- the network administrator decides the criteria to define the cost, for example:
  - Cost 1, the least cost route will be that with fewer hops
  - Cost inversely proportional to the bandwidth of the link, which discourages the use of links with lower bandwidth

OSPF "advanced" features (not in RIP)

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort 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
  - Each AS divided into areas, each area can run its own OSPF routing algorithm

Hierarchical OSPF



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

- Different goals for different types routing:
  - Intra-AS: performance (optimal routes)
  - Inter-AS: reachability (loop-free paths to reach any destination that meets certain policies)
- Routing problems that difficult inter-AS:
  - Scale: the backbone routers may have to handle more than 150,000 entries
  - Administrative autonomy of AS: Use of different metrics
  - Confidence in the remaining AS

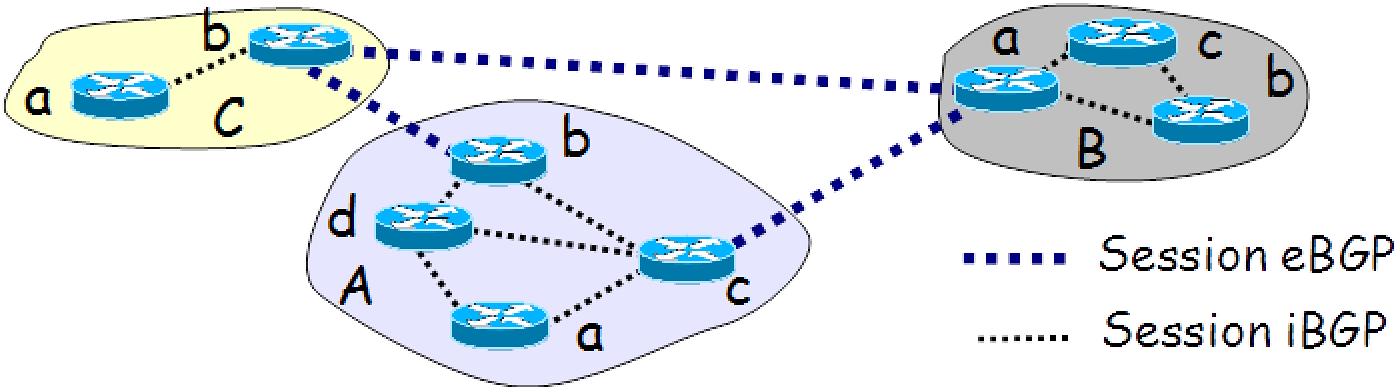
Internet inter-AS routing: BGP

- 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.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

BGP basics

- It is a very complex protocol

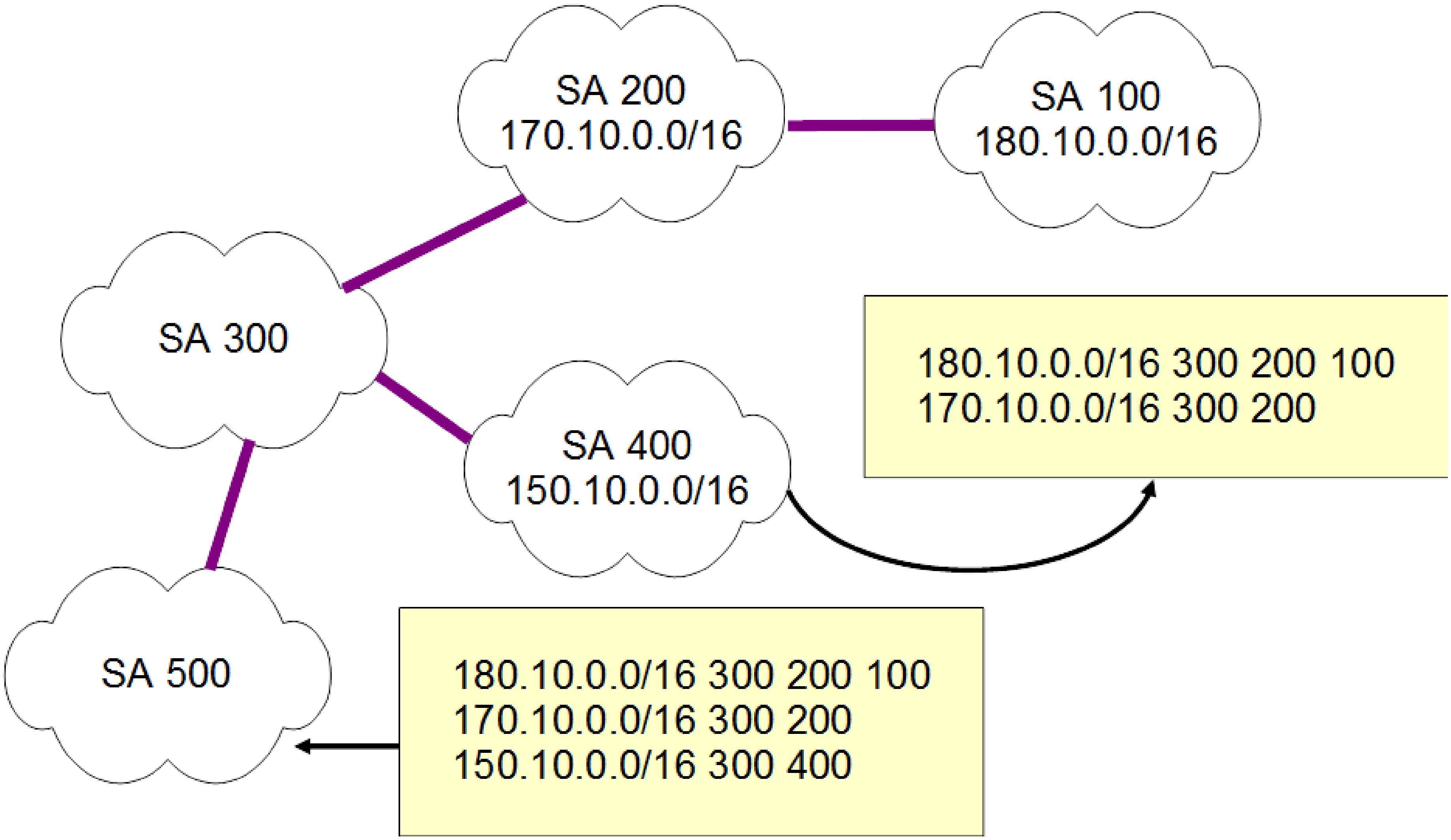
- Uses TCP port 179 (BGP sessions)
  - among routers of different AS and routers in the same AS
  - routers do not need to share a direct physical link



BGP basics

- It is a path vector algorithm:
  - similar to distance vector algorithm but works with full paths
  - each SA has a unique identification number
  - each gateway informs about the sequence of AS through which has to pass to reach a destination network
- Working with full paths prevents routing loops

BGP example



180.10.0.0/16 300 200 100
170.10.0.0/16 300 200

180.10.0.0/16 300 200 100
170.10.0.0/16 300 200
150.10.0.0/16 300 400

Tasks of a BGP router

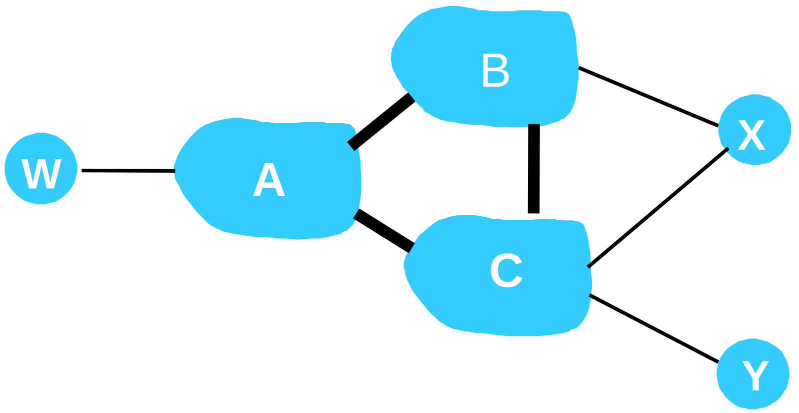
- Reception of route advertisements from directly connected neighbour routers
- Route selection
  - router may learn about more than one route to destination AS, selects route based on:
    1. local policy decision
       - e.g., never route through AS x
       - *policy-based* routing
    2. shortest PATH
       - e.g., AS2 AS17 to 138.16.64/22

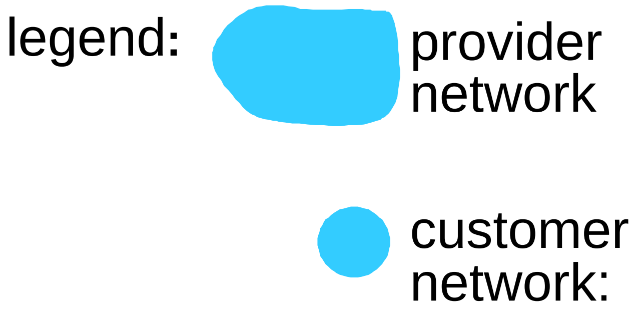
         AS3 AS131 AS201 to 138.16.64/22

select

- Send route advertisements to its neighbours

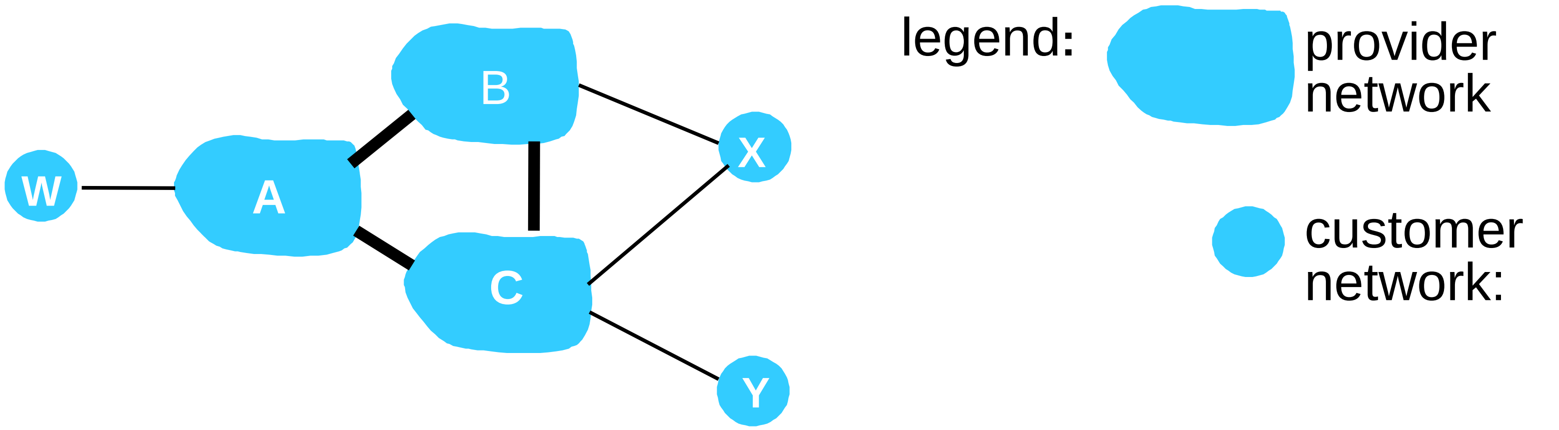
BGP routing policy





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

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!

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

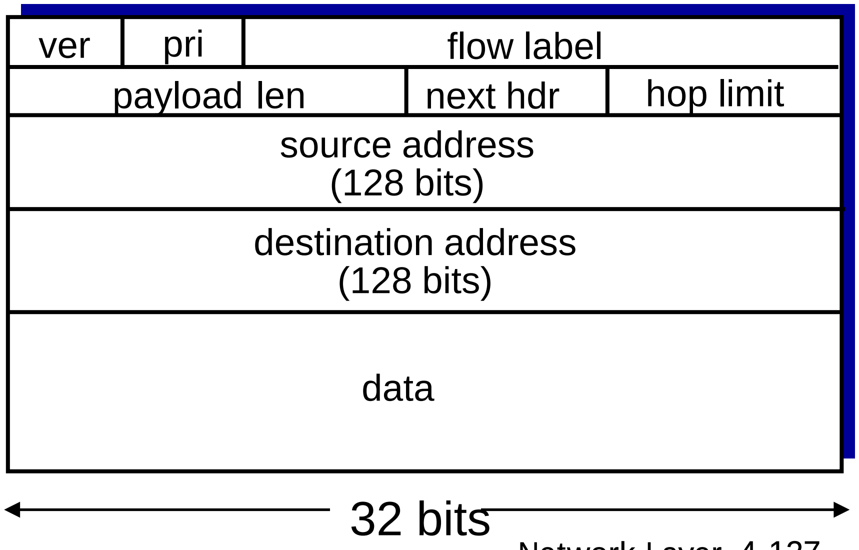
4.7 IPv6

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

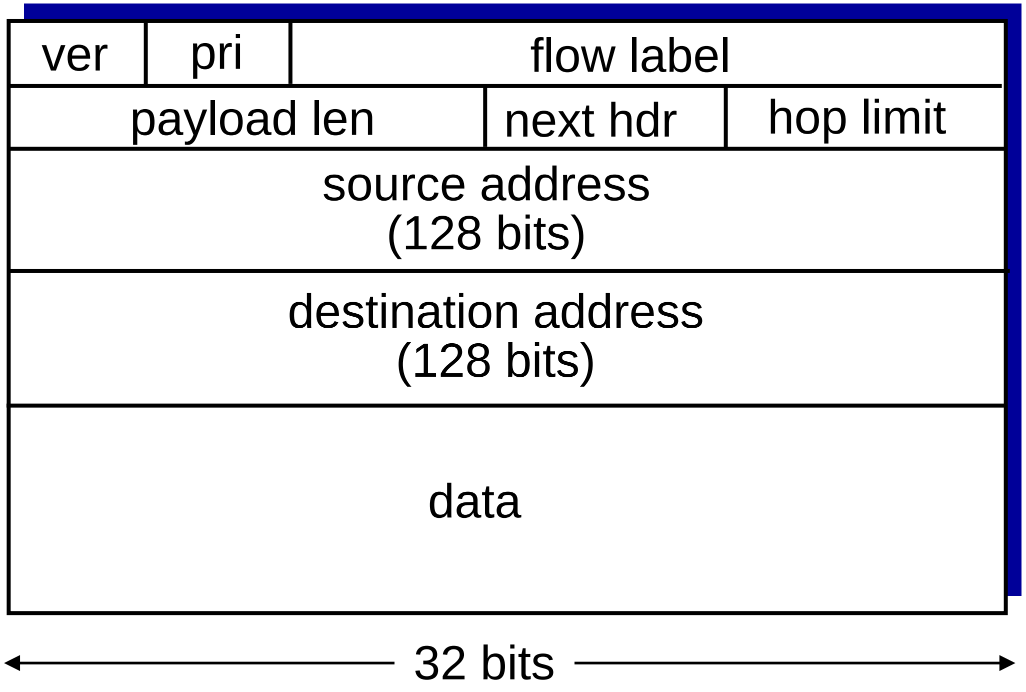
- *Address:*
  - Address of 128 bits
    - 6×10^{23} addresses per m^2
  - Hexadecimal notation separated by ":“
    - Example: 68E6: 8C64: FFFF: FFFF: 0: 1180: 796A: FFFF
    - Includes techniques:
      - zero compression:
        - sequence of zeros is replaced by a pair of “:"
        - Example: FF05: 0: 0: 0: 0: 0: 0: A8B3 -> FF05 :: A8B3
        - You can apply only once in an address
    - Includes decimal notation suffixes:
      - Example: 0: 0: 0: 0: 0: 0: 128.10.2.1 -> :: 128.10.2.1
  - Use CIDR notation: IPv6 Address / x
    - Example: FF05: 0: 0: 0: 0: 0: 0: A8B3 /60

IPv6 datagram format

- *Address:*
  - Type addresses:
    - Unicast: address of a computer
    - Multicast: address a group of computers (all)
    - Anycast: address a group of computers (anyone of the group)

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



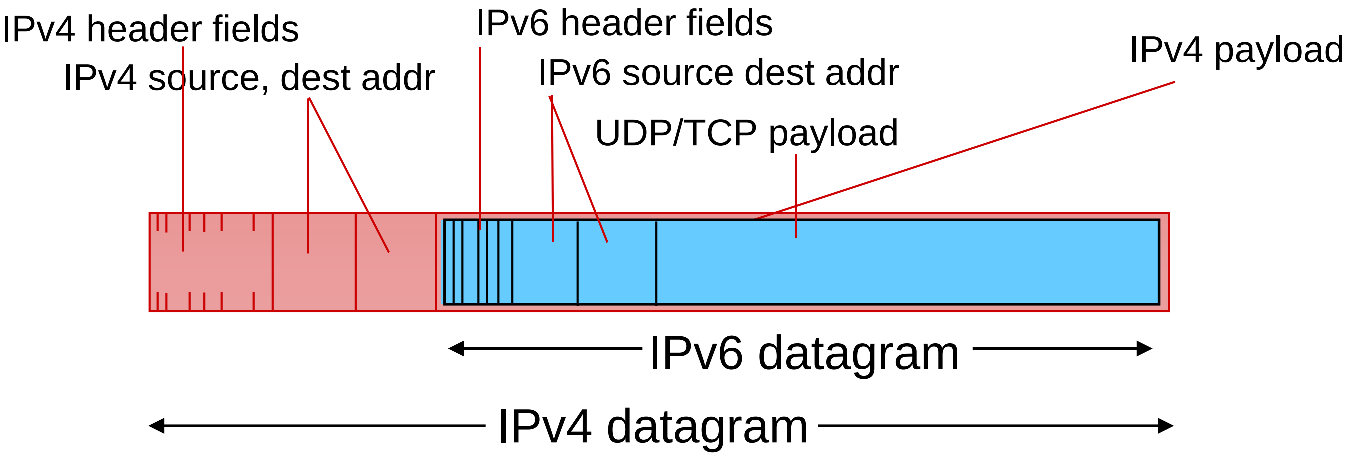
ver
pri
flow label
payload len
next hdr
hop limit
source address
(128 bits)
destination address
(128 bits)
data
32 bits

Other changes from IPv4

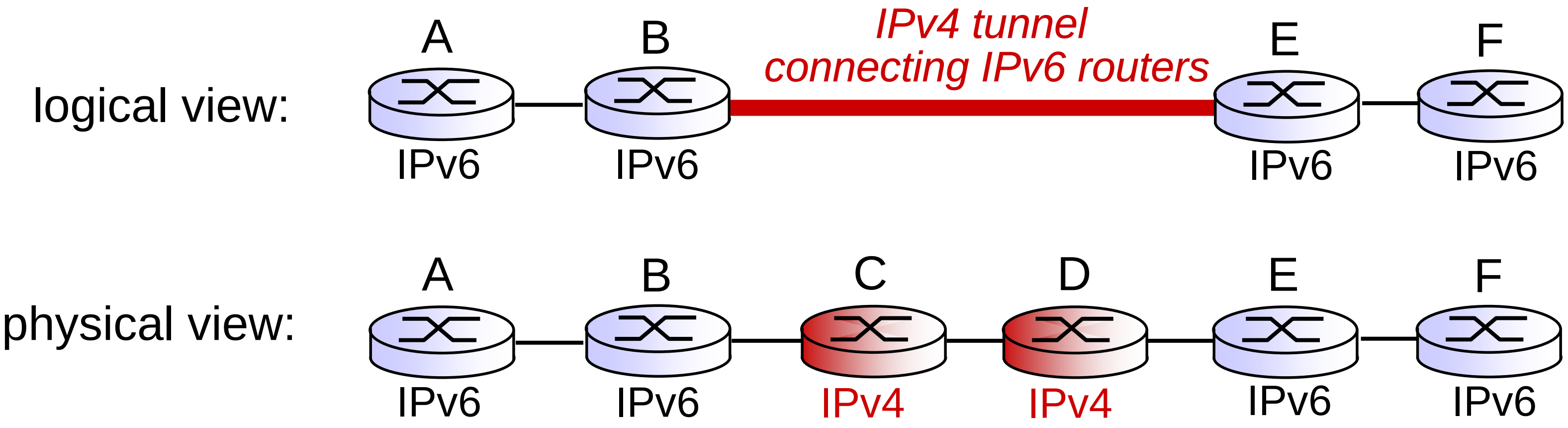
- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition from IPv4 to IPv6

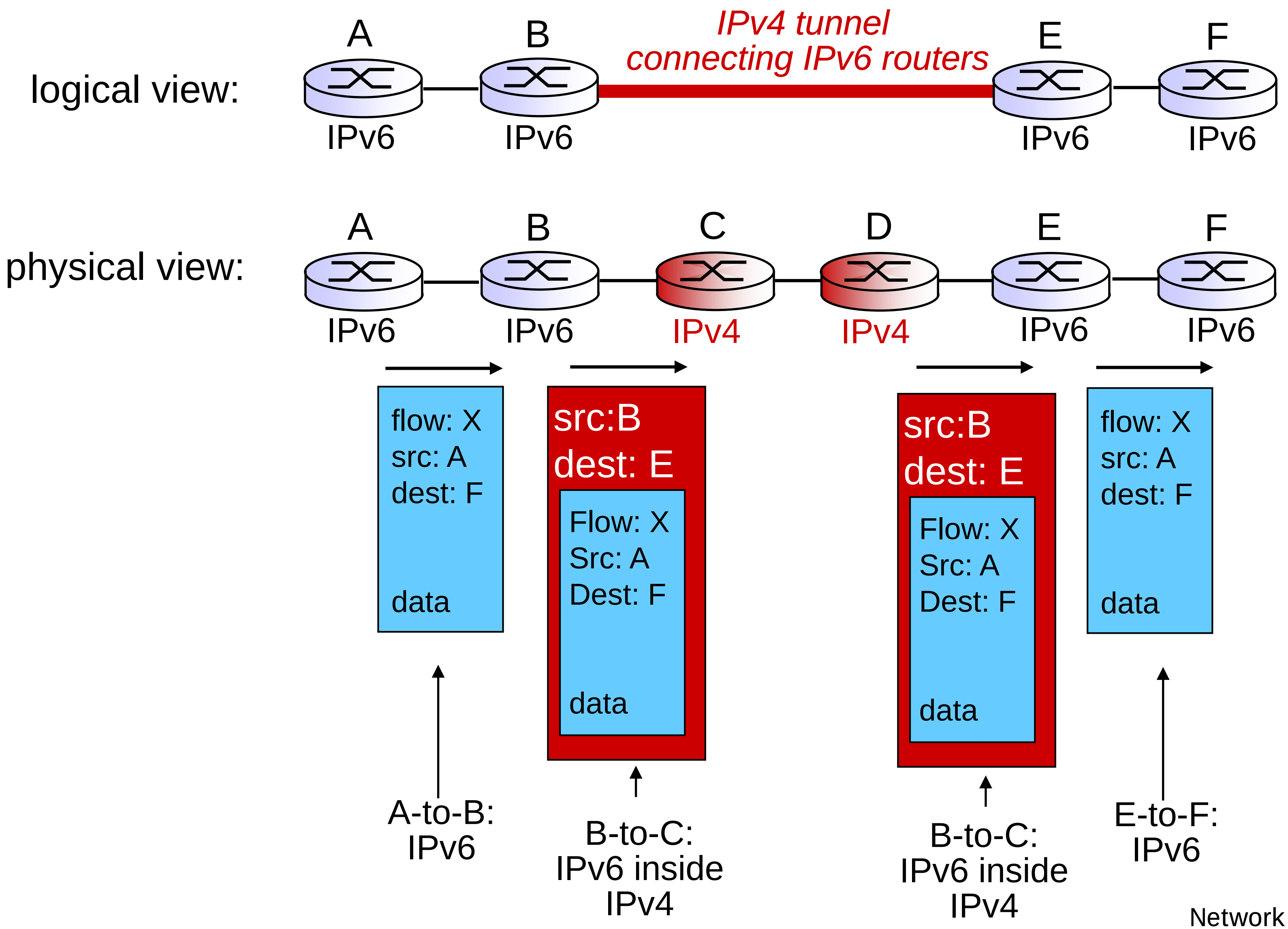
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


IPv4 tunnel connecting IPv6 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
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