Network Layer – Part I

Lecturer: Prof. Xiaoming Fu

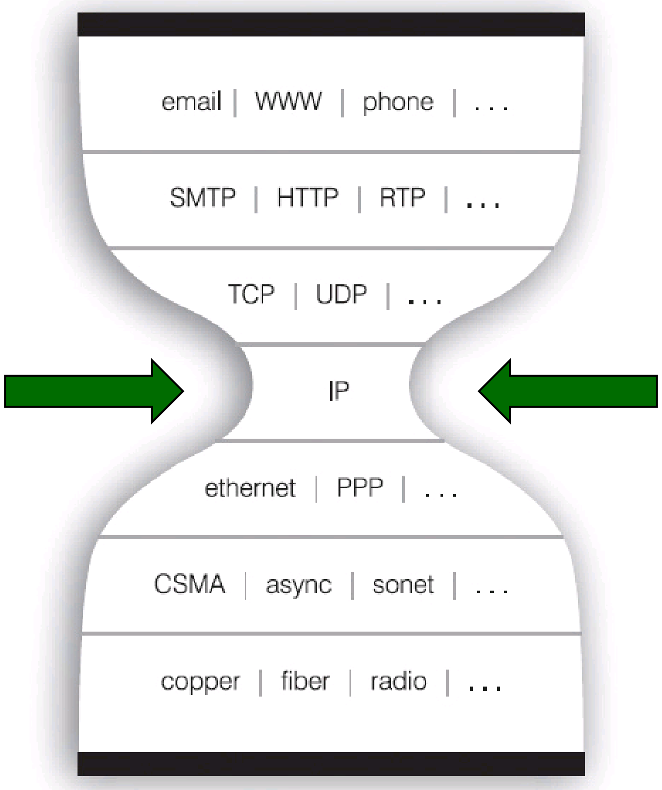
Assistants: Yachao Shao (MSc),

Fabian Wölk (MSc)

Network Layer

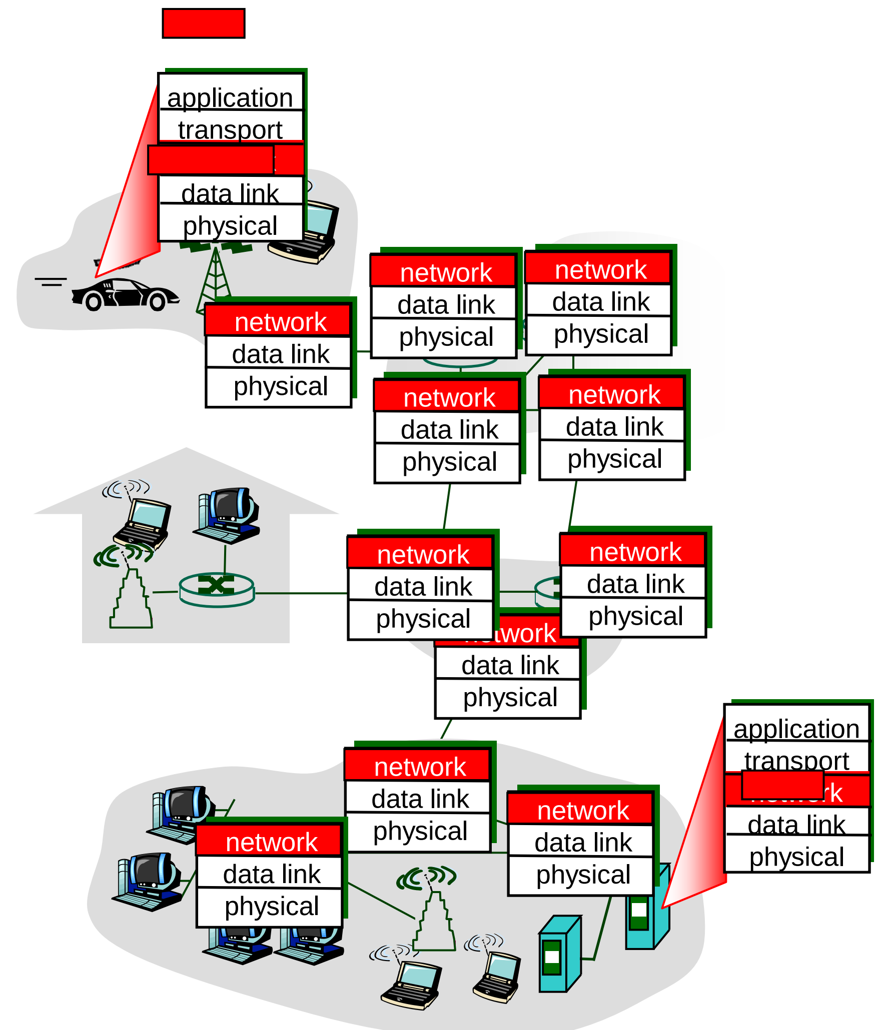
- **4.1 Introduction**
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

“Hourglass” architecture



Network layer

- Transports segments from sending to receiving host
- Sending side: encapsulates segments into datagrams
- 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

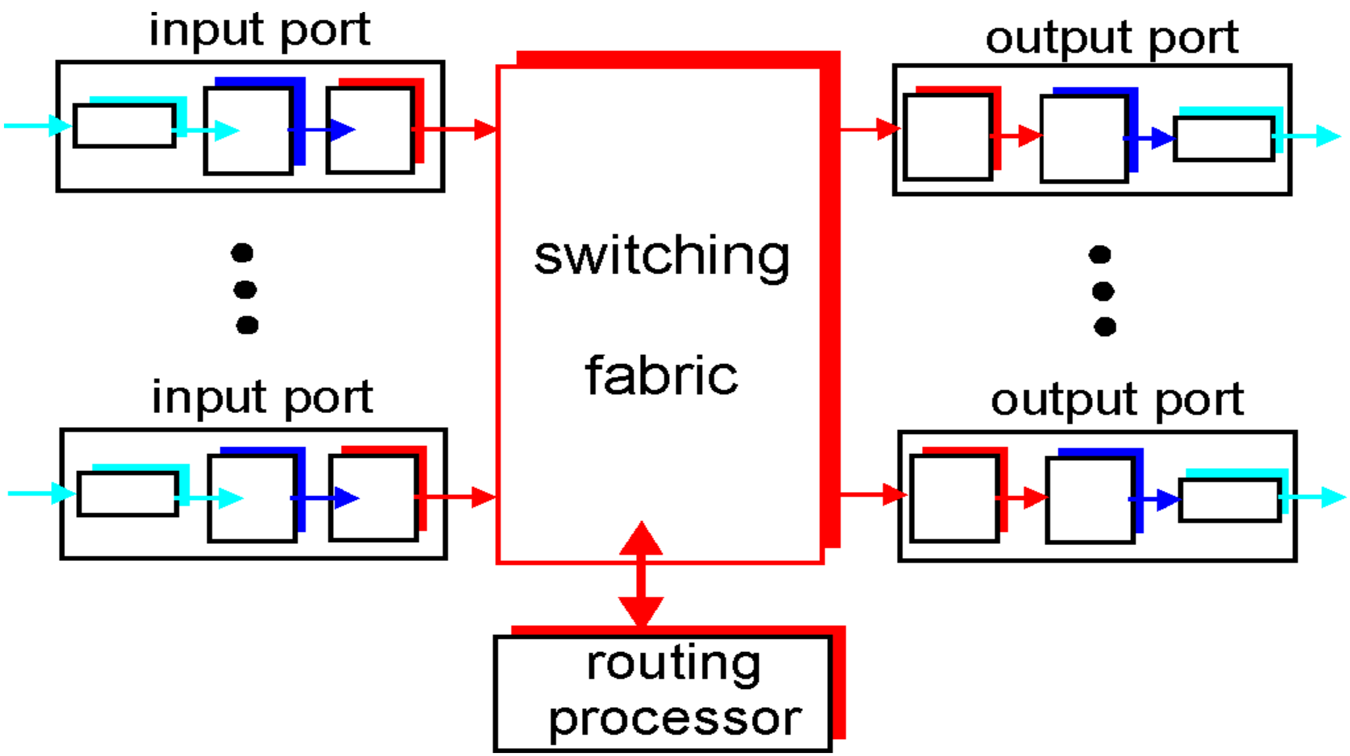
Network Layer

- 4.1 Introduction
- **4.2 What's inside a router**
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

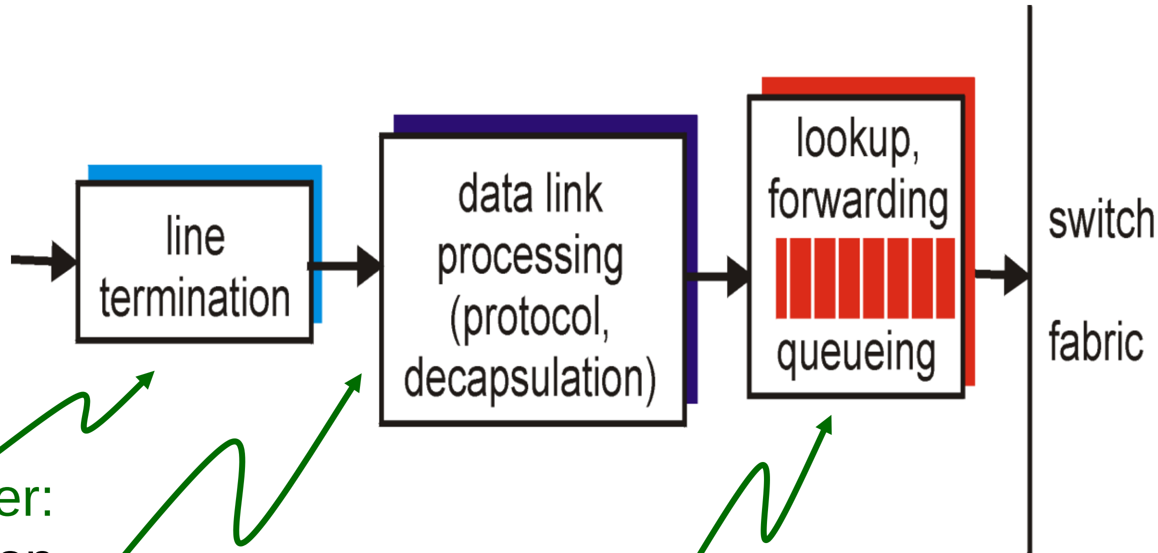
Router Architecture Overview

Two key router functions:

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



Input Port Functions



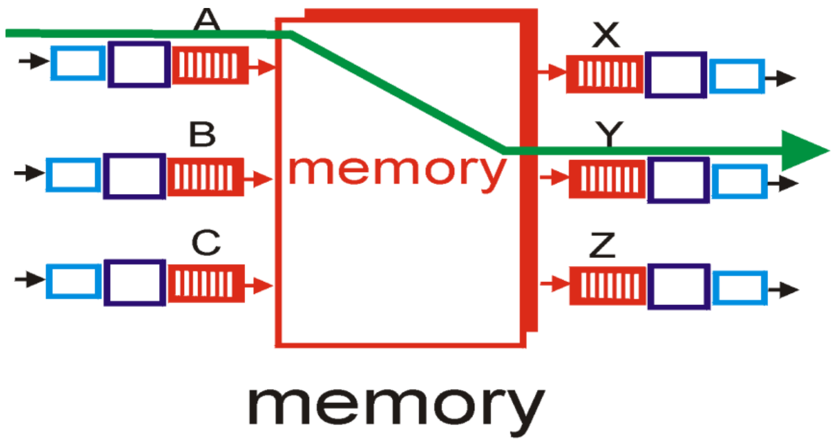
Physical layer:
bit-level reception

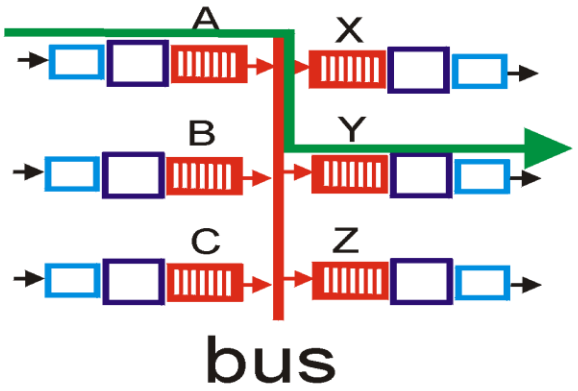
Data link layer:
e.g., Ethernet
see chapter 5

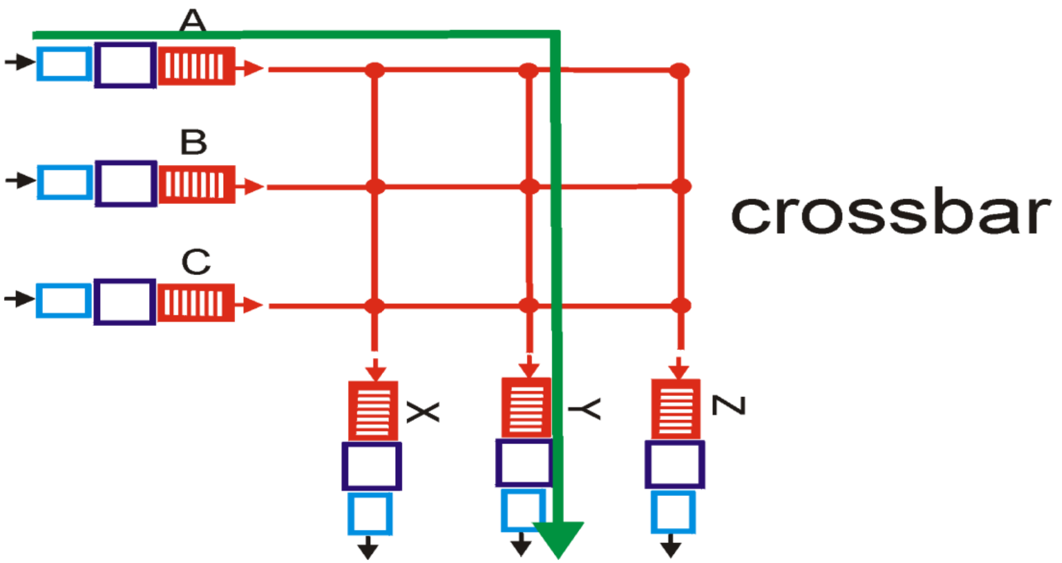
Decentralized switching*:*

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

Three types of switching fabrics



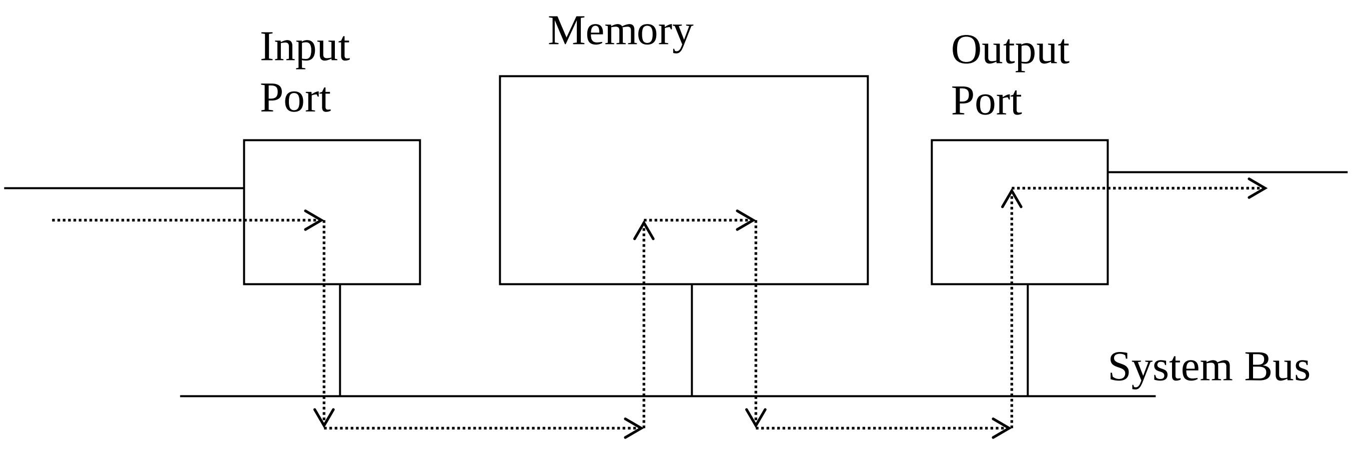




Switching Via Memory

First generation routers:

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



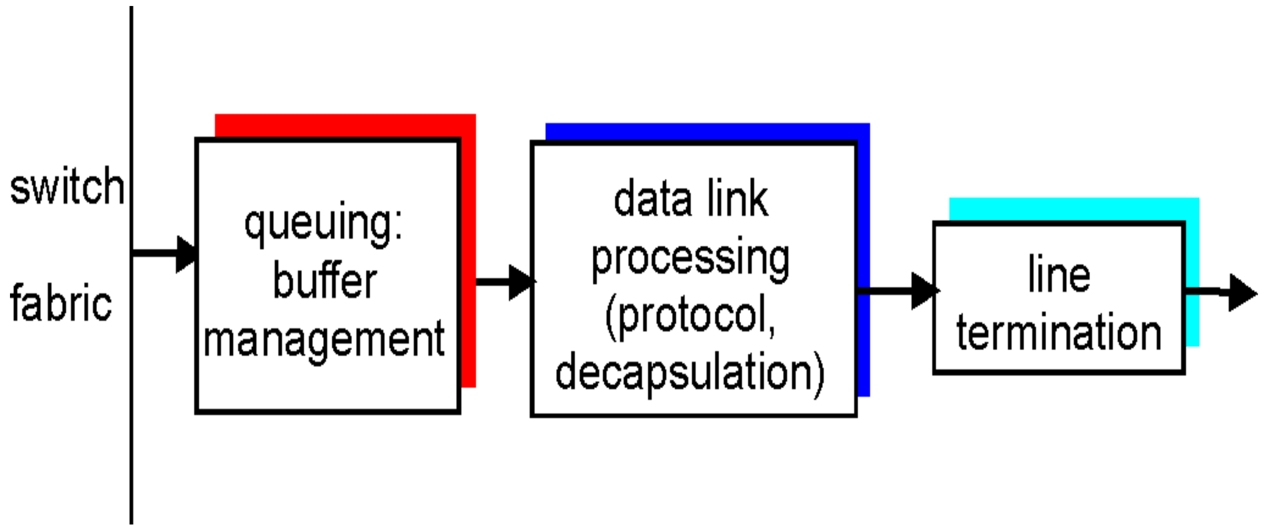
Switching Via a Bus

- Datagram from input port memory to output port memory via a shared bus
- Bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

Switching Via An Interconnection Network

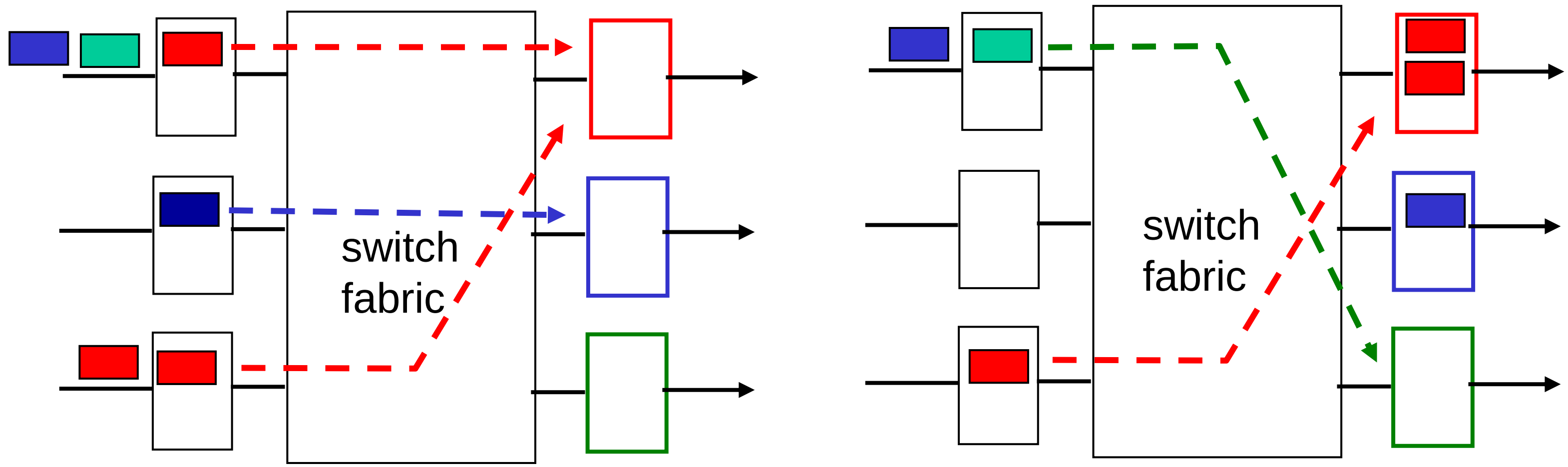
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor architectures
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

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



at *t*, packets more from input to output

one packet time later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

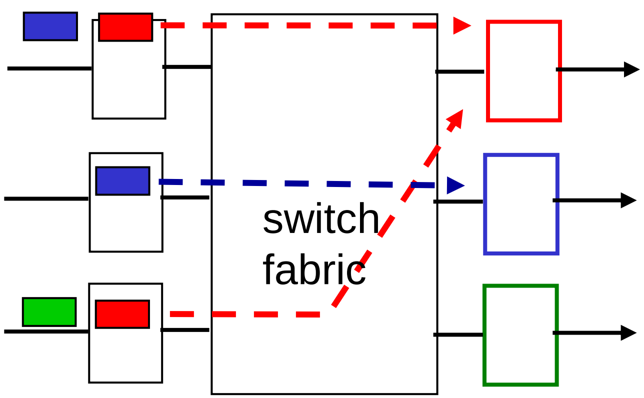
How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gps link: 2.5 Gbit buffer
- Recent recommendation: with *N* flows, buffering equal to

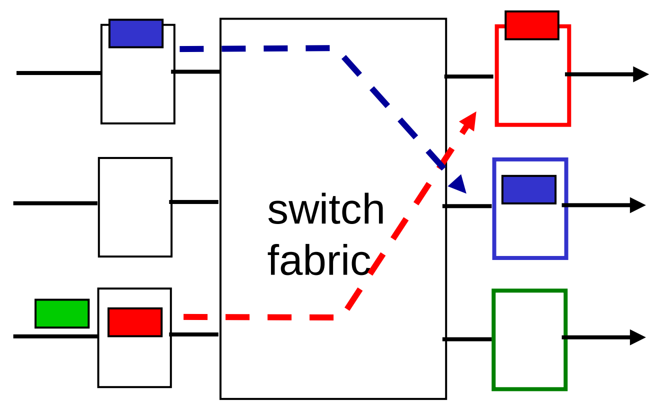
$$\frac{RTT \cdot C}{\sqrt{N}}$$

Input Port Queuing

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



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



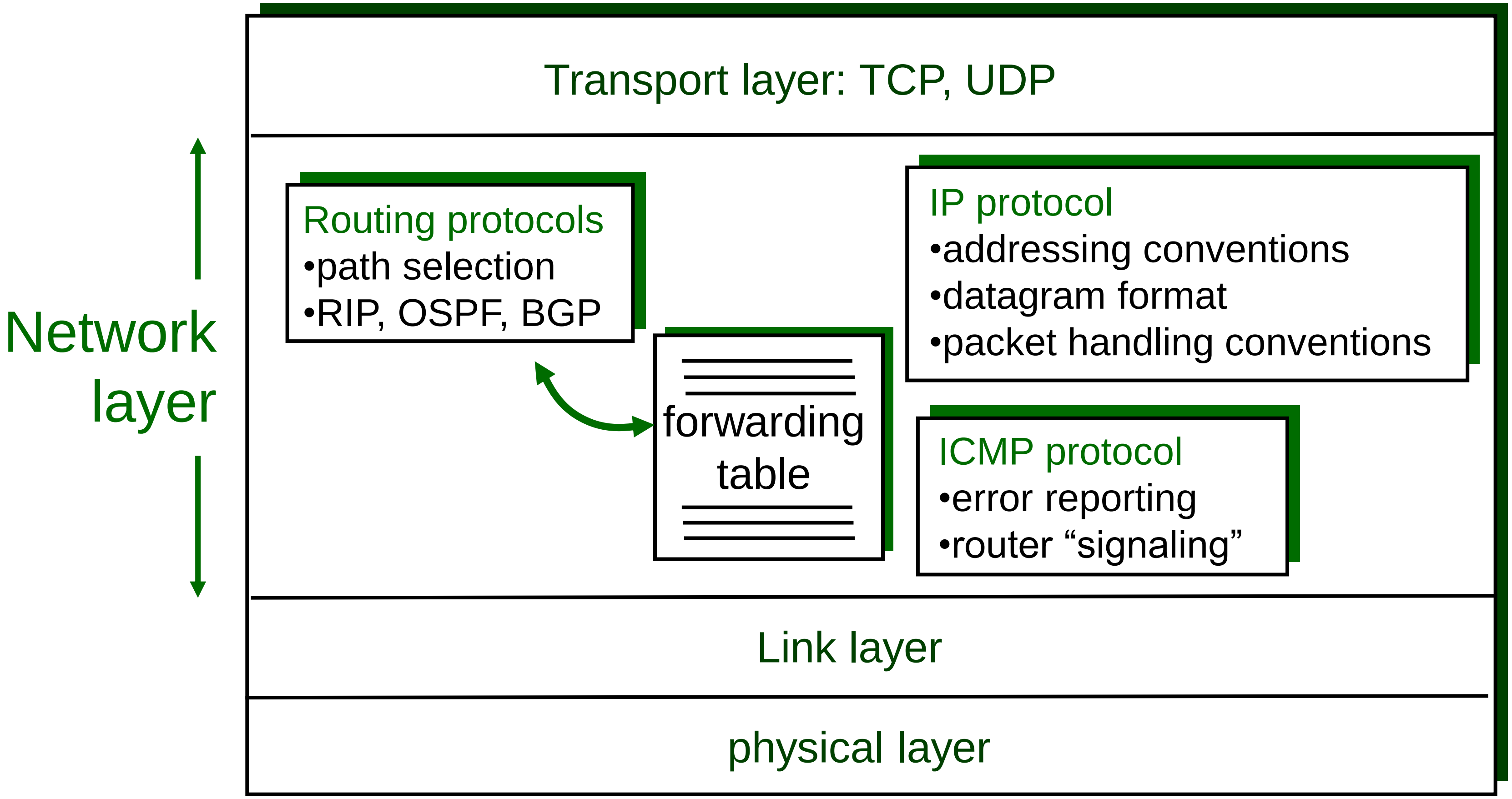
one packet time later:
green packet experiences HOL blocking

Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- **4.3 IP: Internet Protocol**
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

The Internet Network layer

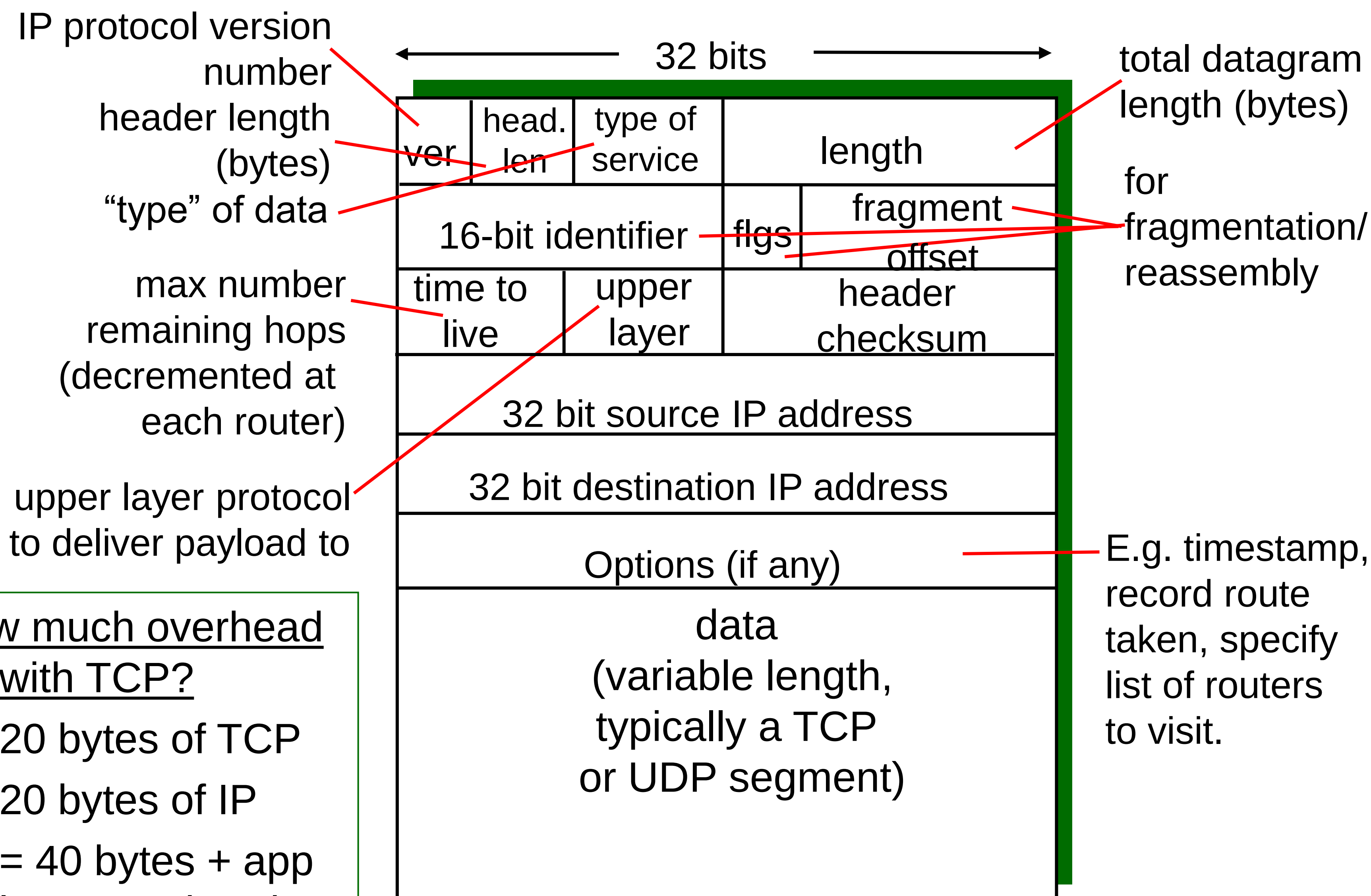
Host, router network layer functions:



Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - **Datagram format**
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

IP datagram format

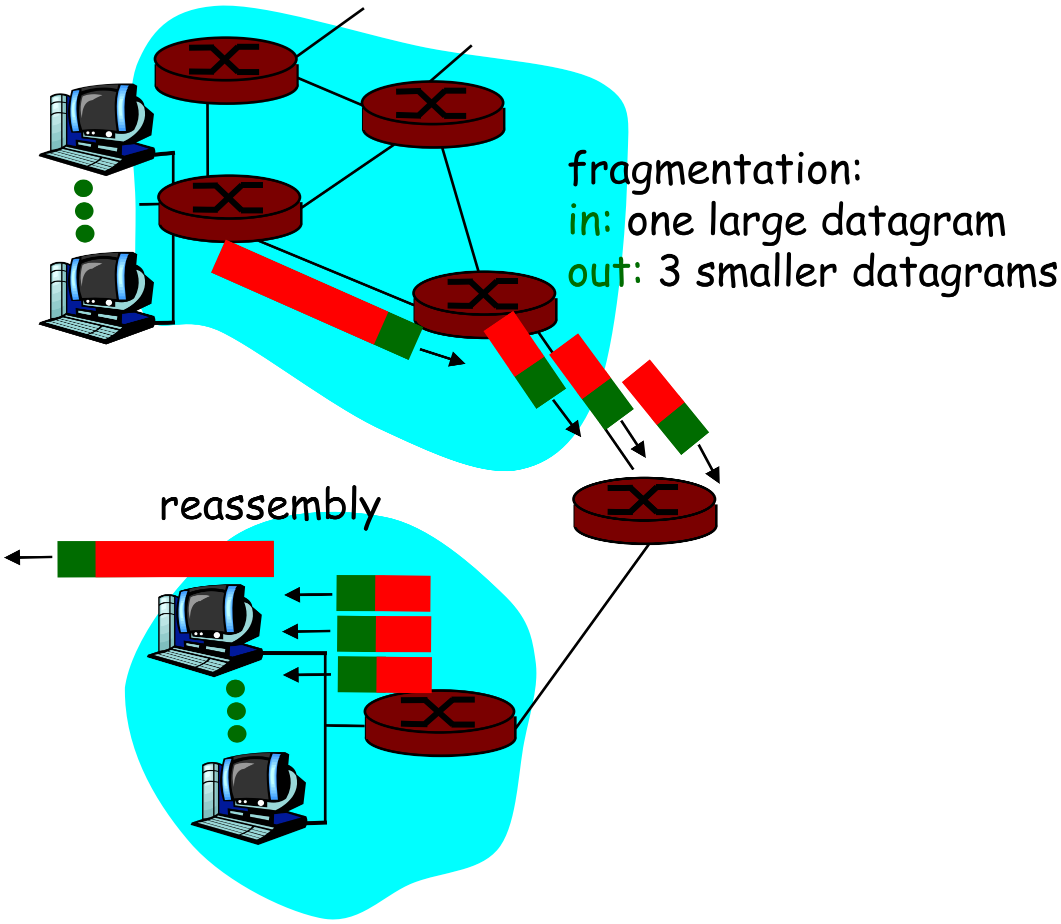


how much overhead with TCP?

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

IP Fragmentation & Reassembly

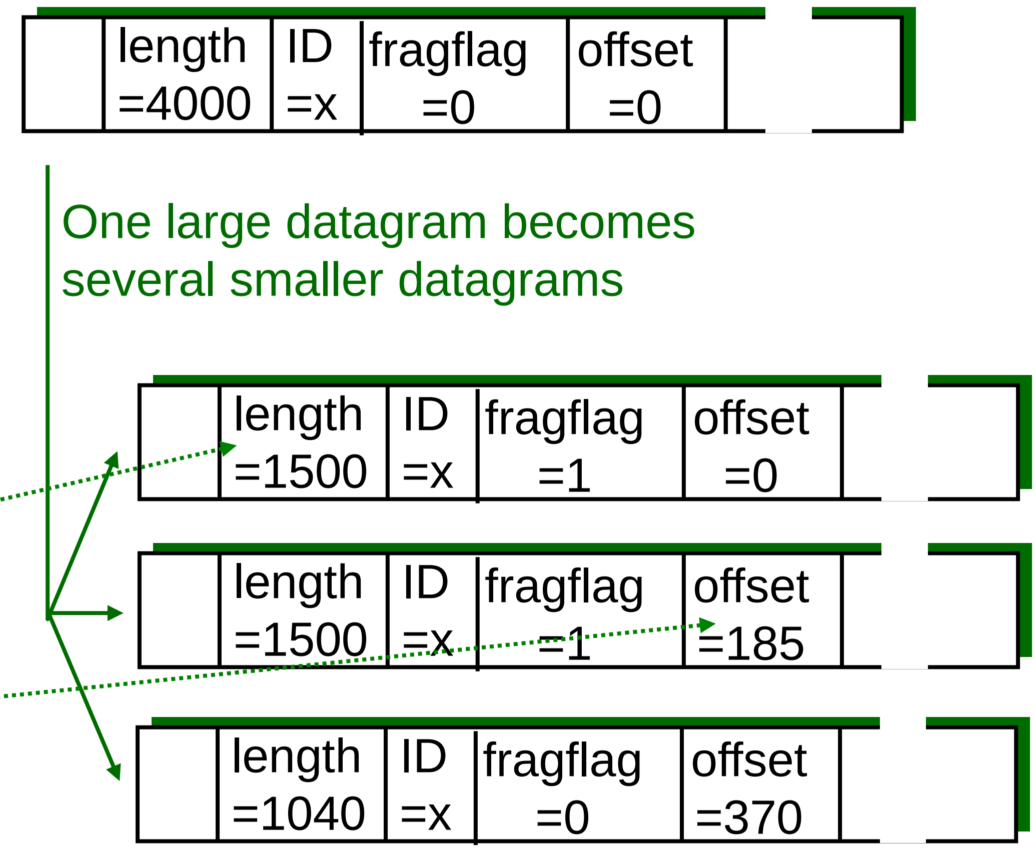
- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify,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



IP Fragmentation and Reassembly

<u>Example</u>

- 4000 byte datagram
- MTU = 1500 bytes

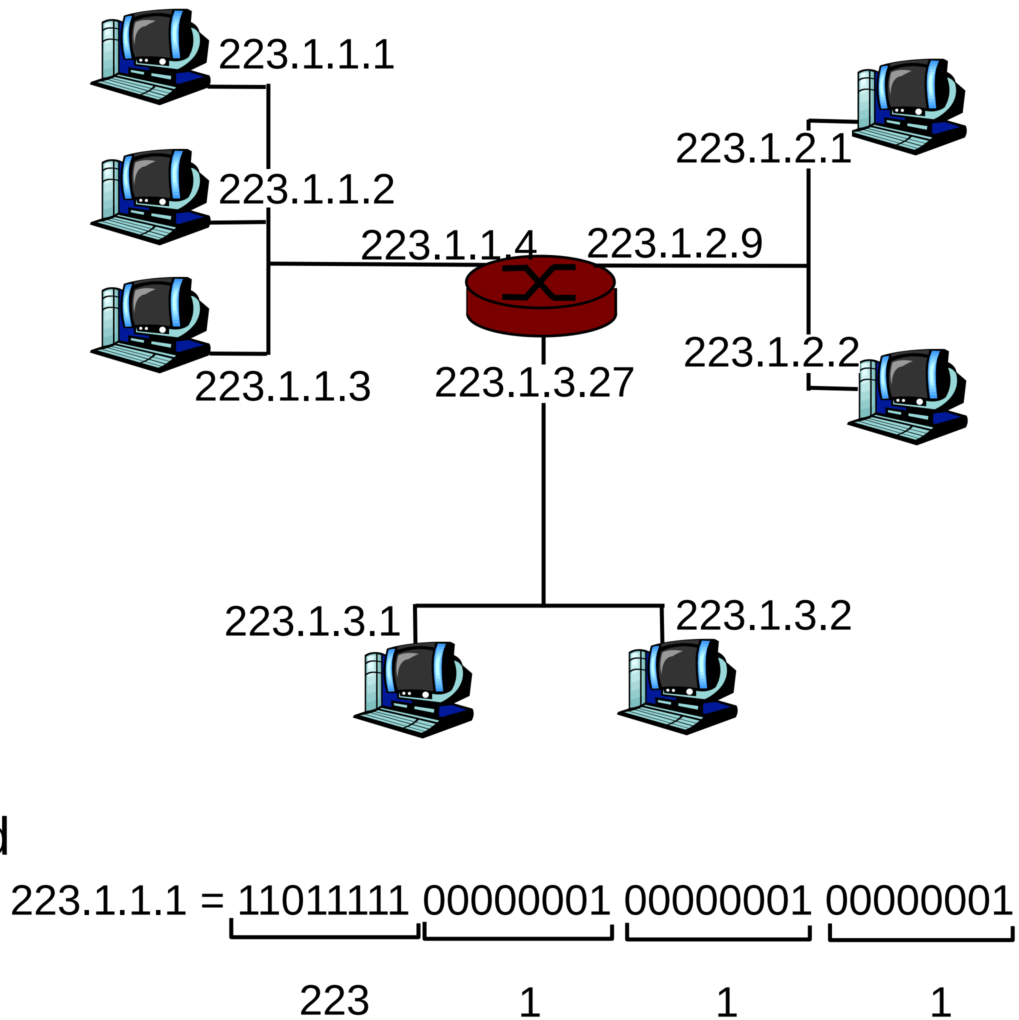


Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - **IPv4 addressing**
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

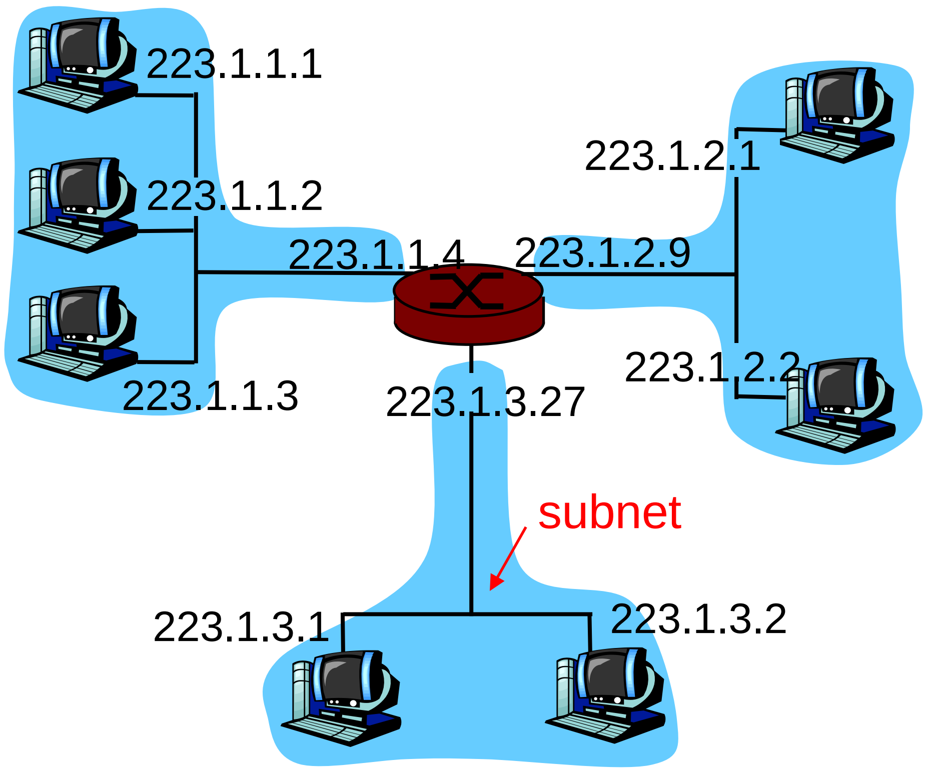
IP Addressing: introduction

- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



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

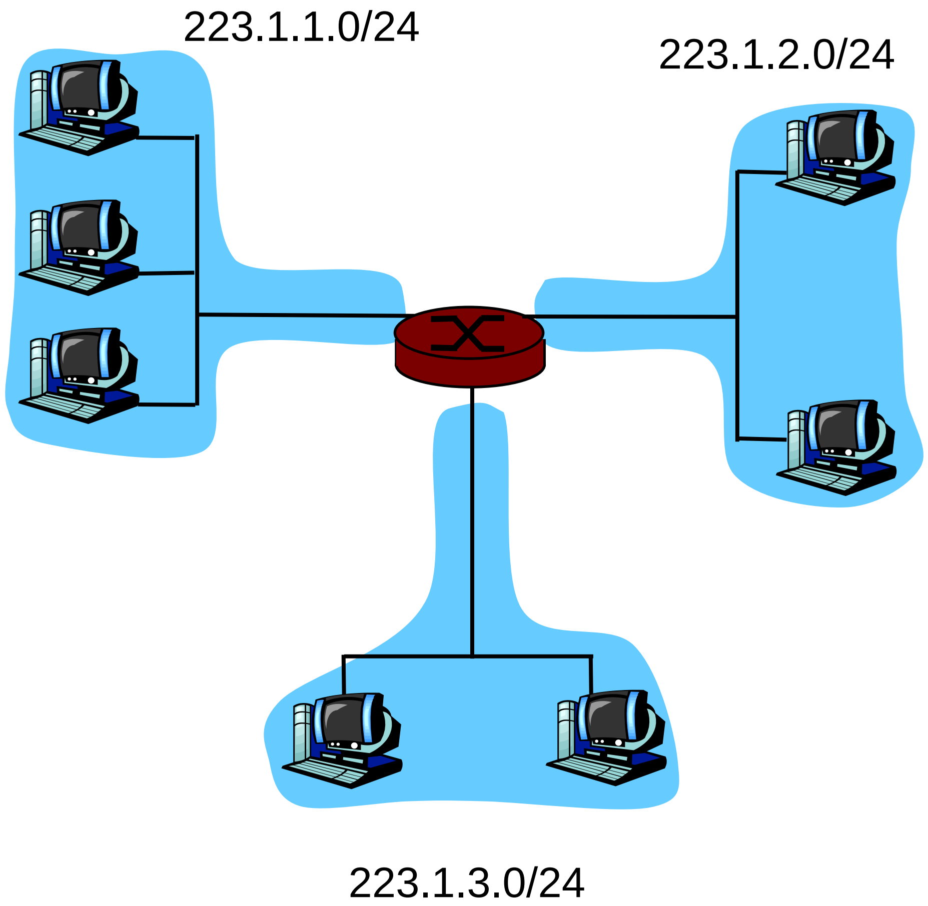


network consisting of 3 subnets

Subnets

Recipe

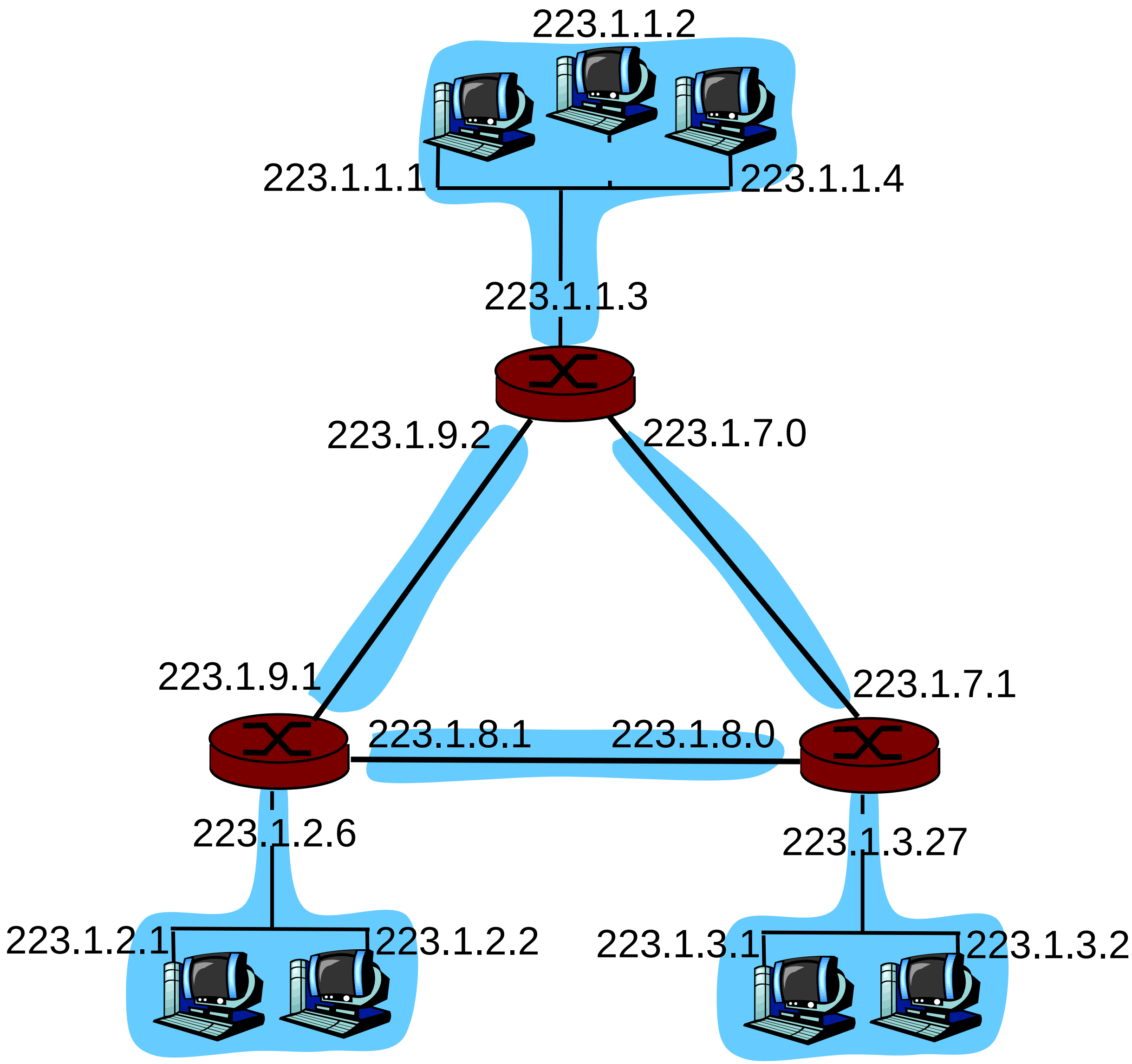
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



Subnet mask: /24

Subnets

How many?



IP addressing: Classful Network

Class	Lead. Bits	Netw. Addr. Bits	No. Of Networks	No. Of Hosts	Addresses
A	0	8	128 (= 2^7)	16,777,216 (= 2^{24})	0.0.0.0 to 127.255.255.255
B	10	16	16,384 (= 2^{14})	65,536 (= 2^{16})	128.0.0.0 to 191.255.255.255
C	110	24	2,097,152 (= 2^{21})	256 (= 2^8)	192.0.0.0 to 223.255.255.255
D	1110	n/d	n/d	n/d	224.0.0.0 to 239.255.255.255
E	1111	n/d	n/d	n/d	240.0.0.0 to 255.255.255.254

IP addressing: CIDR

CIDR: **C**lassless **I**nter**D**omain **R**outing

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

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP addresses: how to get one?

- Q: How does a host get IP address?

- A1: hard-coded by system admin in a file
- A2: DHCP: Dynamic Host Configuration Protocol
  - dynamically get address from a server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

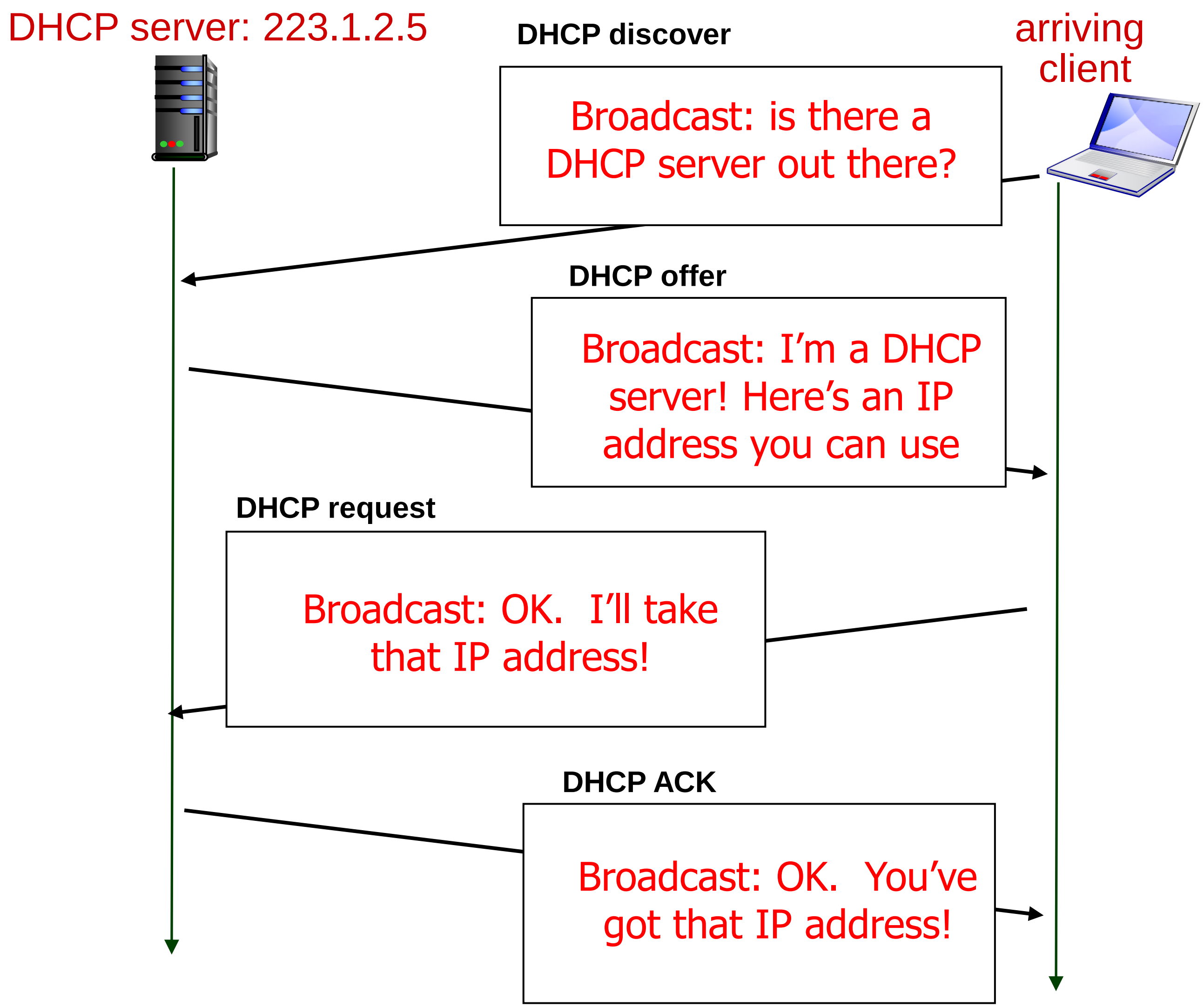
Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- Can renew its lease on address in use
- Allows reuse of addresses (only hold address while connected and "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
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



IP addresses: how to get one?

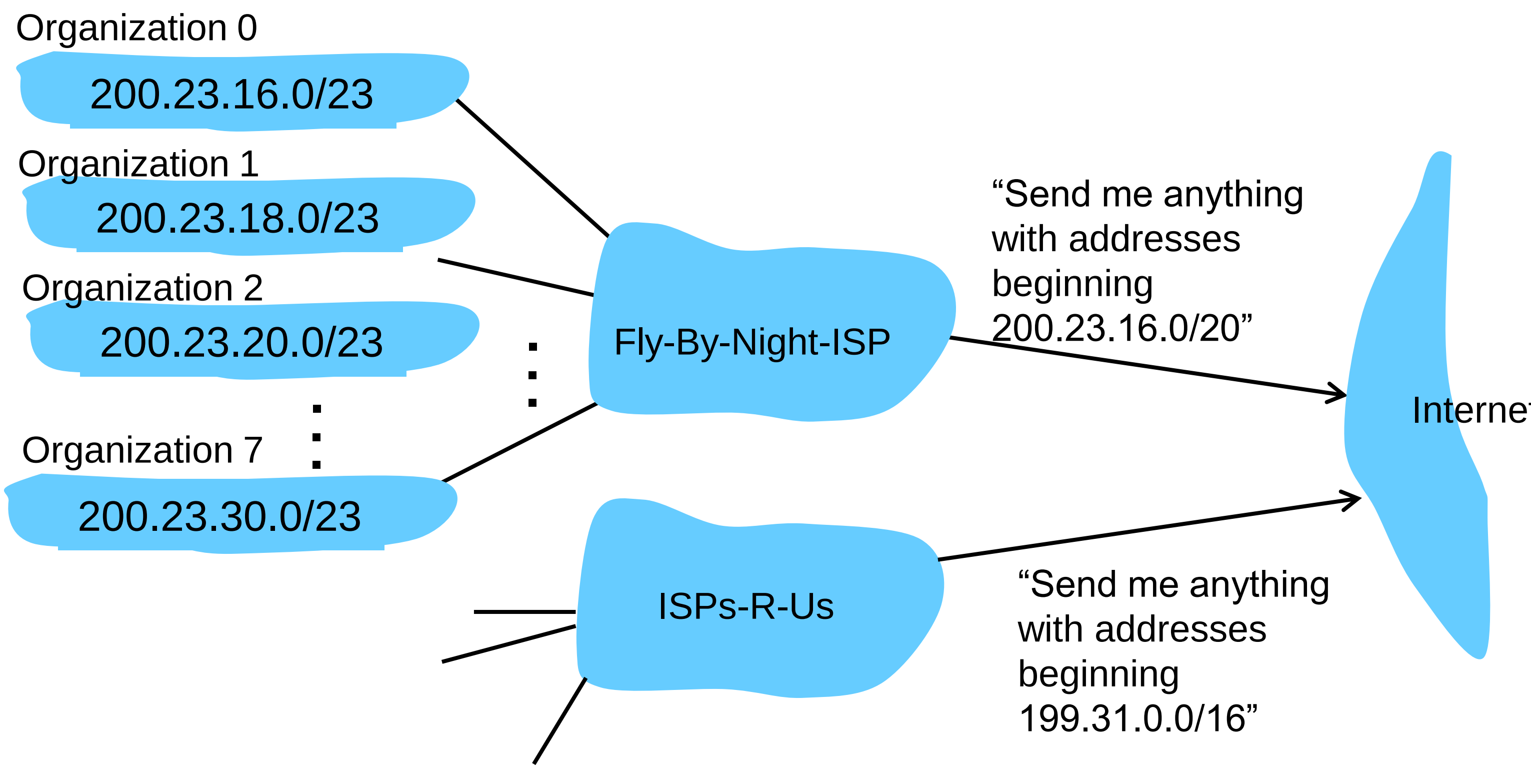
Q: How does *network* get subnet part of IP addr?

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

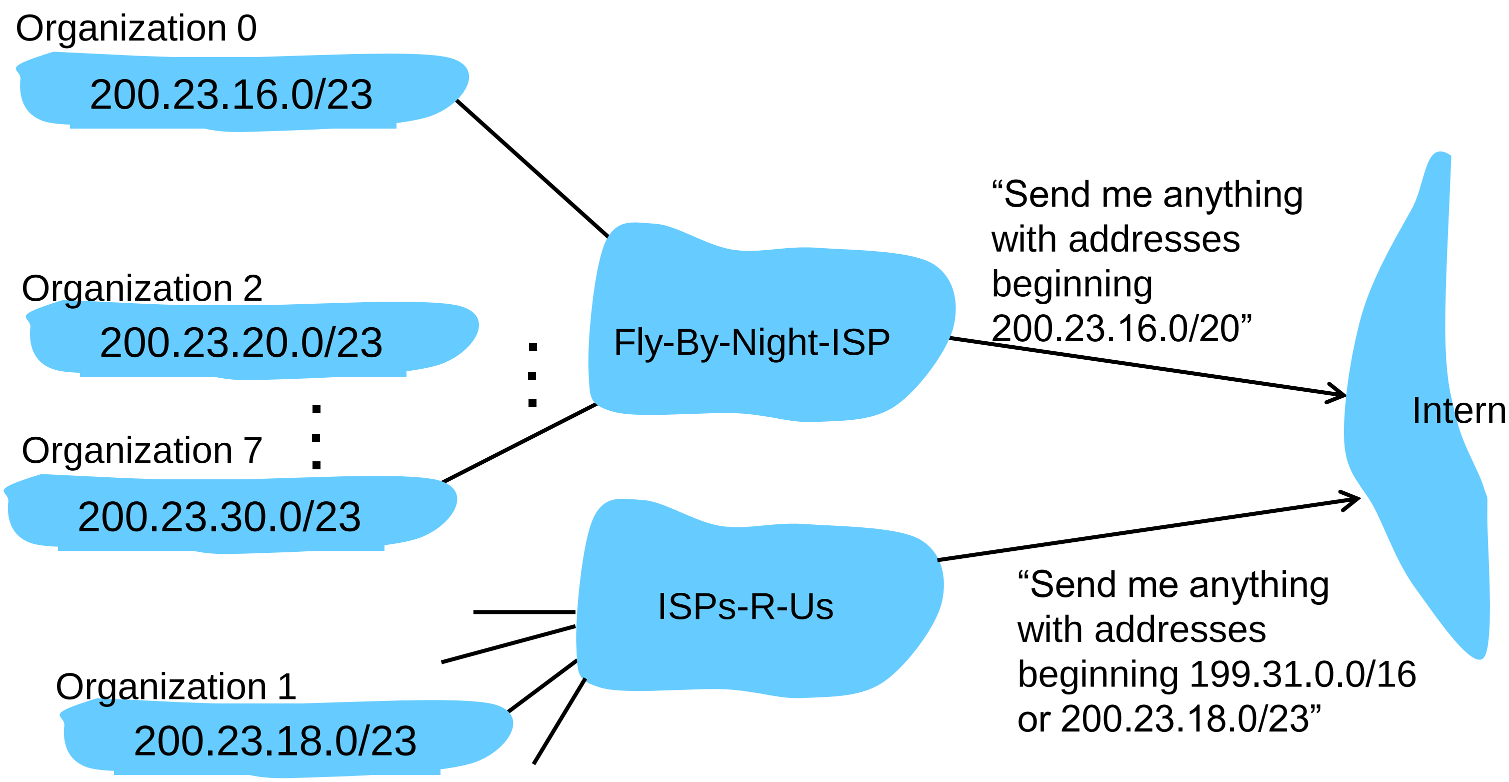
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



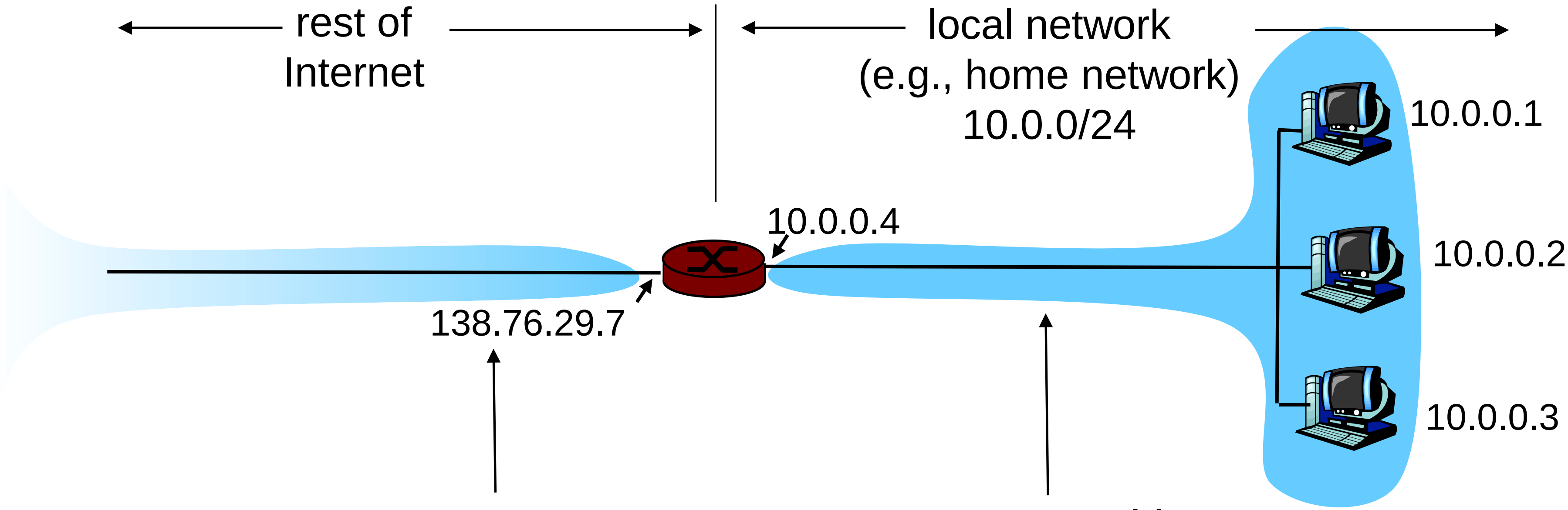
IP addressing: the last word...

- Q: How does an ISP get block of addresses?
- A: ICANN: Internet Corporation for Assigned Names and Numbers
  - allocates addresses
  - manages DNS
  - assigns domain names, resolves disputes

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



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

Implementation: NAT router must:

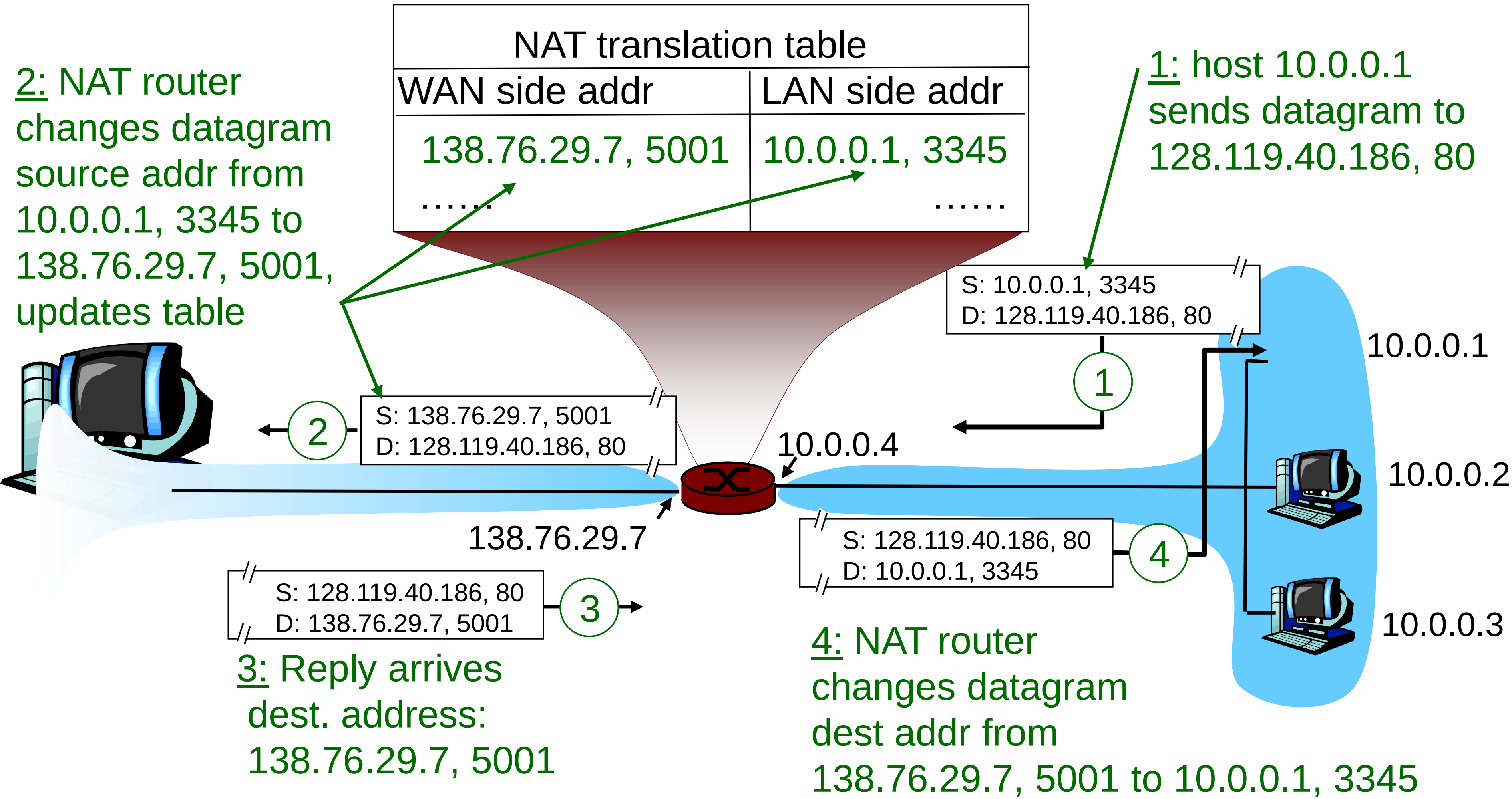
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . 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

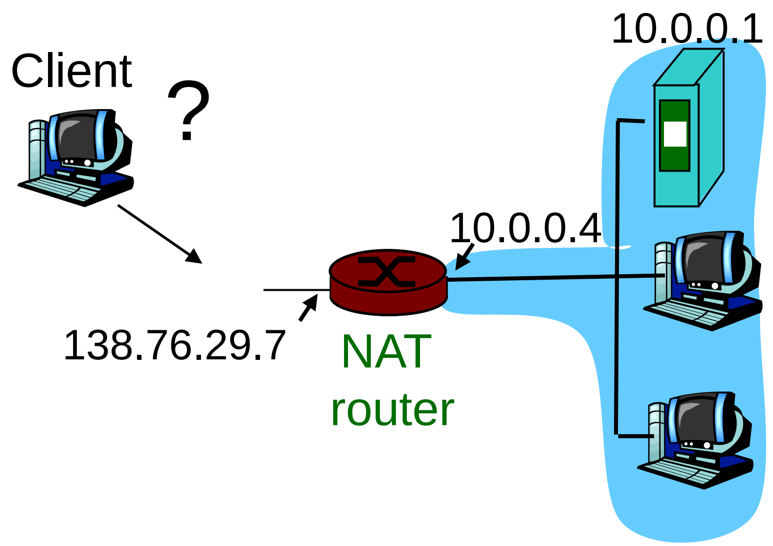


NAT: Network Address Translation

- 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
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, eg, P2P applications
  - address shortage should instead be solved by IPv6

NAT traversal problem

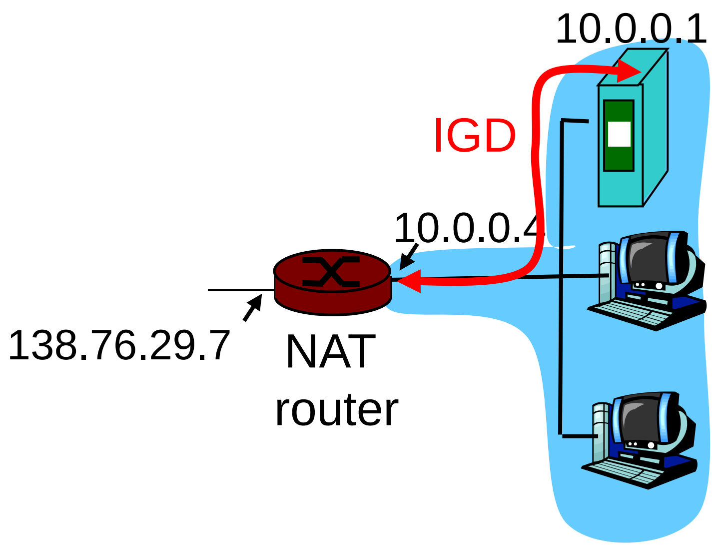
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

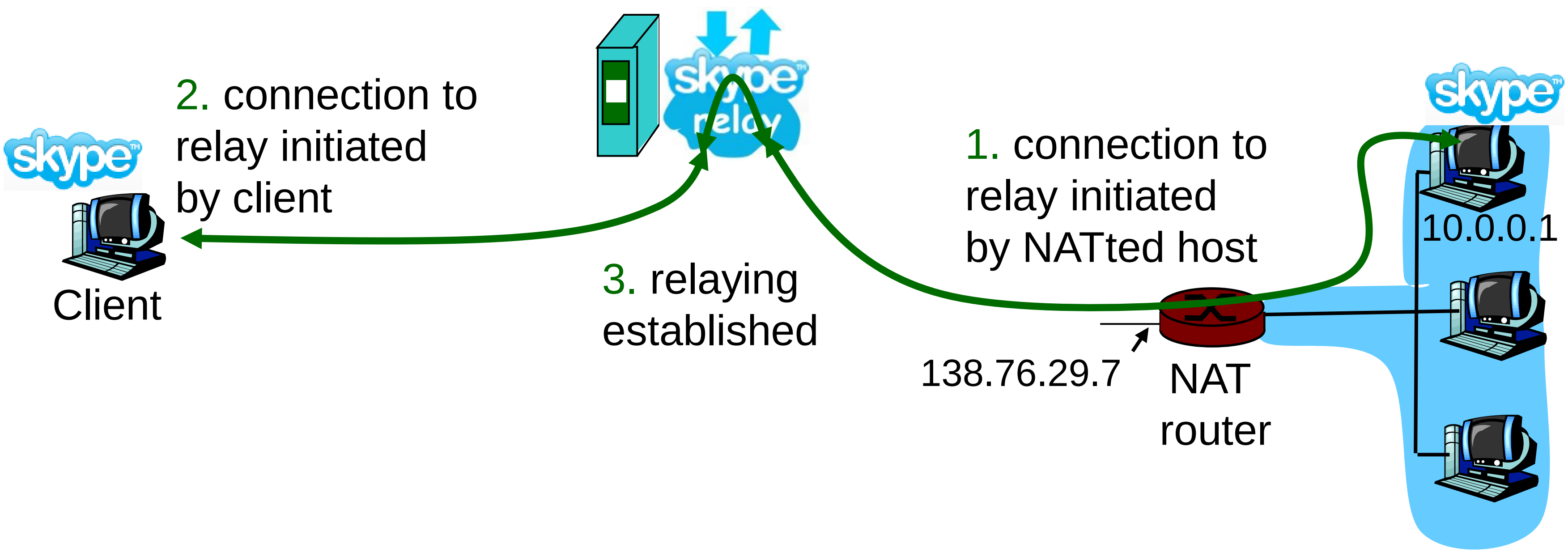
- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  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 to connections



Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - **ICMP**
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical 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, code plus first 8 bytes of IP datagram causing error

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

Traceroute and ICMP

- Source sends series of UDP segments to dest
  - First has TTL =1
  - Second has TTL=2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router& IP address

- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times

Stopping criterion

- UDP segment eventually arrives at destination host
- Destination returns ICMP "host unreachable" packet (type 3, code 3)
- When source gets this ICMP, stops.

Network Layer

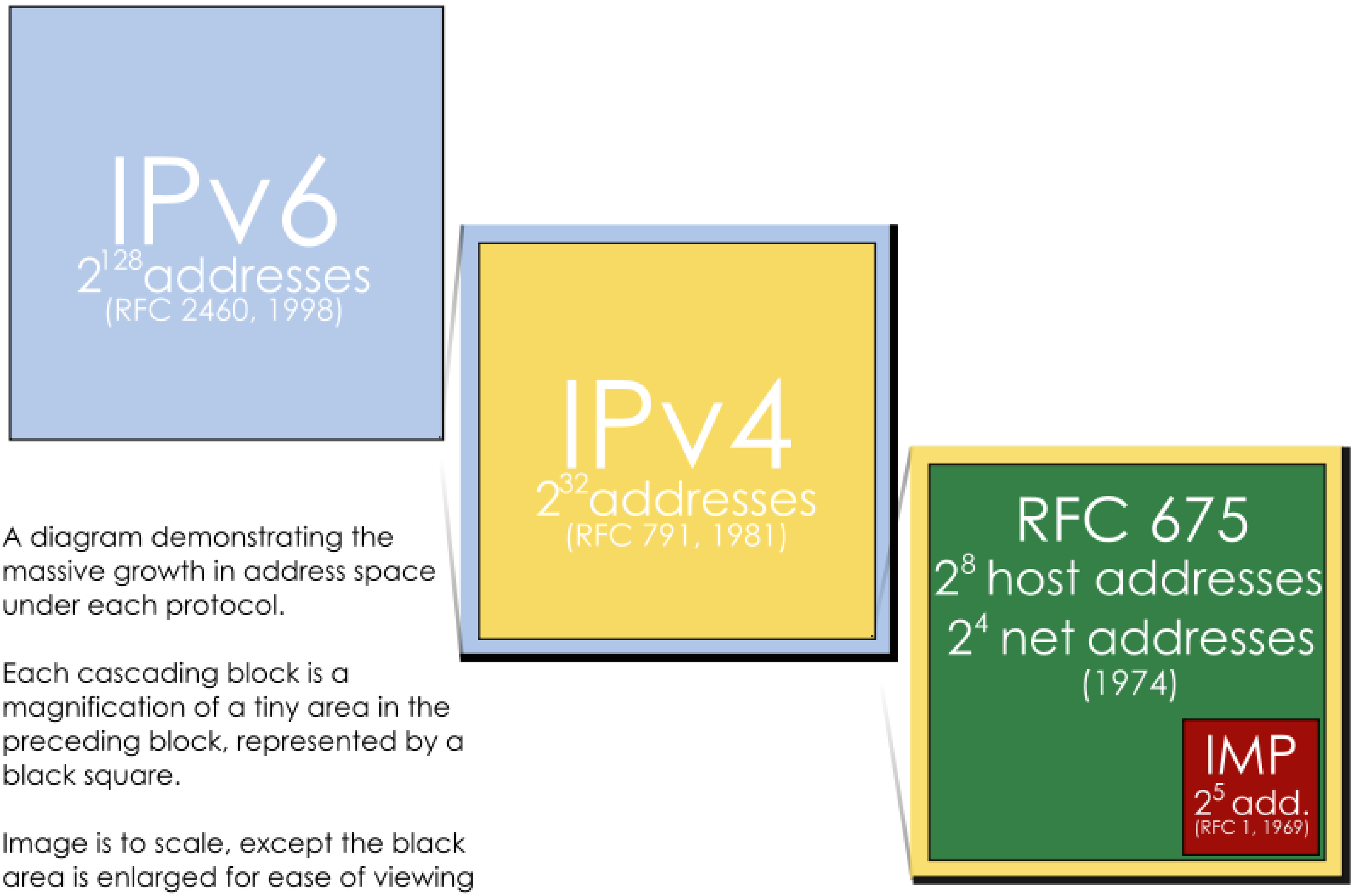
- 4.1 Introduction
- 4.2 What’s inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - **IPv6**
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

IPv6

- 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

IPv4 vs IPv6

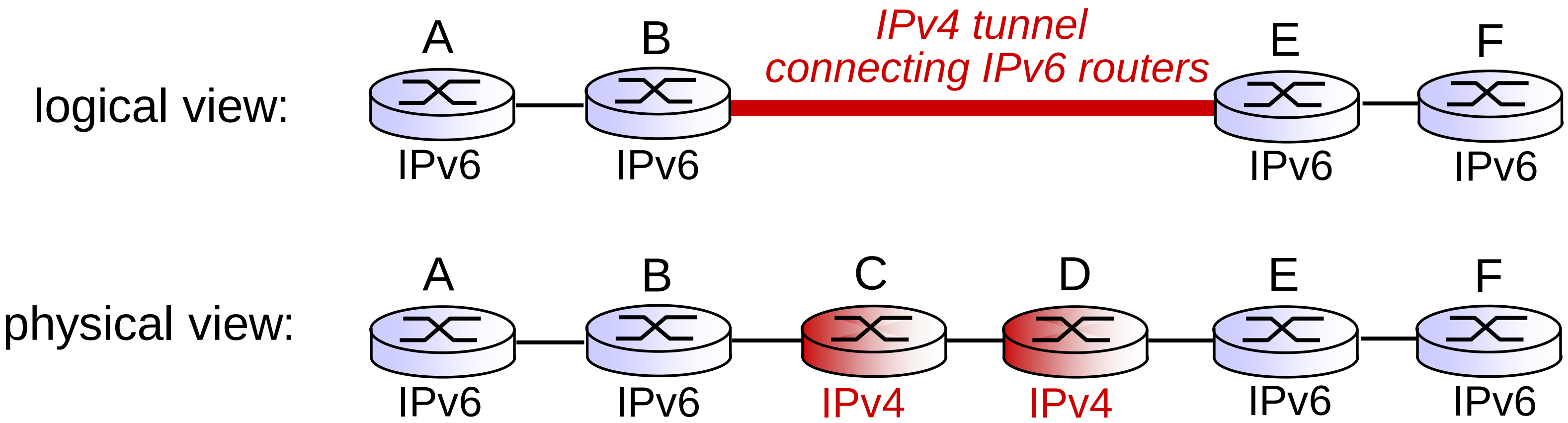


Source: http://commons.wikimedia.org/wiki/File:Internet_address_spaces.svg

Transition From IPv4 To IPv6

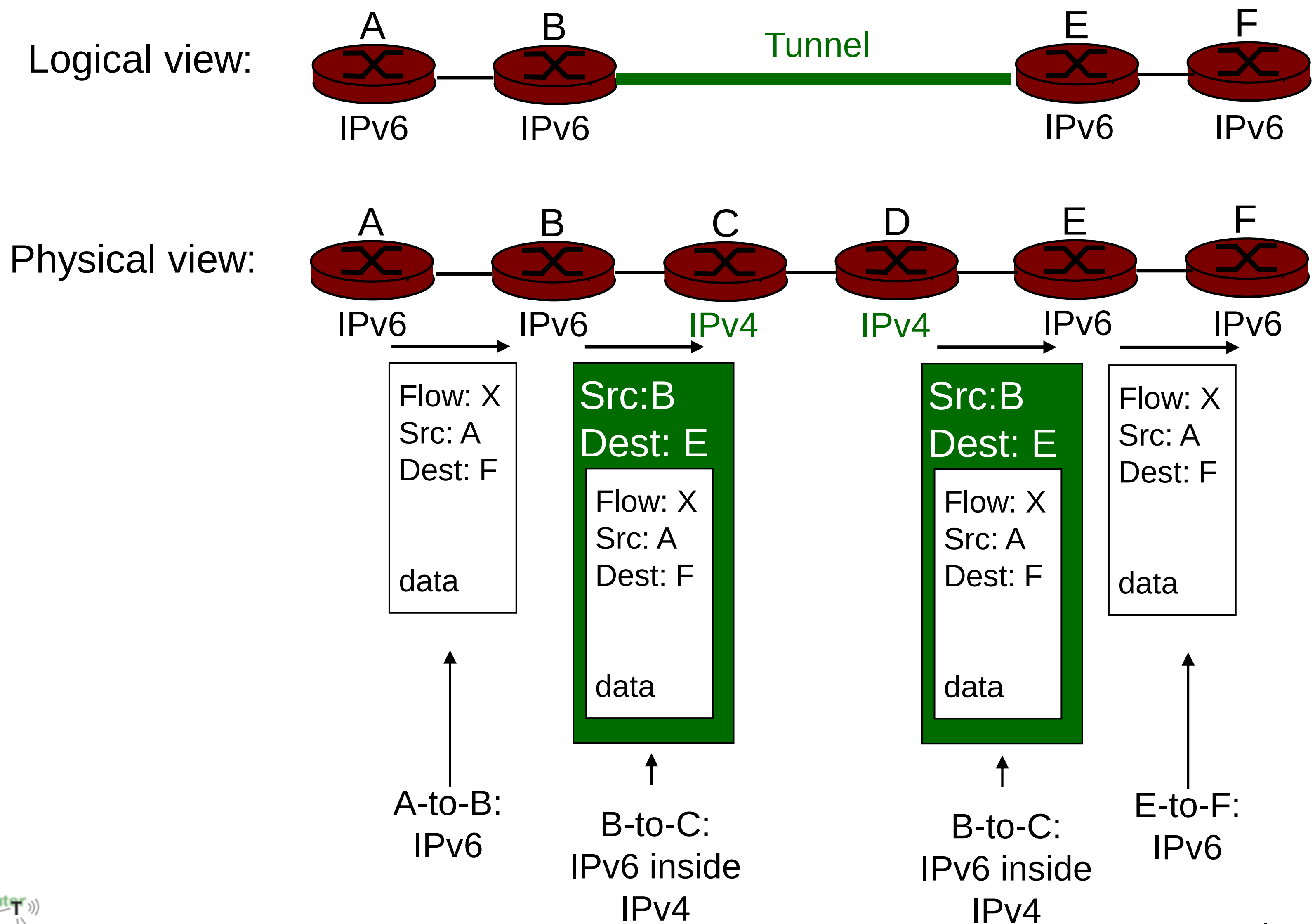
- Not all routers can be upgraded simultaneously
  - no "flag days"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers
- Dual stack: Network stack supports IPv4 and IPv6

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

Tunneling


Logical view:
A
B
Tunnel
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

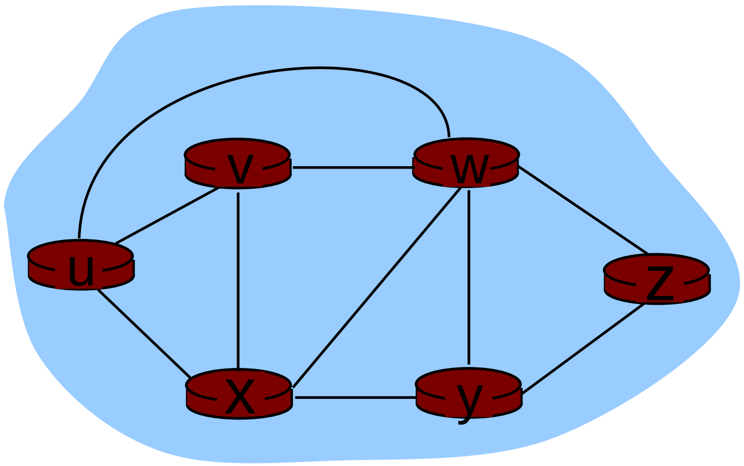
Dual stack

- Nodes have the ability to send and receive both IPv4 and IPv6 packets
- Direct connection to IPv4 nodes using IPv4 packets
- Direct connection to IPv6 nodes using IPv6 packets
- Can be used together with tunneling

Network Layer

- 3.1 Introduction
- 3.2 What's inside a router
- 3.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 3.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

Graph abstraction



Graph: G = (N,E)

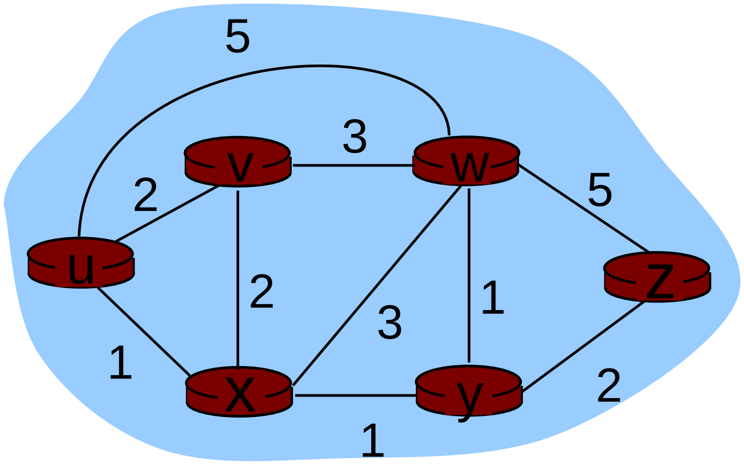
N = set of routers = { u, v, w, x, y, z }

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

Remark: Graph abstraction is useful in other network contexts

Example: 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 $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

- Global:
  - all routers have complete topology, link cost info
  - "link state" algorithms
- Decentralized:
  - router knows physically-connected neighbors, link costs to neighbors
  - iterative process of computation, exchange of info with neighbors
  - "distance vector" algorithms

Static or dynamic?

- Static:
  - routes change slowly over time
- Dynamic:
  - routes change more quickly
    - periodic update
    - in response to link cost changes

Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - **Link state**
  - Distance Vector
  - Hierarchical routing

A Link-State Routing Algorithm

Dijkstra's algorithm

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

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 is definitively known

Dijsktra's Algorithm

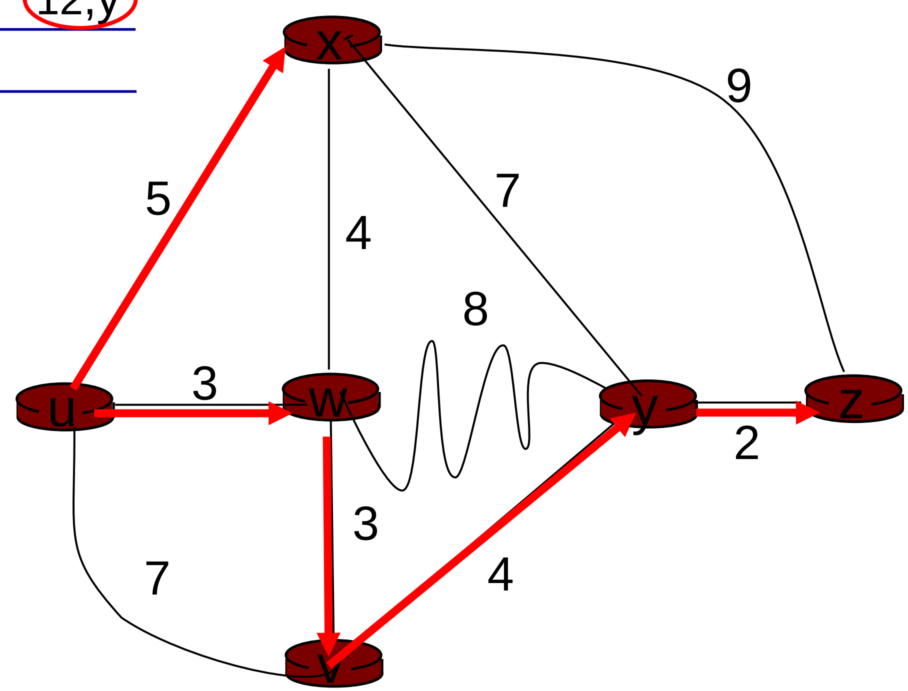
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

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					

notes:

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



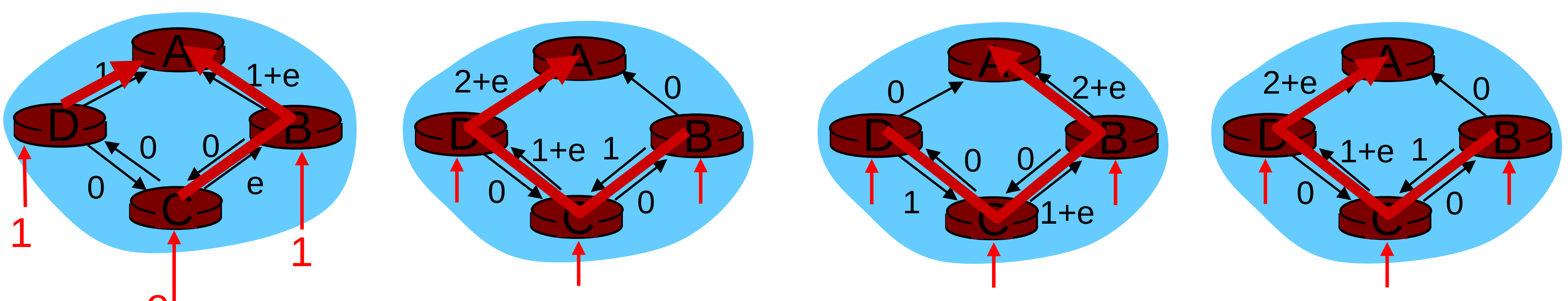
Dijkstra’s algorithm, discussion

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(n*log(n))

Oscillations possible:

- e.g., link cost = amount of carried traffic



initially

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

Network Layer II

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - **Distance Vector**
  - Hierarchical routing

Bellman–Ford algorithm

- Finds shortest paths in weighted, directed graph for given source

- Approach: relax all edges repeatedly until stable ($|V| - 1$ times)

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

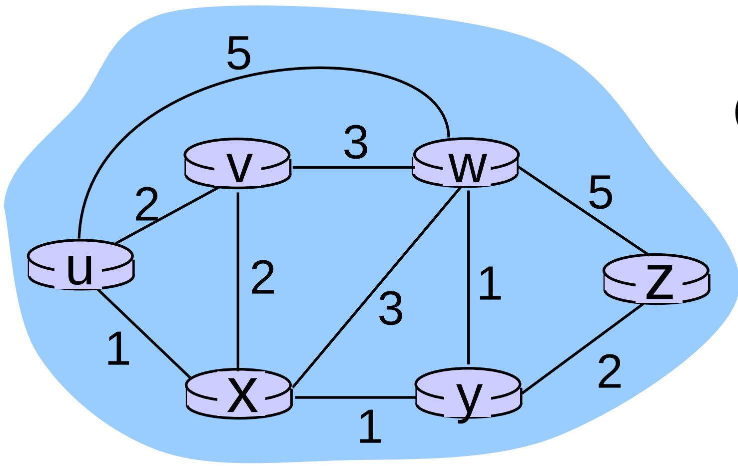
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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 that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

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

Distance vector algorithm – Basic Idea

- From time-to-time, each node sends its own distance vector estimate to neighbors
- Asynchronous
- When x receives new DV estimate from neighbor, it updates its own DV using BF 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 Dx(y) converge to the actual least cost dx(y)

Distance Vector Algorithm (cont'd)

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

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

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

node y table

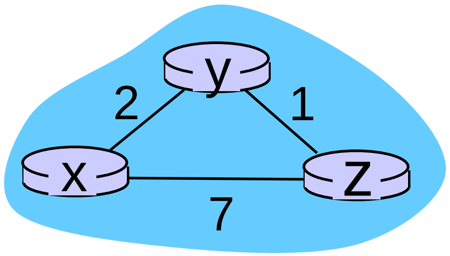
cost to

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



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

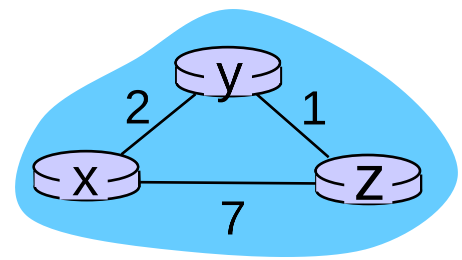
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

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

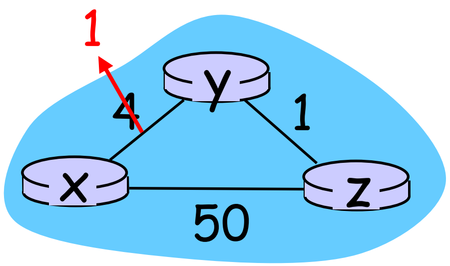


time

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



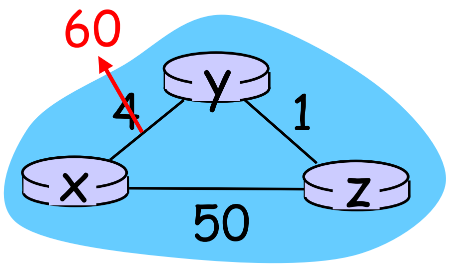
"good news travels fast"

At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbors.

At time t_1, *z* receives the update from *y* and updates its table. It computes a new least cost to *x* and sends its neighbors its DV.

At time t_2, *y* receives *z*'s update and updates its distance table. *y*'s least costs do not change and hence *y* does *not* send any message to *z*.

Distance Vector: link cost changes



Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see textbook

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)

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 through network

Network Layer II

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - **Hierarchical routing**

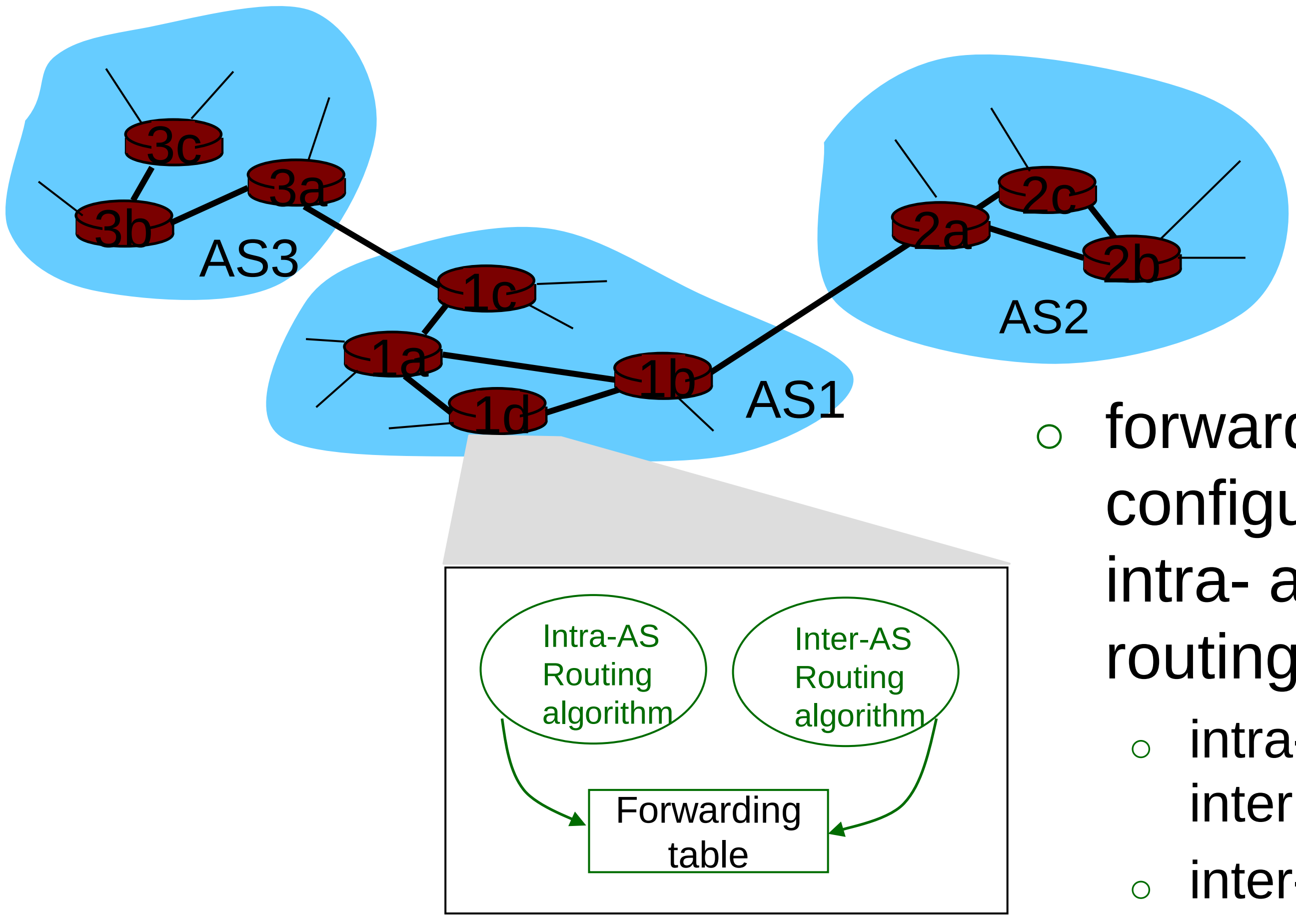
Hierarchical Routing

- Our routing study thus far - idealization
  - all routers identical
  - network “flat”
  - … not true in practice
- Scale: with 200 million destinations:
  - can’t store all dest’s in routing tables
  - routing table exchange would swamp links
- Administrative autonomy
  - internet = network of networks
  - each network admin may want to control routing in its own network

Hierarchical Routing

- Aggregate routers into regions
  - Autonomous Systems (AS)
- Routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol
- Gateway router
  - Direct 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!