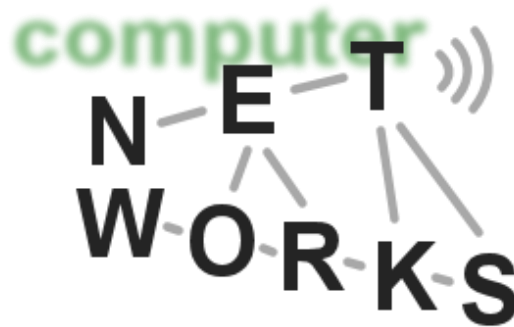


# Network Layer – Part I

Computer Networks, Winter 2012/2013

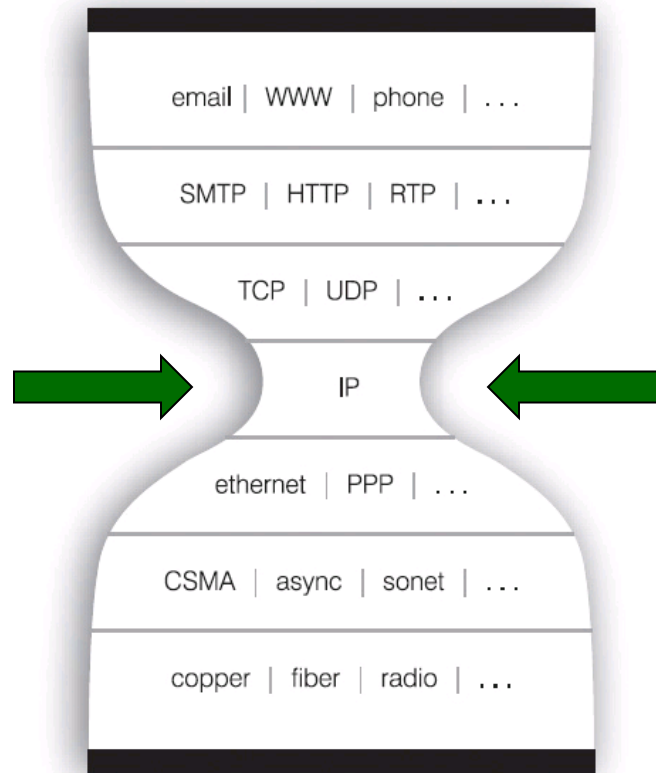


Acknowledgements: J. Kurose & K. Ross – Computer Networking: A Top-Down Approach Featuring the Internet

# Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6

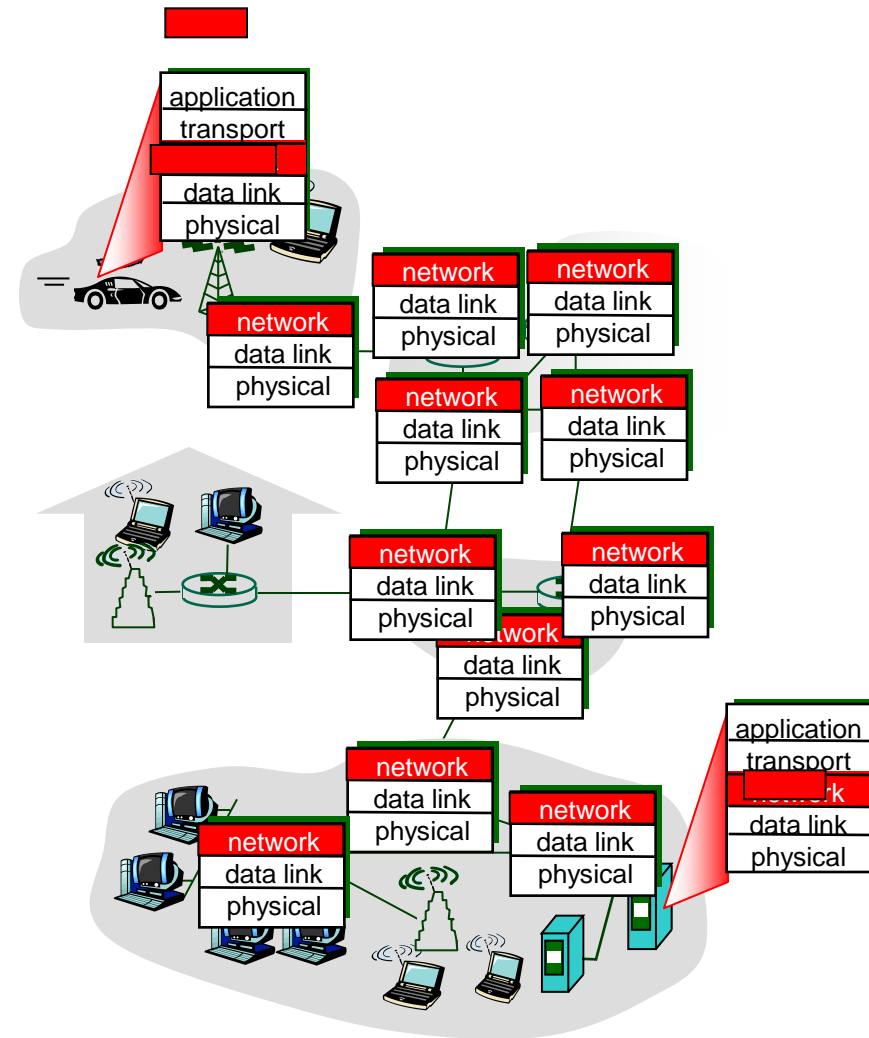
# “Hourglass” architecture



© Jonathan L. Zittrain (<http://yupnet.org/zittrain/archives/13>)

# Network layer

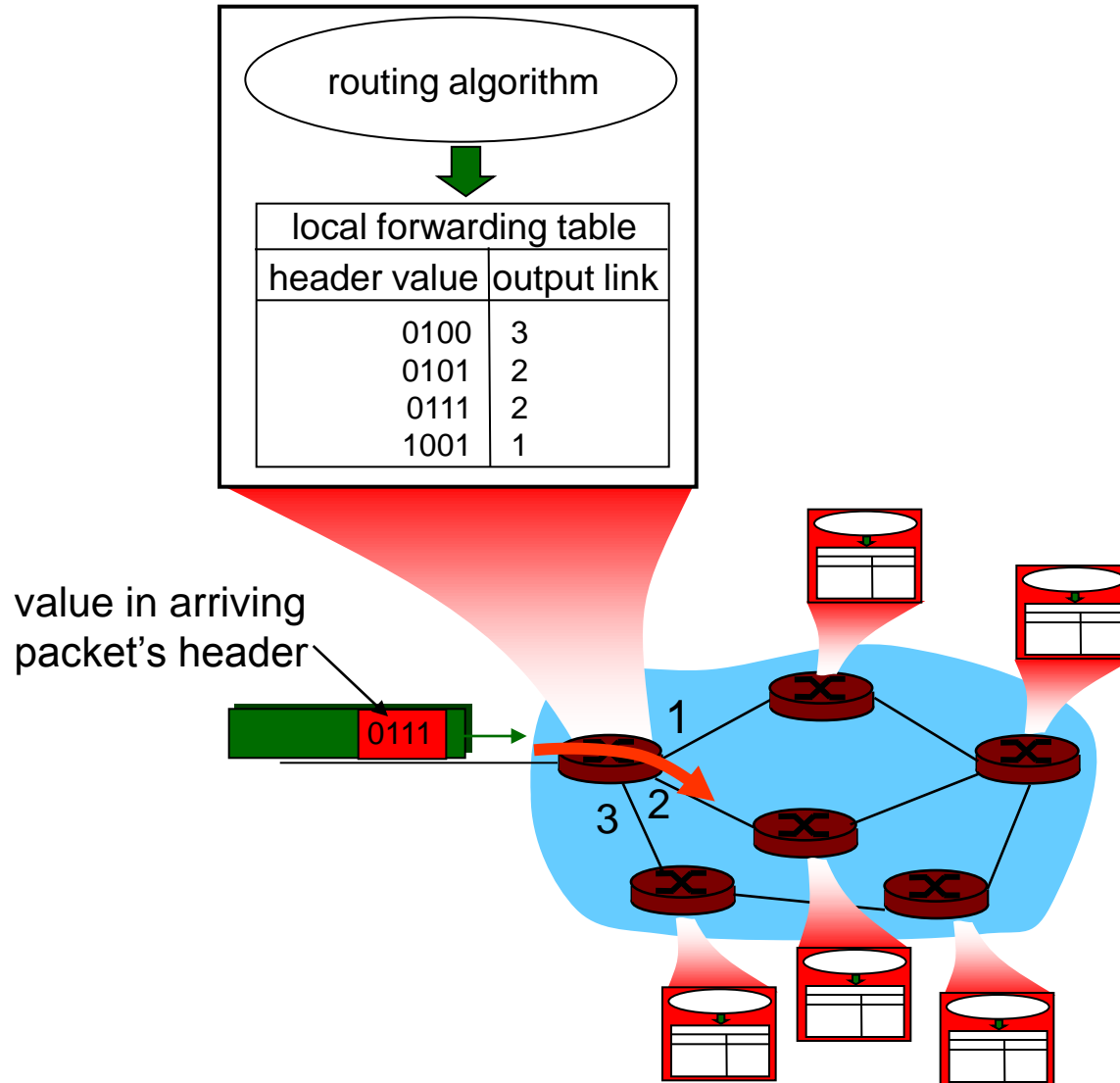
- Transports segments from sending to receiving host
- **Sending side:** encapsulates segments into datagrams
- **Receiving side:** delivers segments to transport layer
- Network layer protocols in *every* host, router
- Router examines header fields in all IP datagrams passing through it



# 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

# Interplay between routing and forwarding



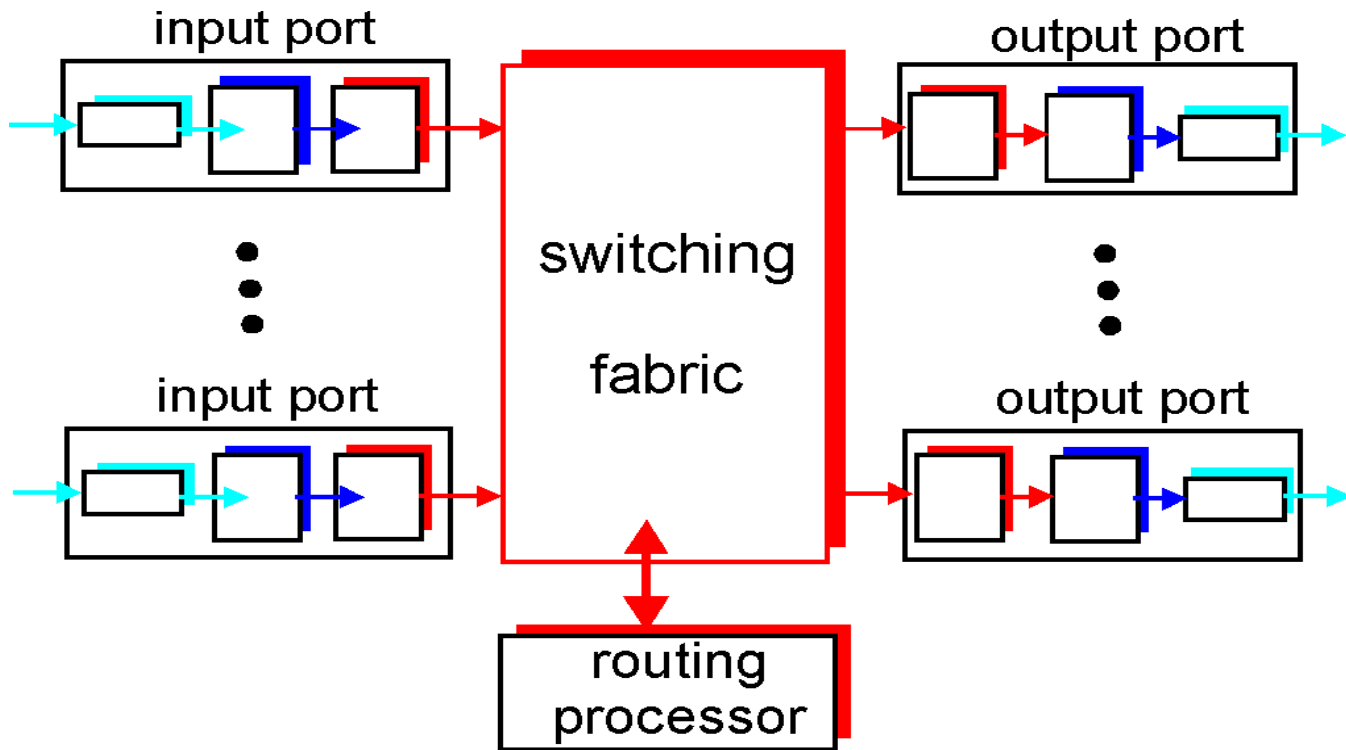
# Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6

# Router Architecture Overview

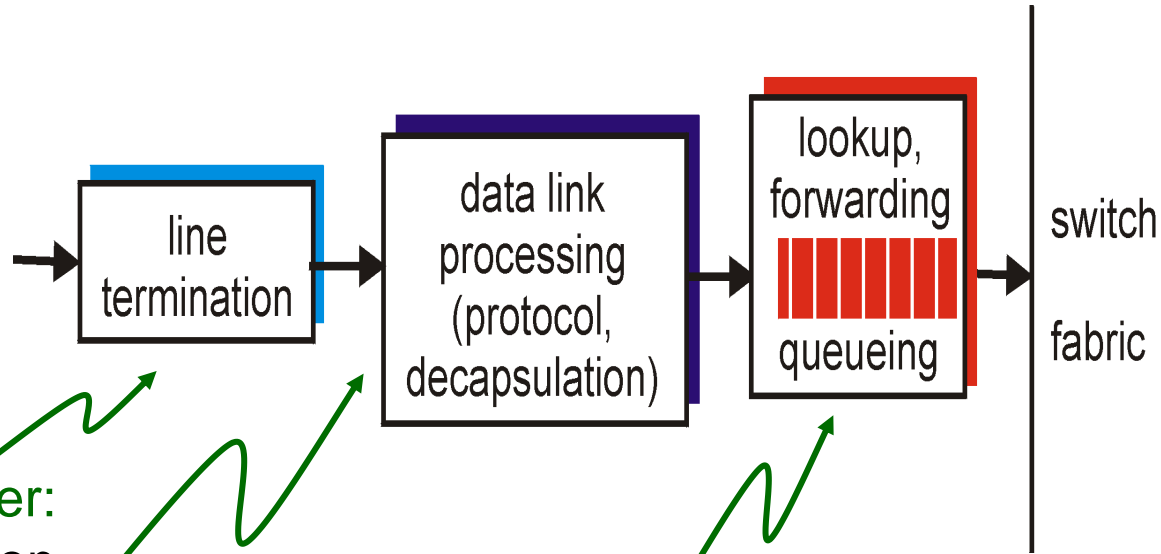
Two key router functions:

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





# Input Port Functions



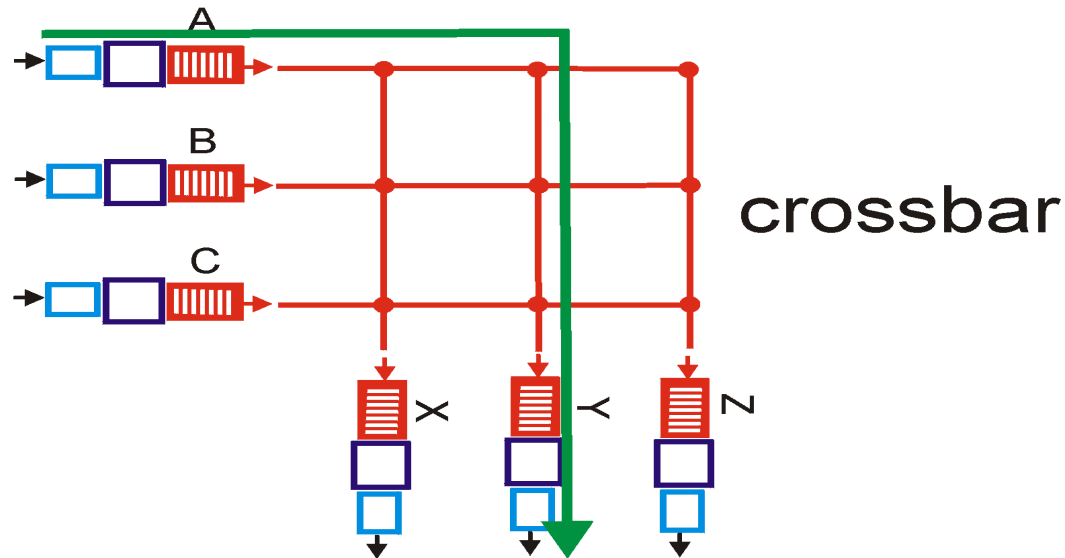
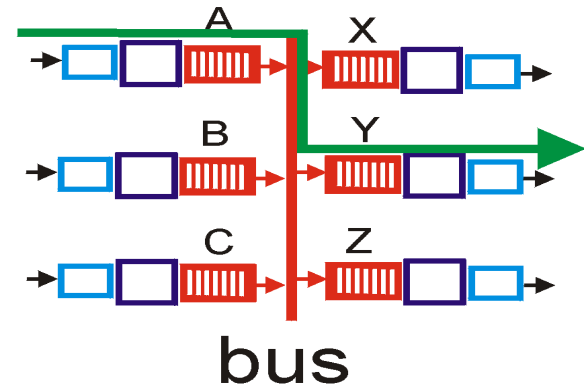
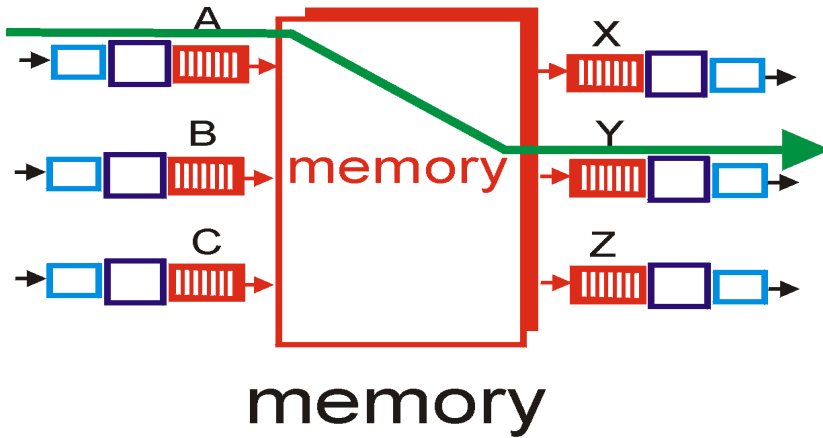
Physical layer:  
bit-level reception

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

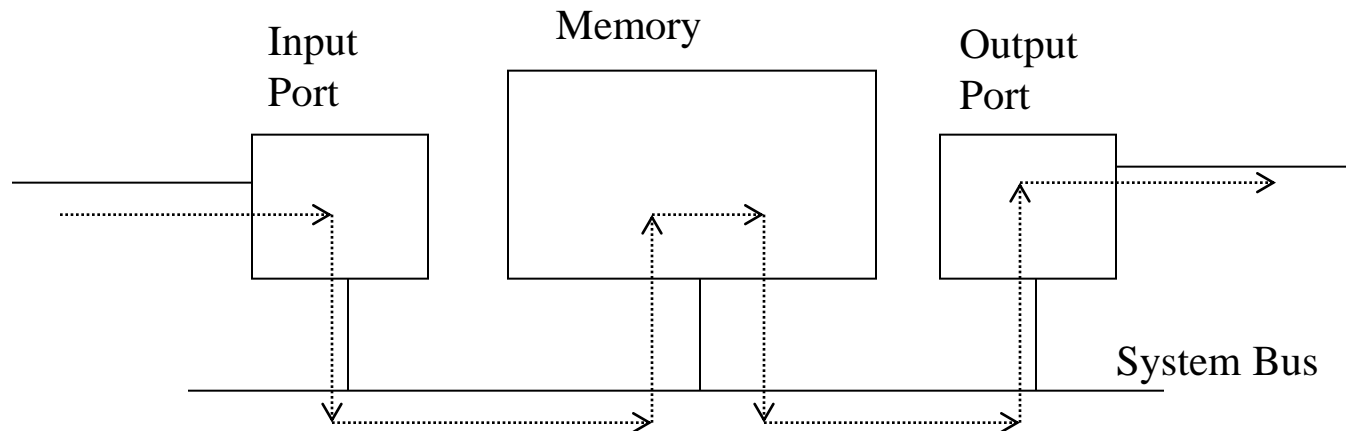
# Three types of switching fabrics



# 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)



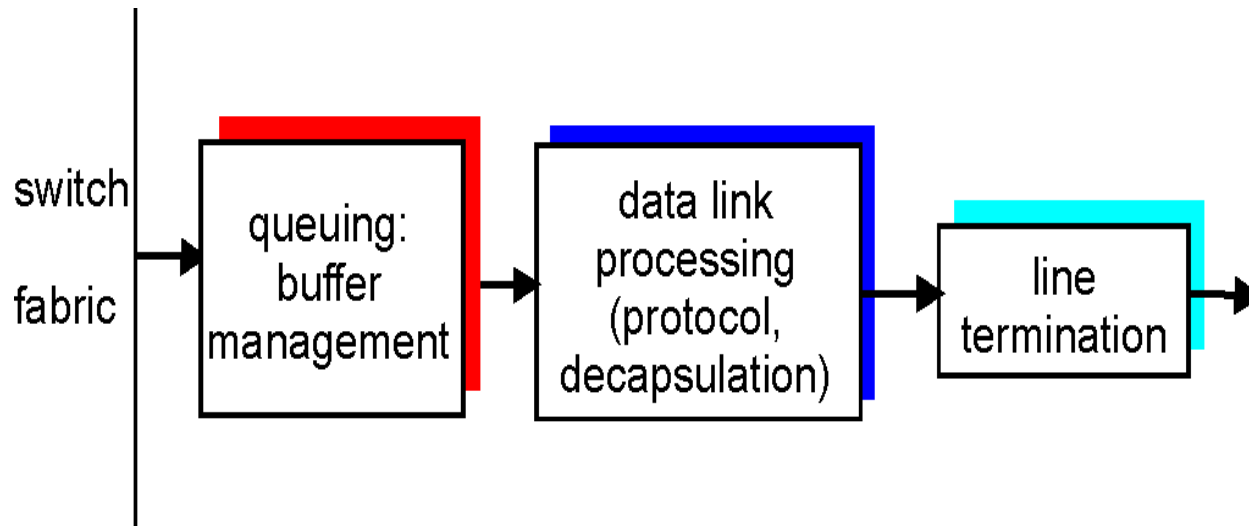
# Switching Via a Bus

- 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

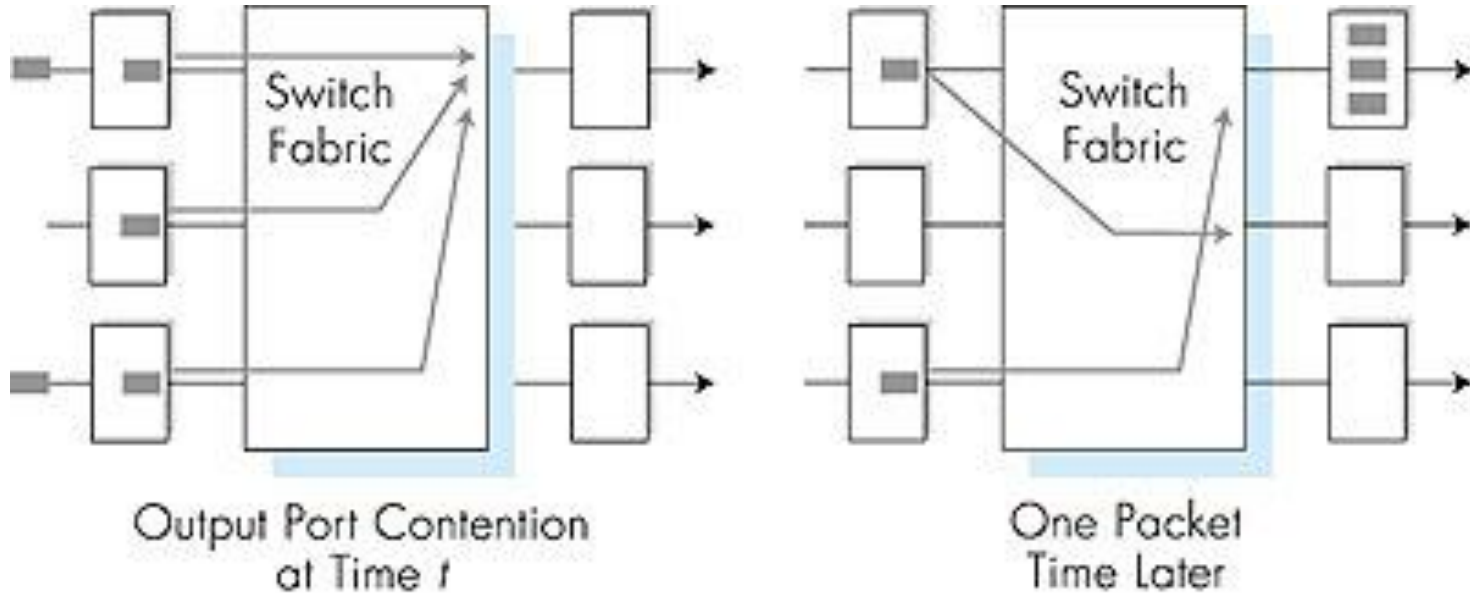
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection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



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

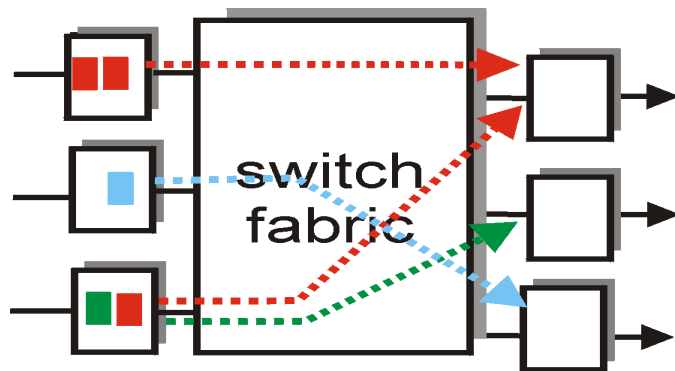
# How much buffering?

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

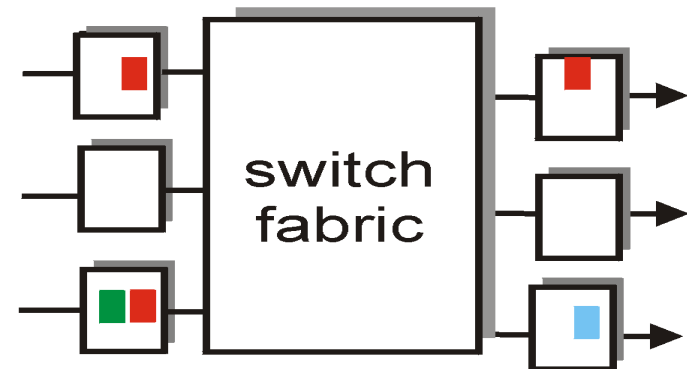


# Input Port Queuing

- 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  
at time t - only one red  
packet can be transferred



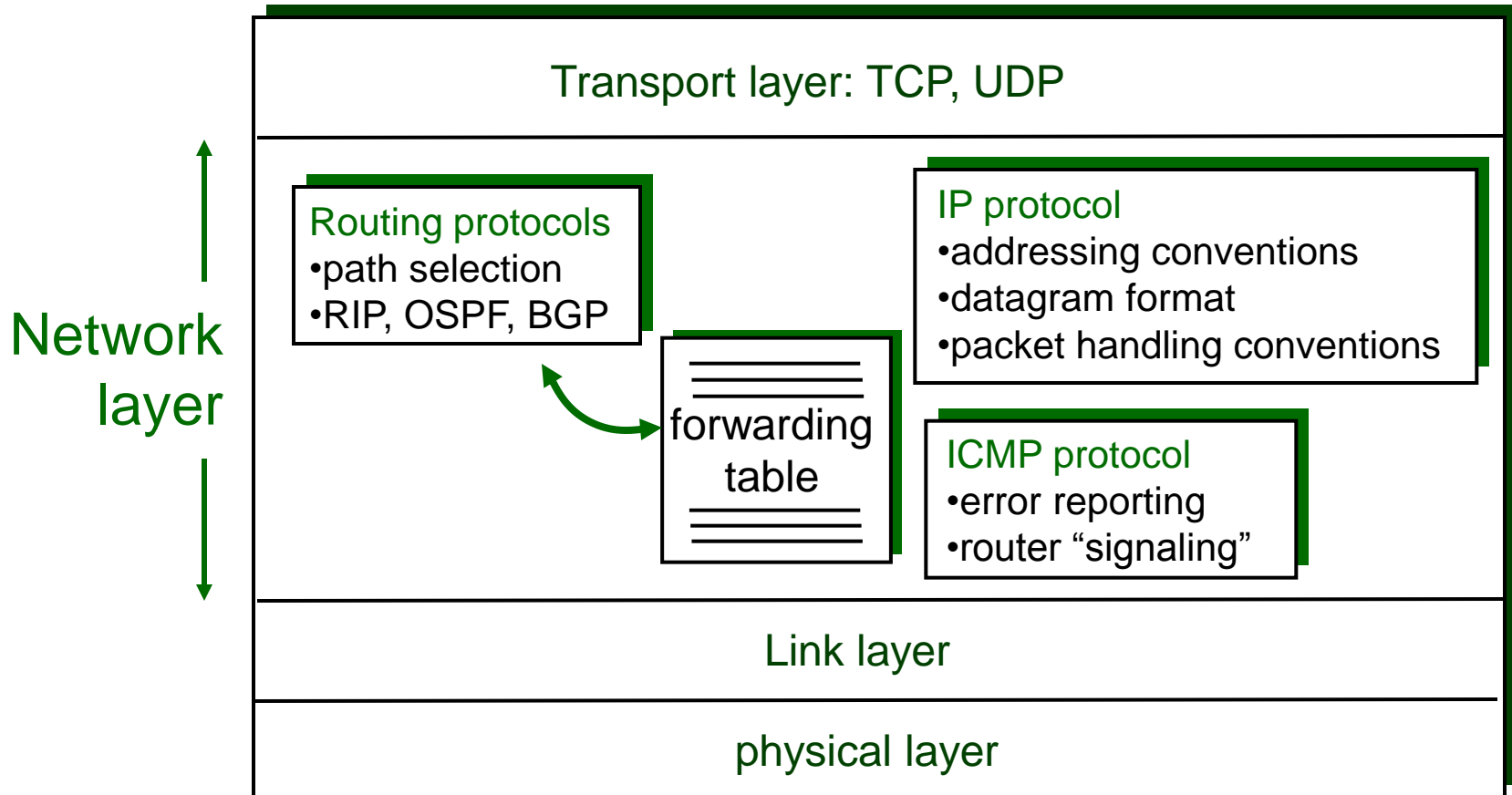
green packet  
experiences HOL blocking

# Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6

# The Internet Network layer

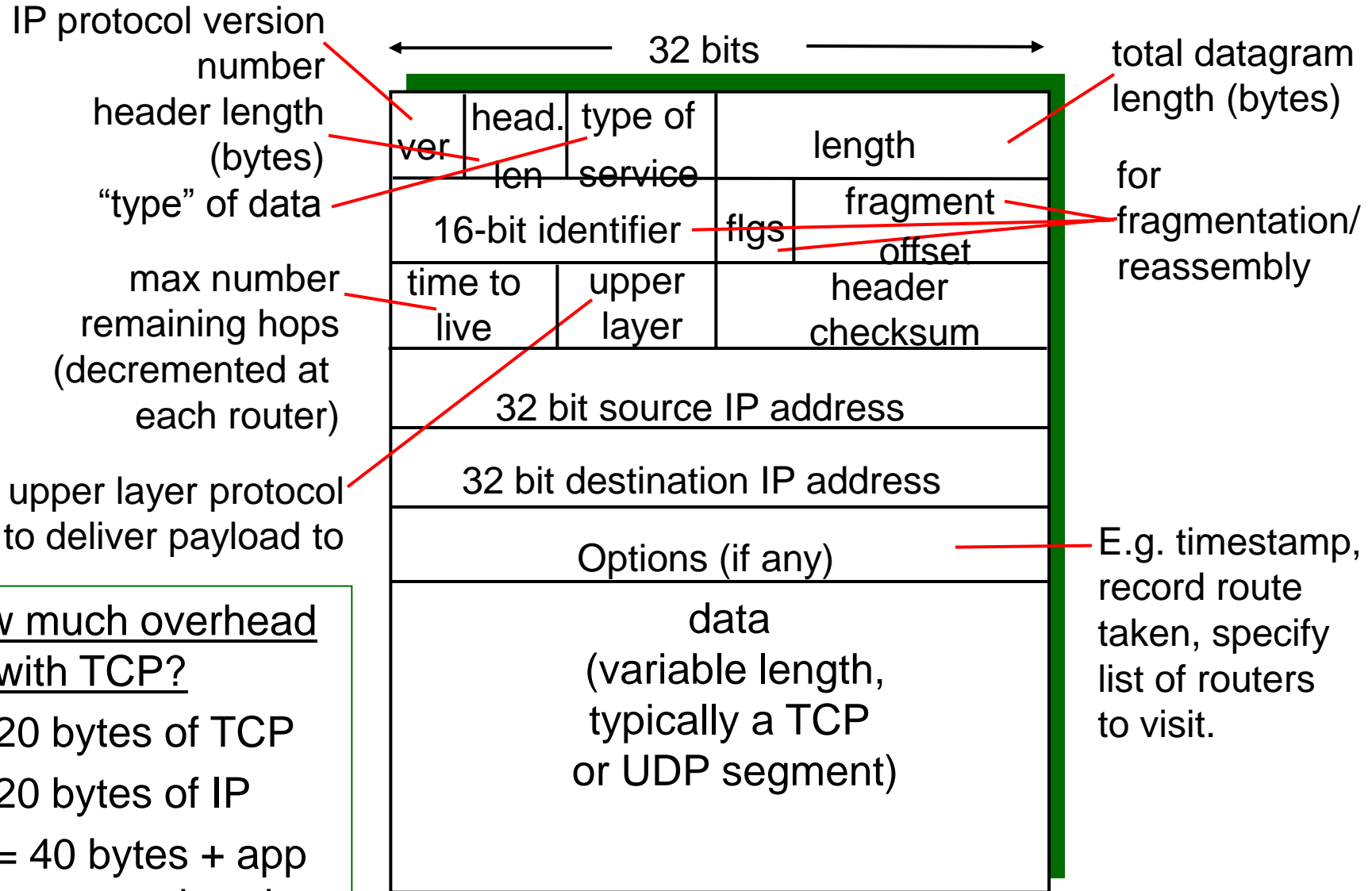
Host, router network layer functions:



# Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6

# IP datagram format

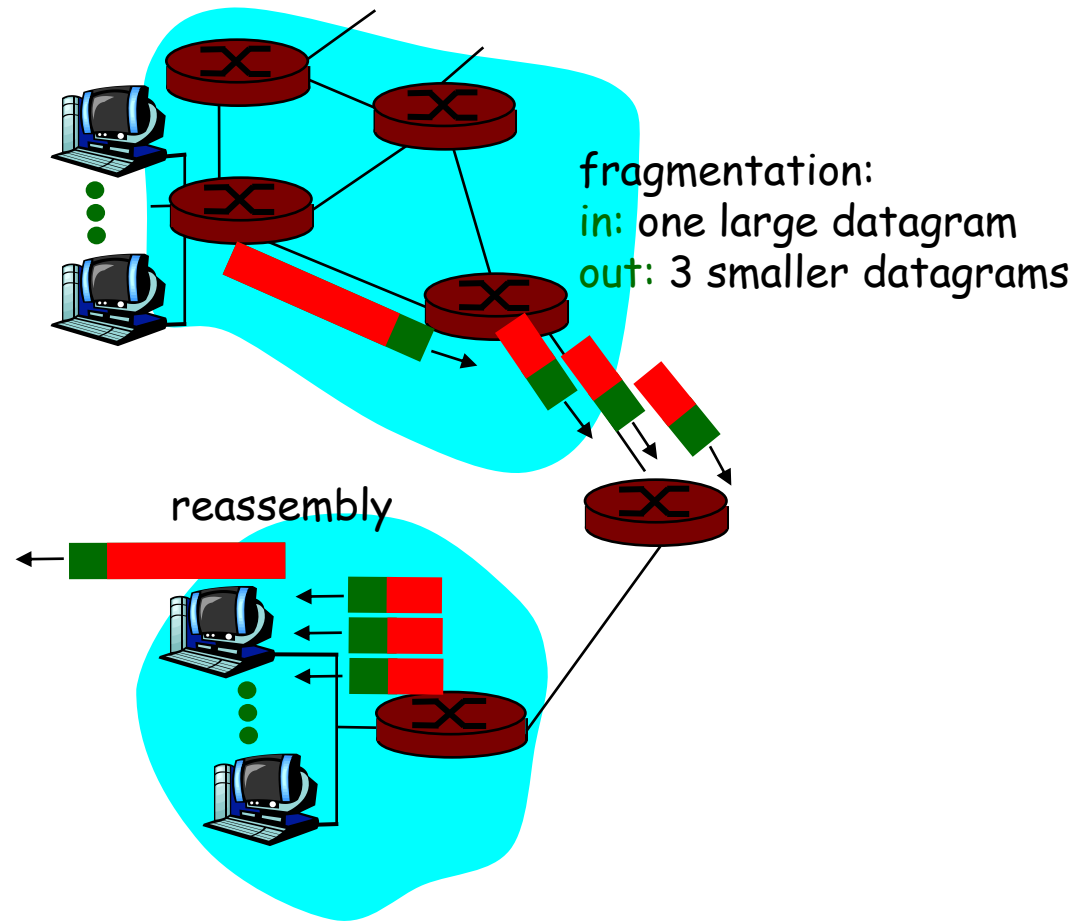


## how much overhead with TCP?

- 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

## Example

- 4000 byte datagram
- MTU = 1500 bytes

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

One large datagram becomes several smaller datagrams

1480 bytes in data field

offset =  
 $1480/8$

	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--

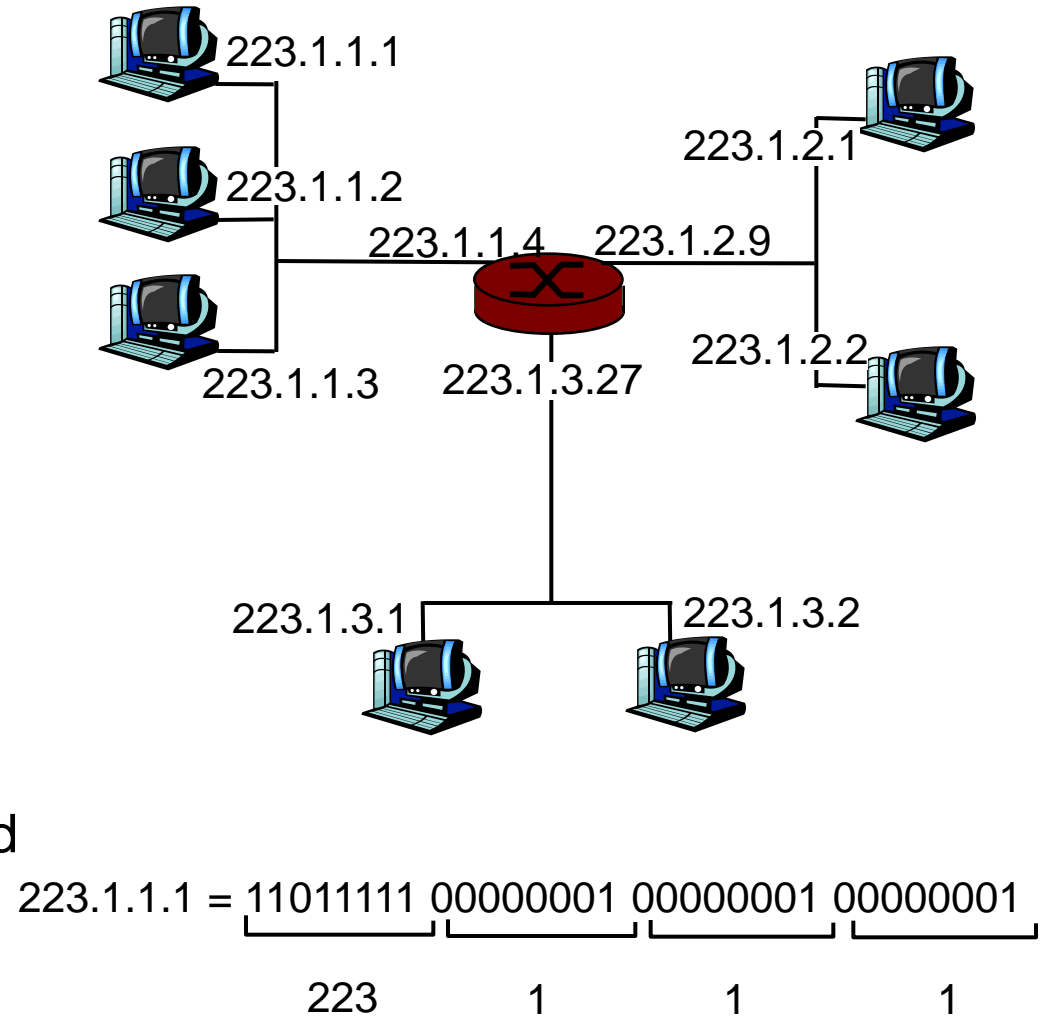
# Network Layer

- 4.1 Introduction
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6



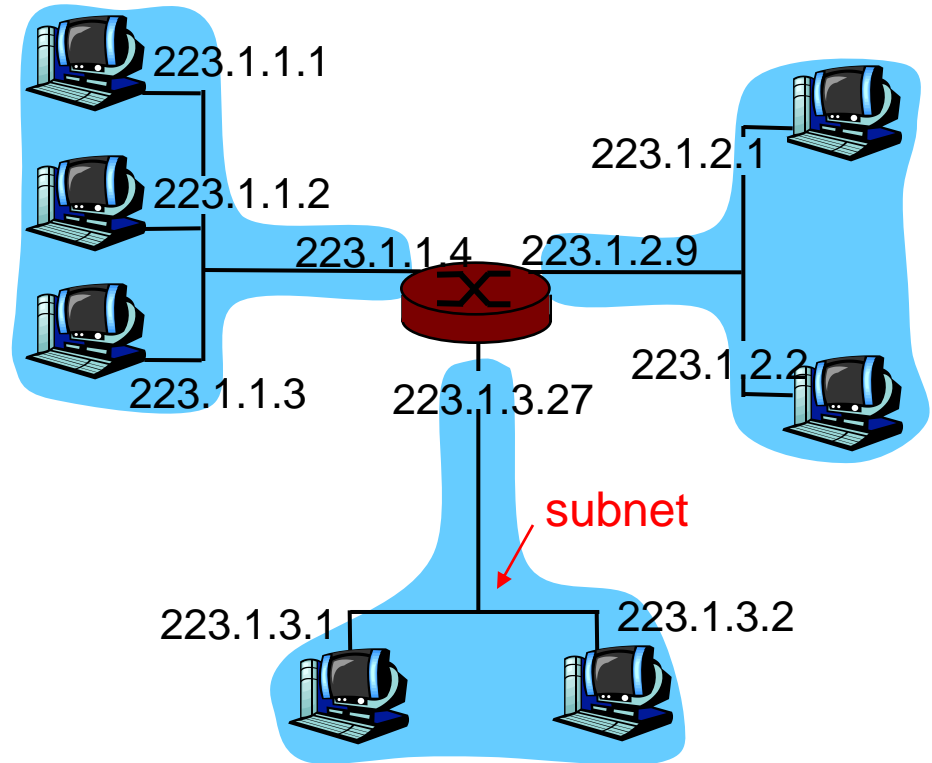
# IP Addressing: introduction

- 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

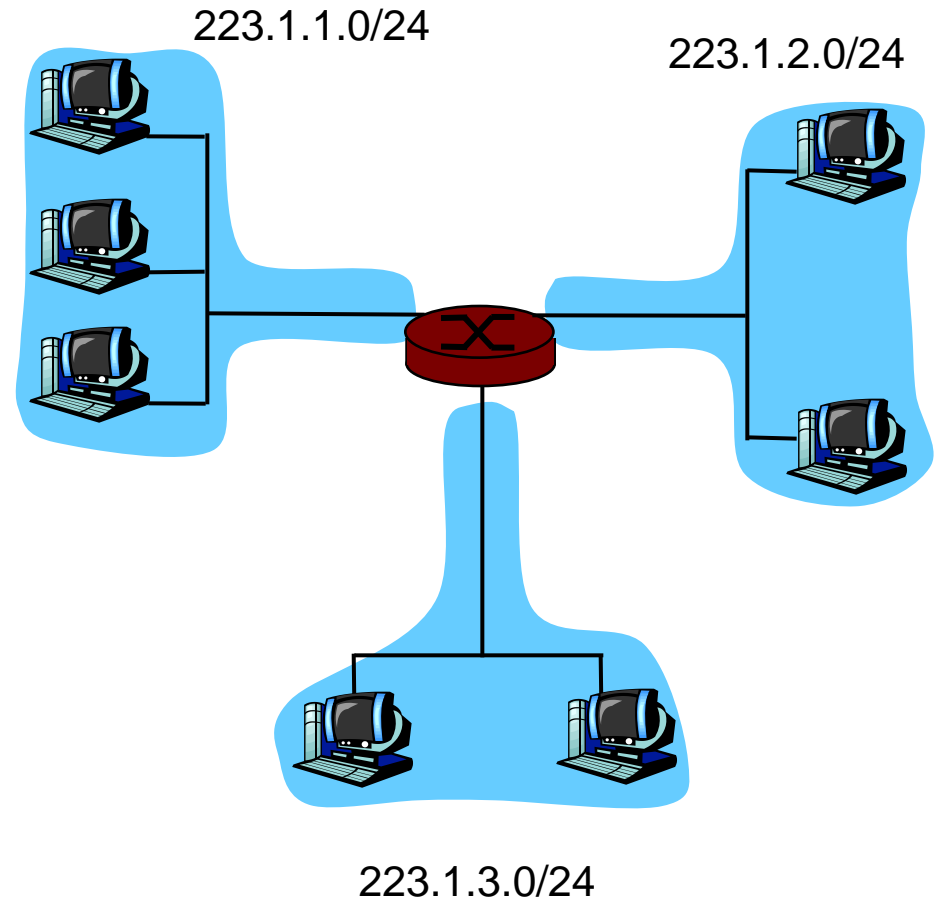


network consisting of 3 subnets

# Subnets

## Recipe

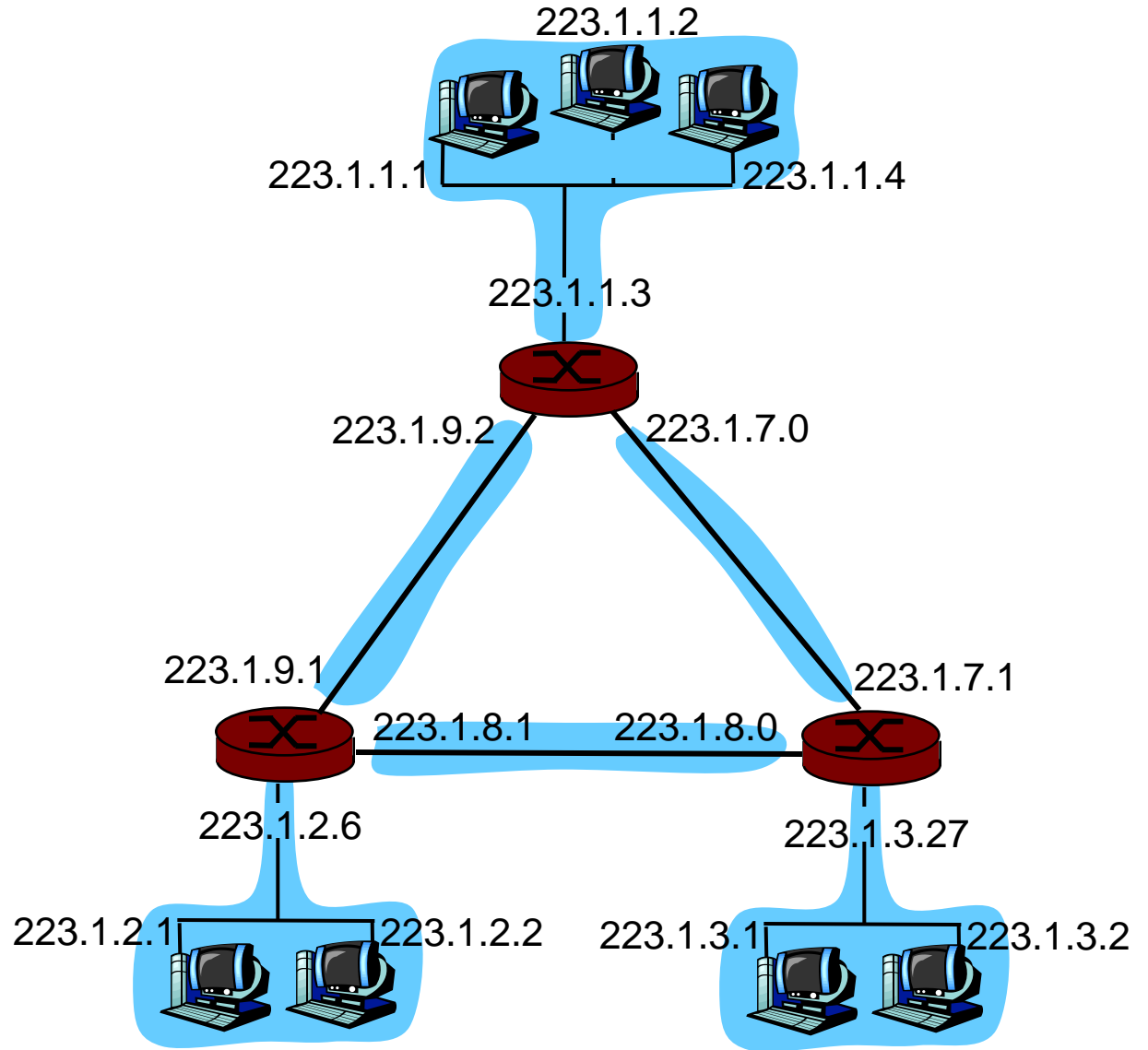
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a **subnet**.



Subnet mask: /24

# Subnets

How many?



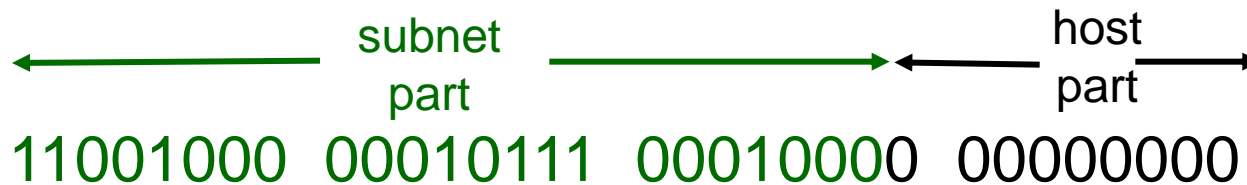
# IP addressing: Classful Network

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: Classless InterDomain Routing

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



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 as server
  - “plug-and-play”

# DHCP: Dynamic Host Configuration Protocol

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

Can renew its lease on address in use

Allows reuse of addresses (only hold address while connected 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



# IP addresses: how to get one?

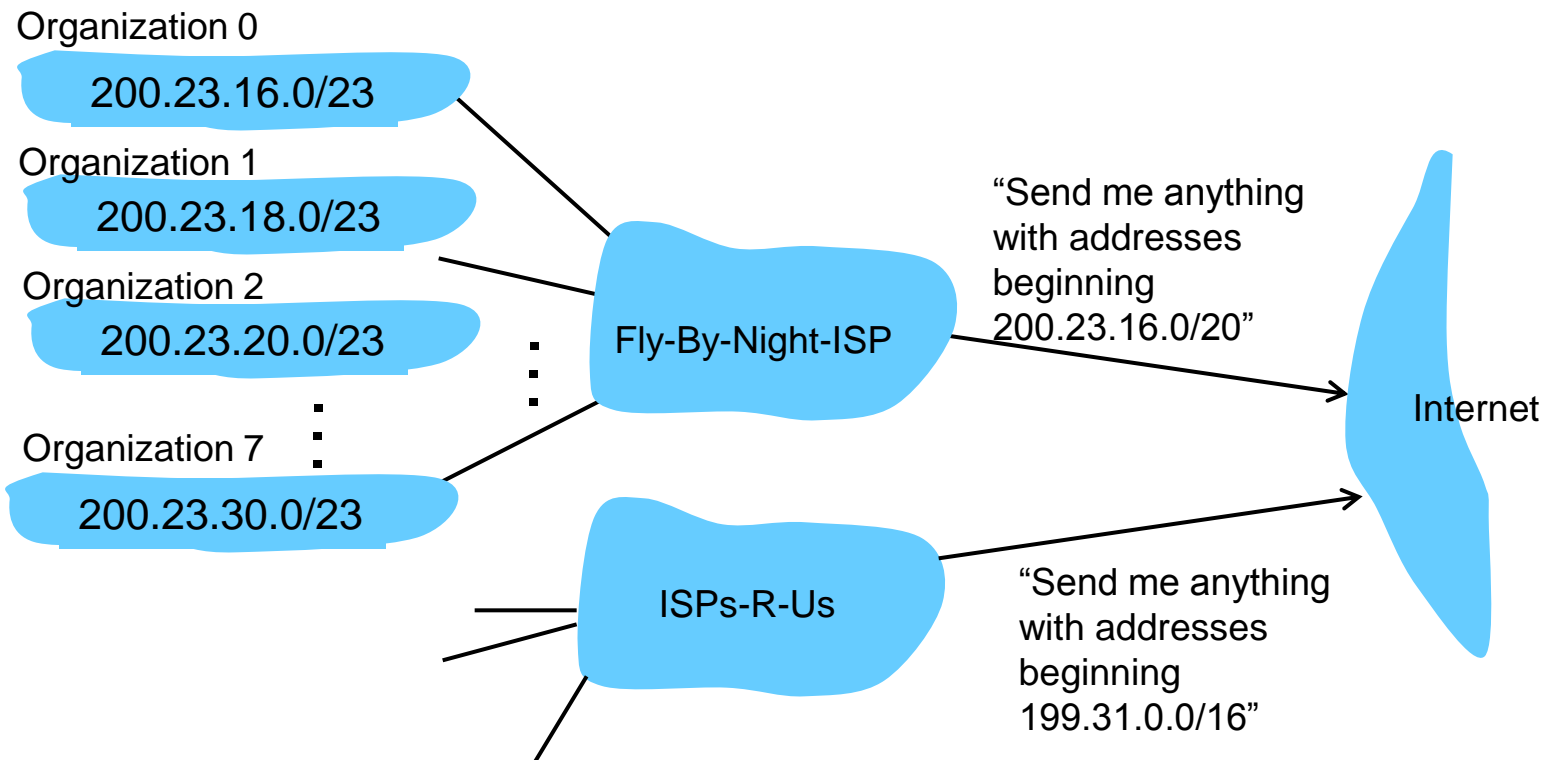
Q: How does *network* get subnet part of IP addr?

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

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

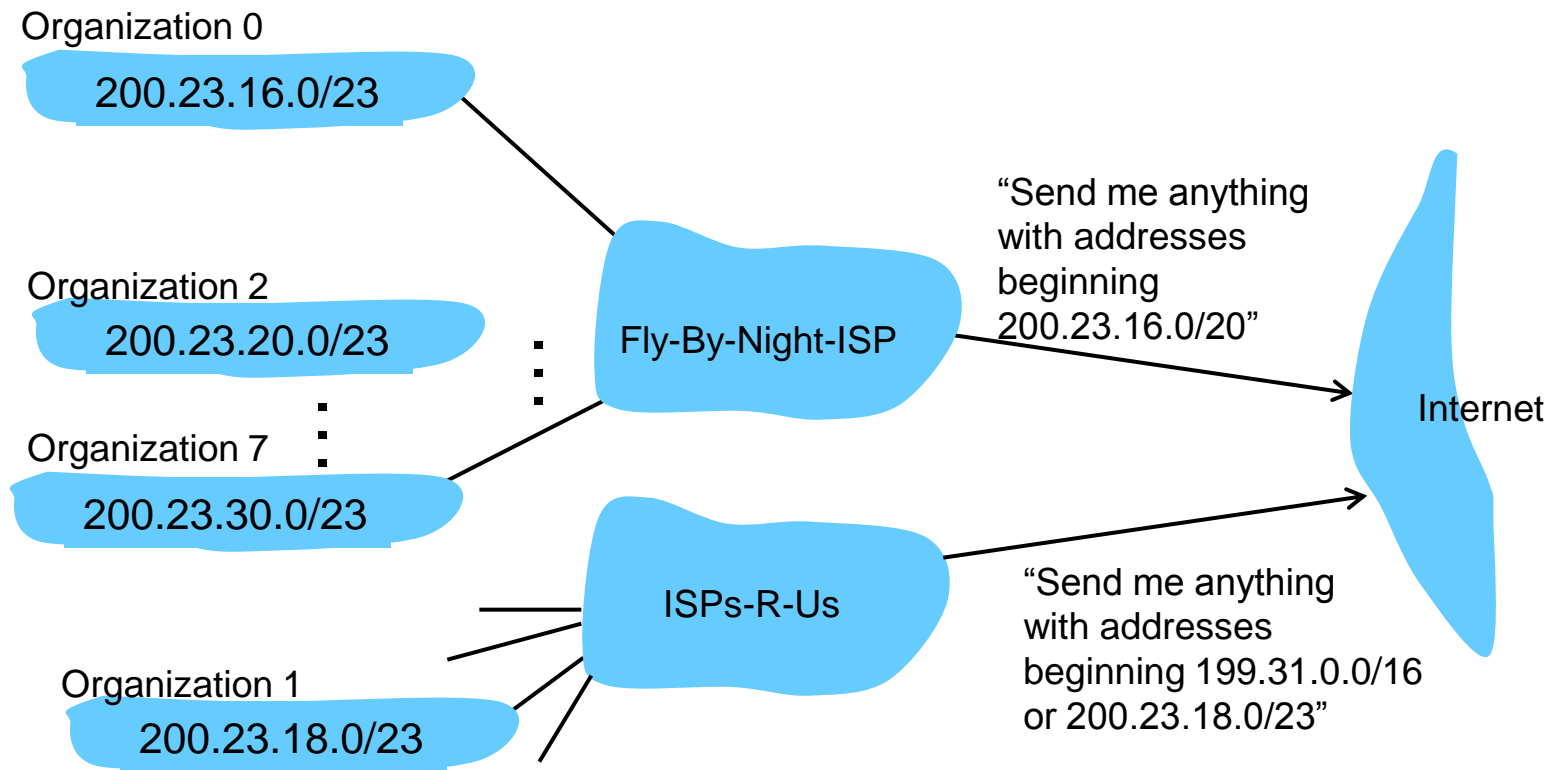
# Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



# Hierarchical addressing: more specific routes

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



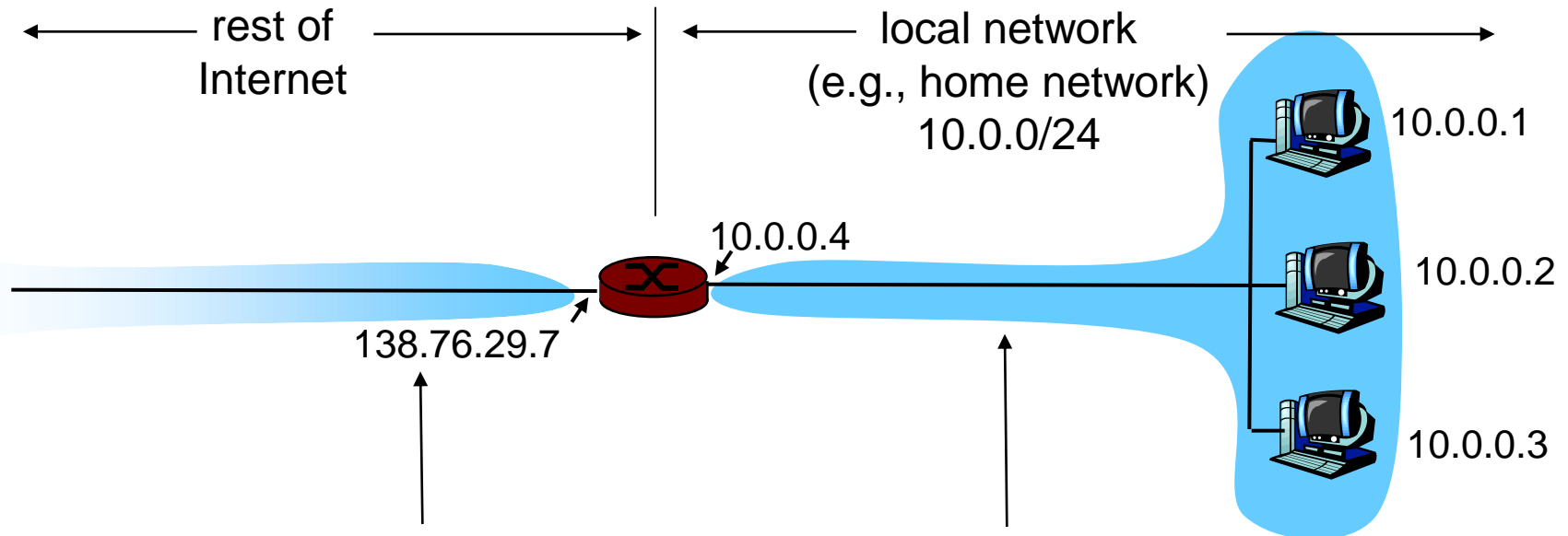
# IP addressing: the last word...

- **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

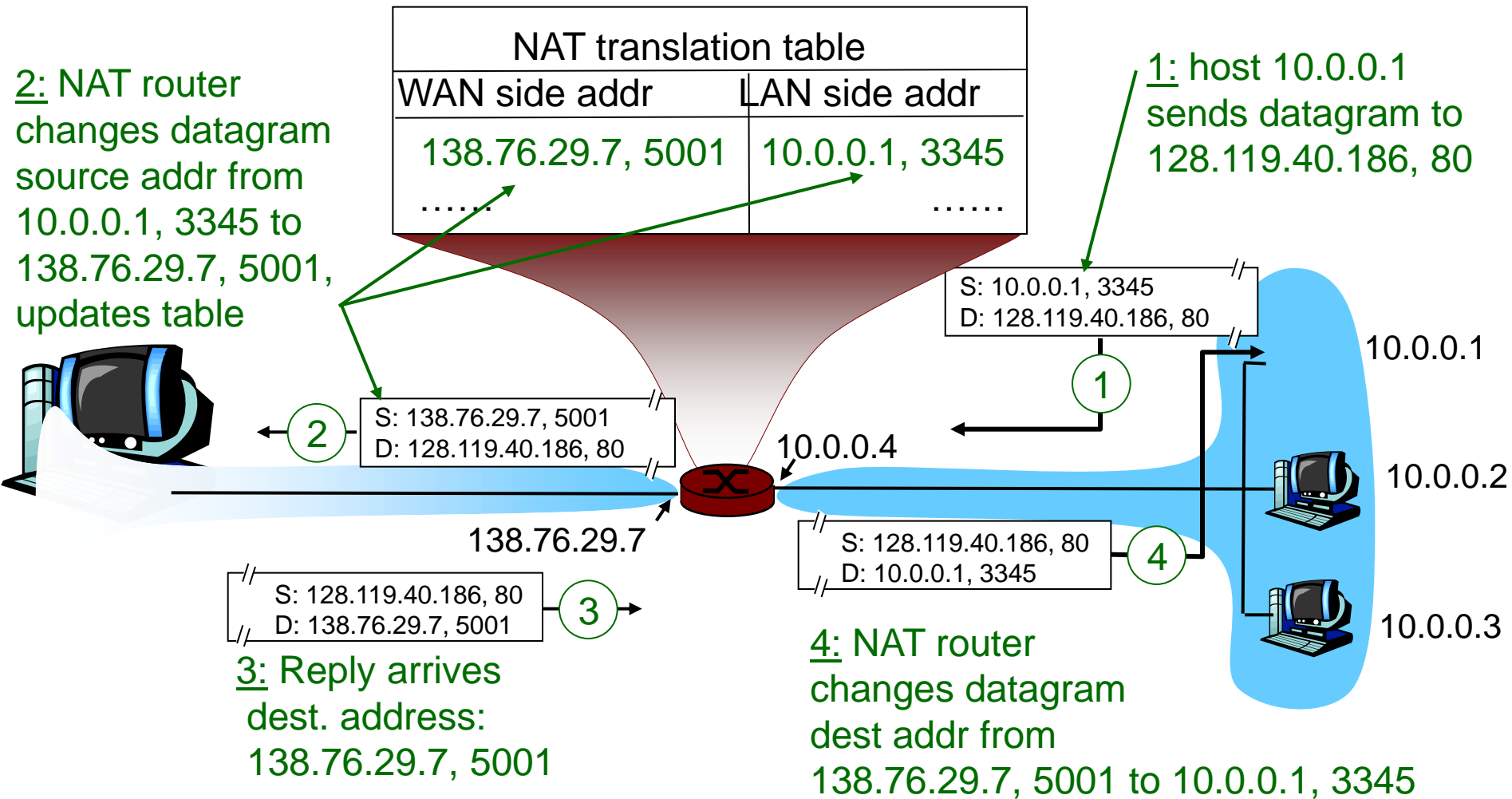
Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

# 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



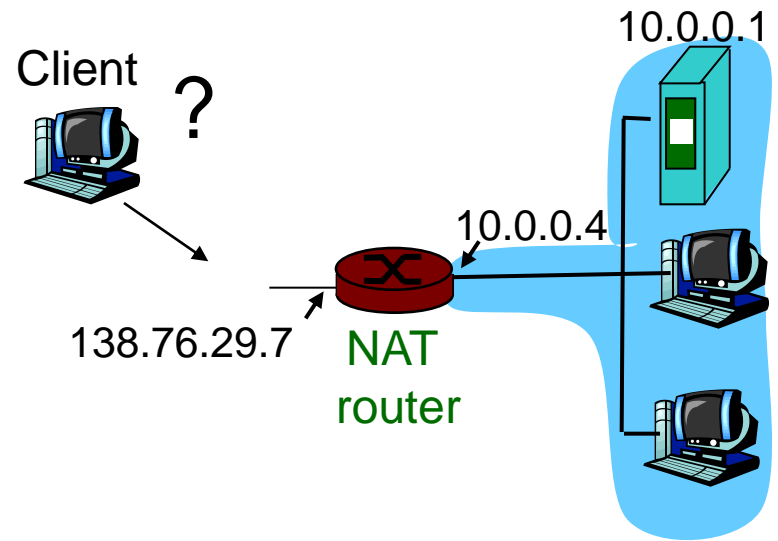


# NAT: Network Address Translation

- 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

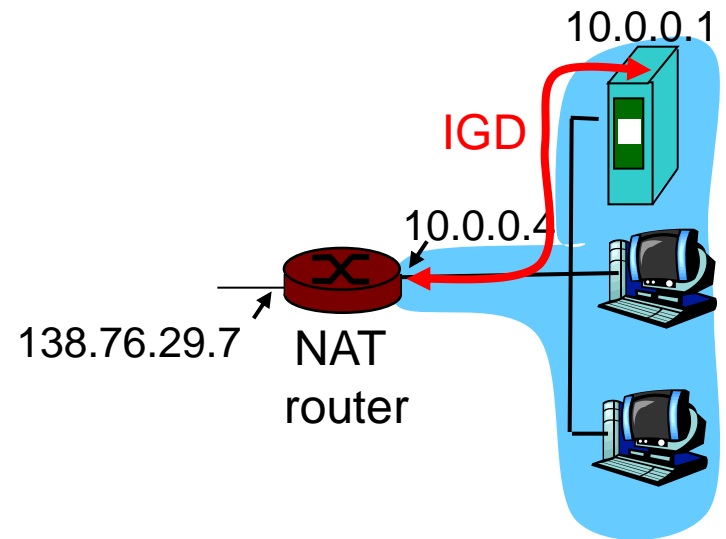
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server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



# NAT traversal problem

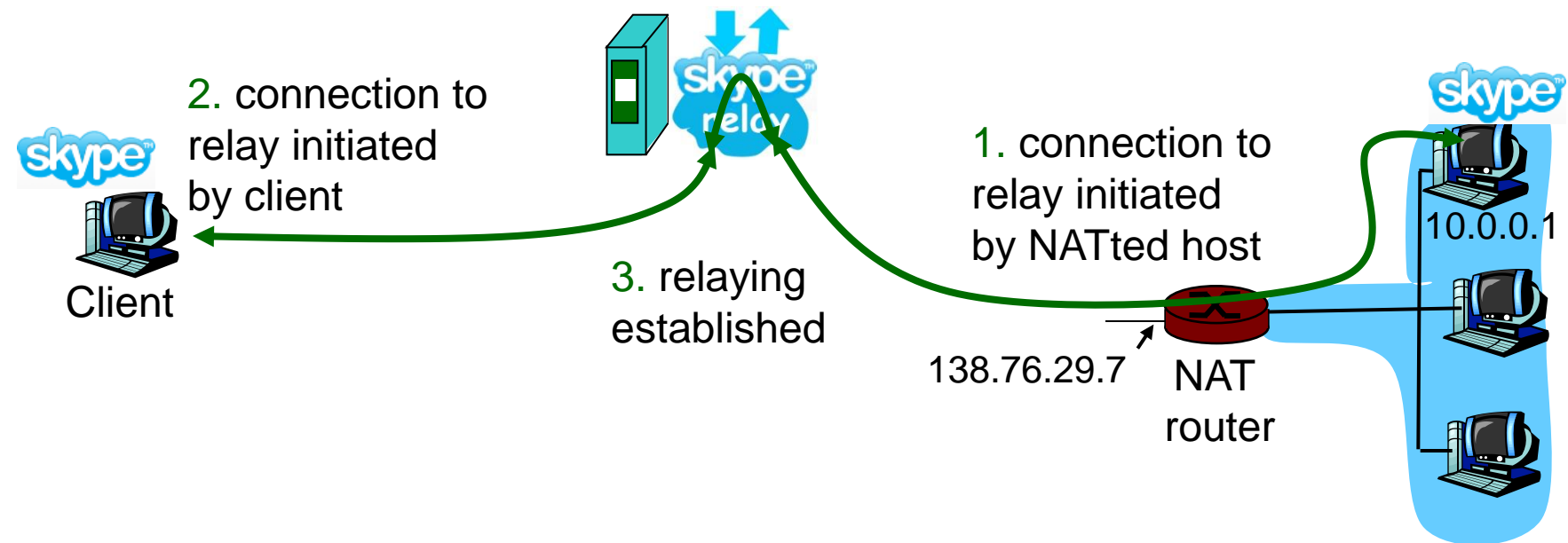
- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
  - ❖ learn public IP address (138.76.29.7)
  - ❖ add/remove port mappings (with lease times)

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

# 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

<u>Type</u>	<u>Code</u>	<u>description</u>
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



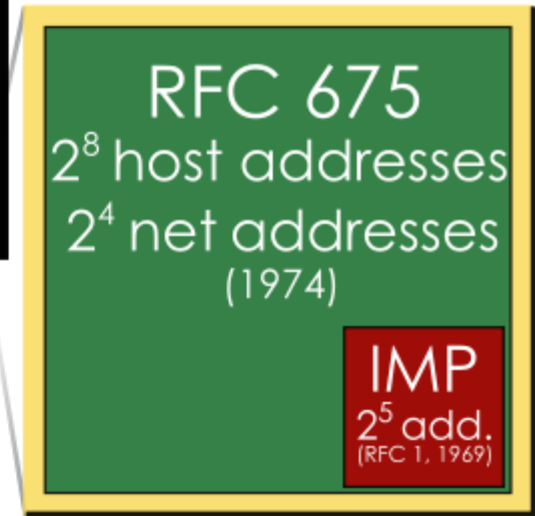
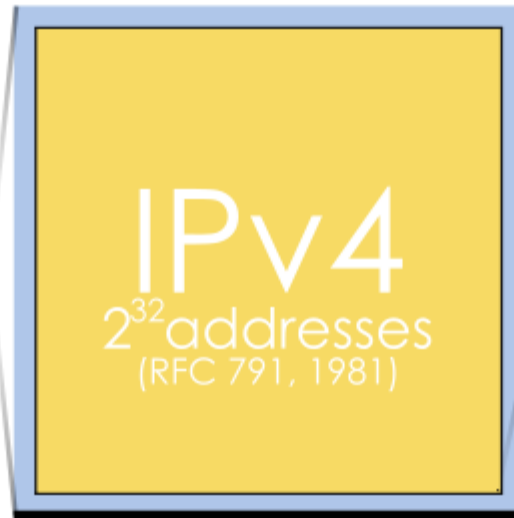
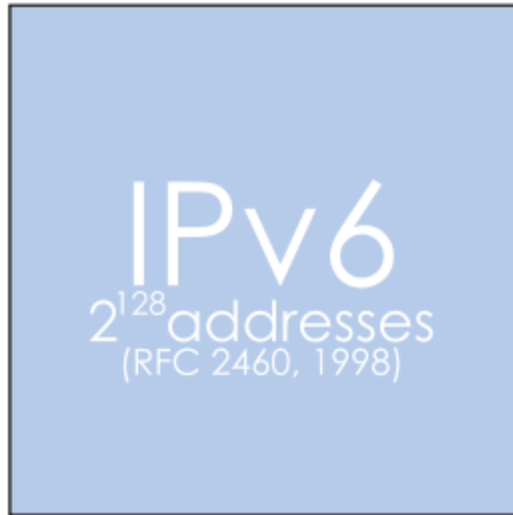
# IPv6

- **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



A diagram demonstrating the massive growth in address space under each protocol.

Each cascading block is a magnification of a tiny area in the preceding block, represented by a black square.

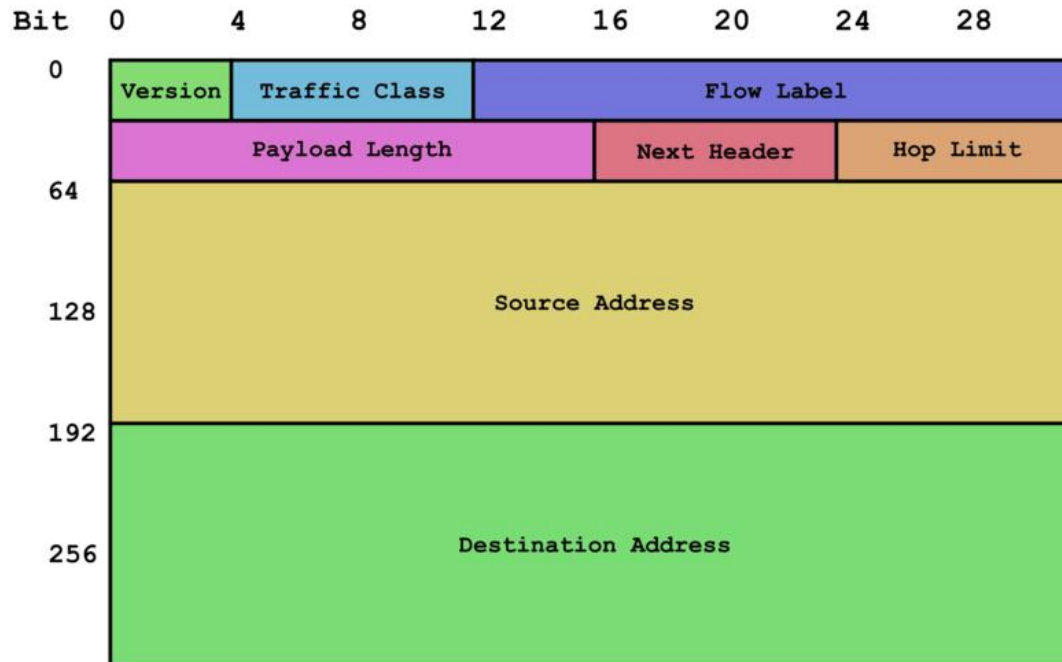
Image is to scale, except the black area is enlarged for ease of viewing

# IPv6 Header (Cont)

*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



Source: [http://commons.wikimedia.org/wiki/File:IPv6\\_header\\_rv1.png](http://commons.wikimedia.org/wiki/File:IPv6_header_rv1.png)

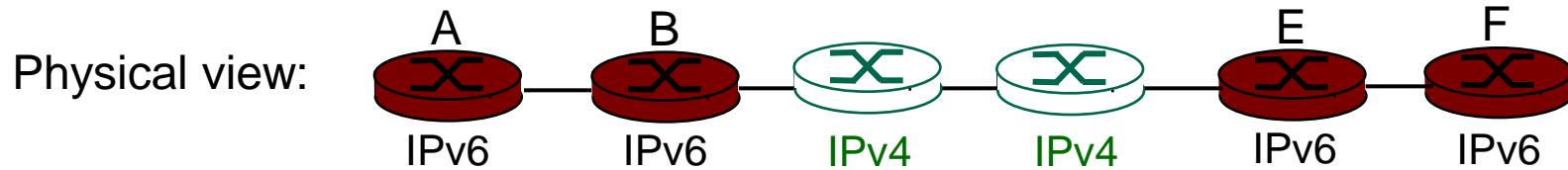
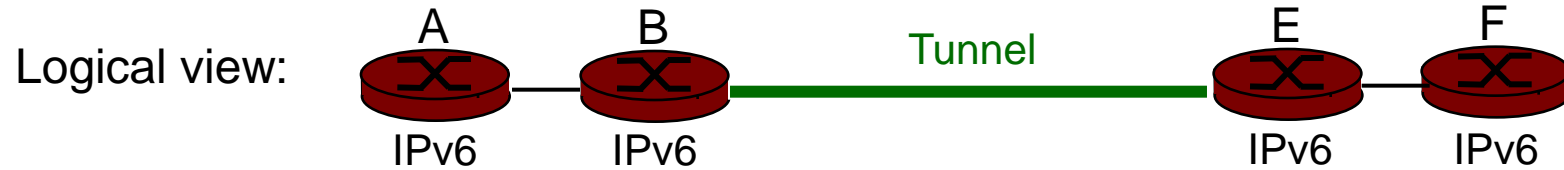
# Other Changes from IPv4

- *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

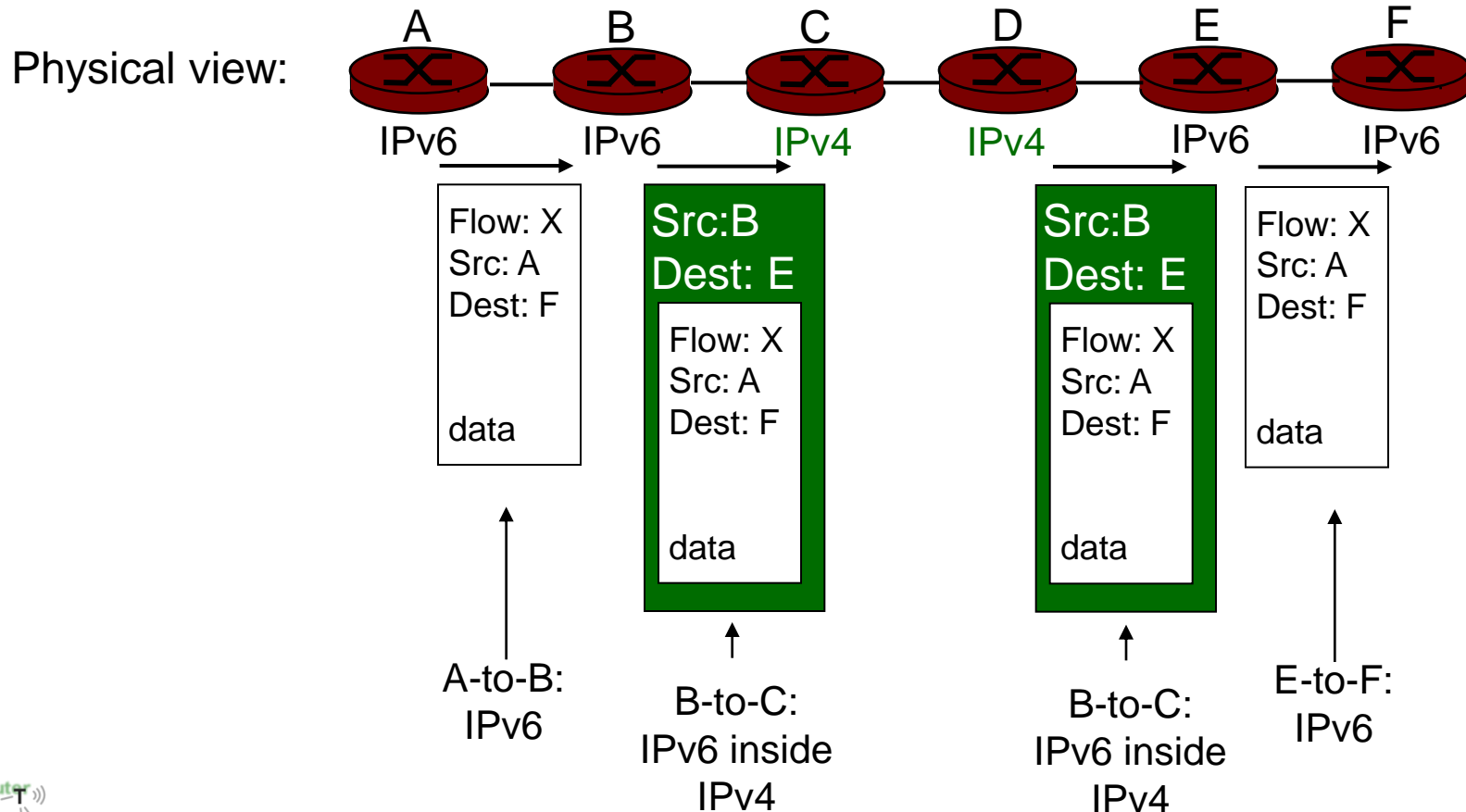
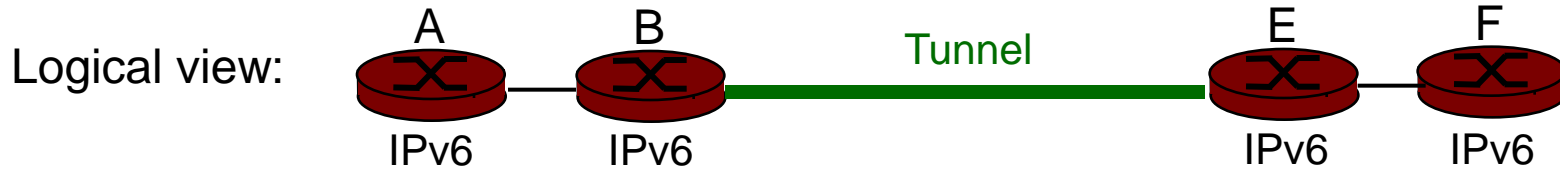
# Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously
  - no “flag days”
  - How will 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



# Tunneling



# 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



# Done for Today

- Thank you for your patience
- Any Questions?