Redes de Computadores

The Network Layer

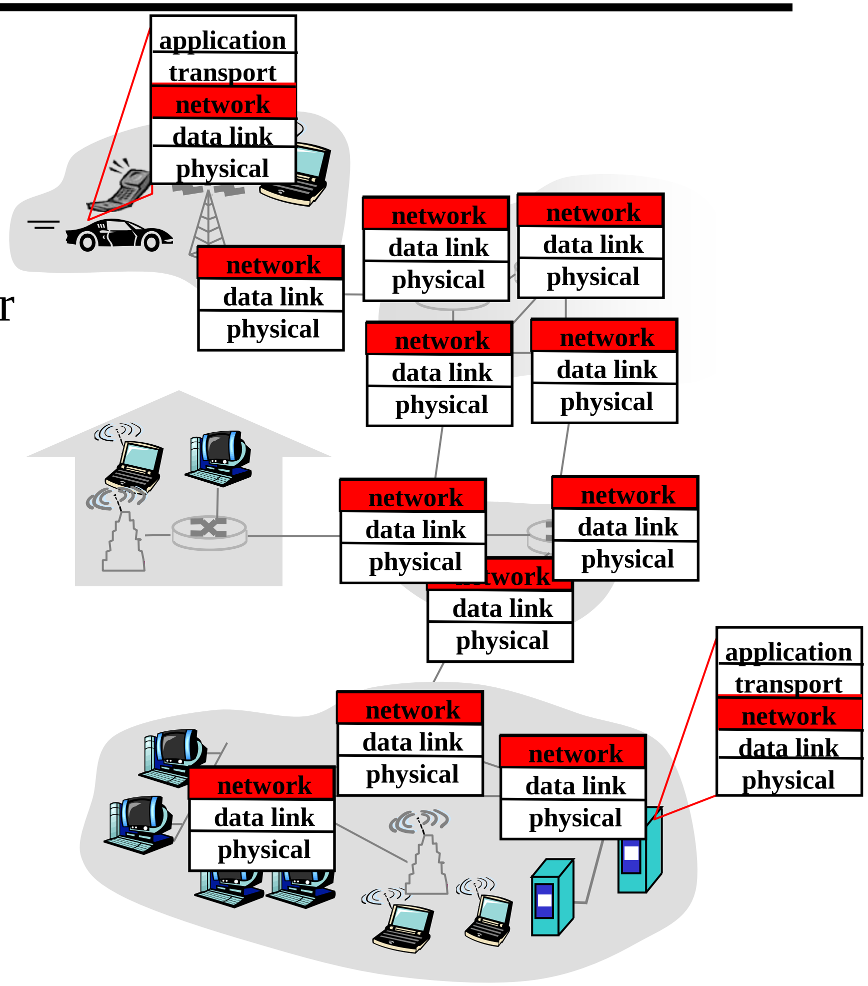
Manuel P. Ricardo

Faculdade de Engenharia da Universidade do Porto

- What are the main functions of the network layer?
- What are the differences between virtual circuit and datagram networks?
- How is forwarding handled in both type of networks?
- What are the main building blocks of a router and their functions?
- What are the formats of IP addresses?
- How to form subnets?
- What services are provided by ARP, ICMP, DHCP and NAT? How do these protocols work?
- What are differences the between IPv4 and IPv6?

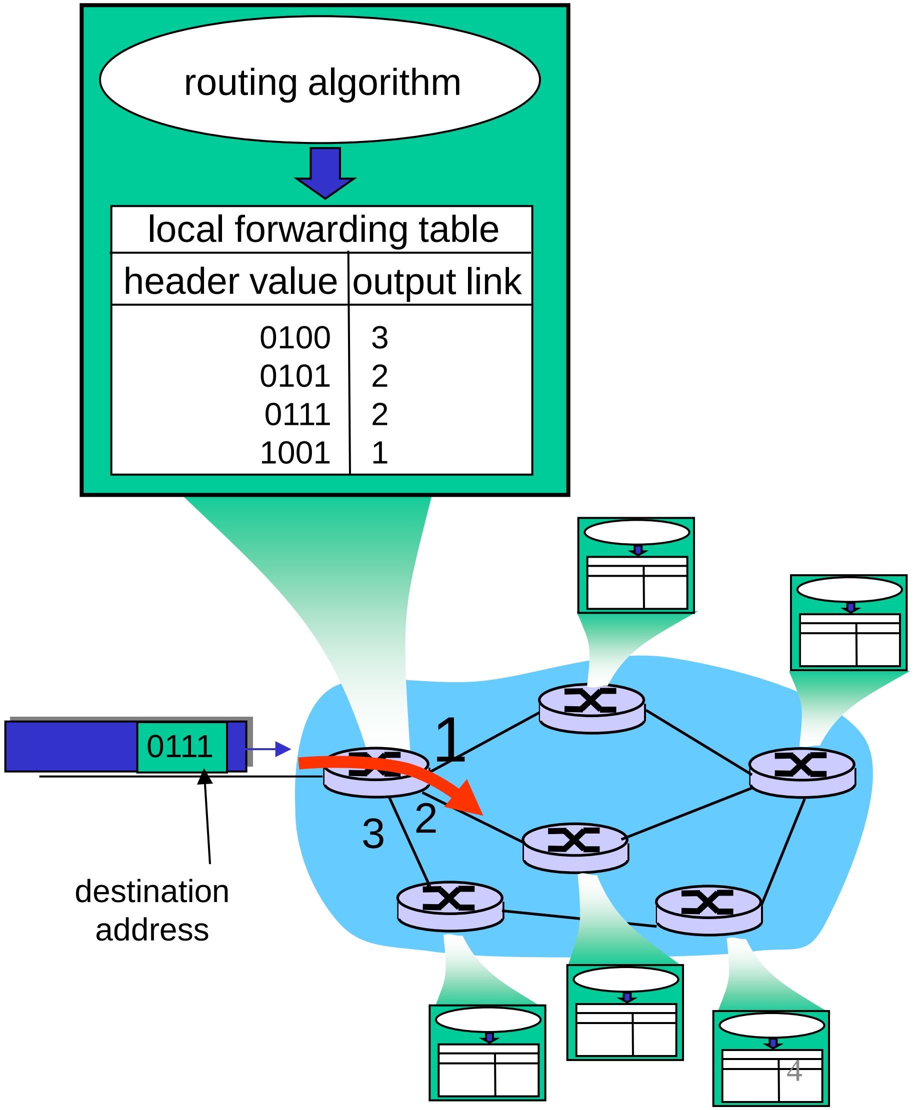
Network Layer Overview

- Network layer
  - » transports packets (datagrams)
  - » from sending host to receiving host
  - » functions located in every host and router
- Sender
  - » encapsulates transport data into packets
  - » generates packets
- Receiver
  - » receives packets
  - » delivers data to transport layer
- Router
  - » Receives packets from input line
  - » examines network layer header
  - » forwards packets through adequate output line(s)



Network Layer – Main Functions

- Forwarding
  - » router forwards packet from input port to output port
- Routing
  - » determine route taken by packets, from source to destination
  - » algorithms, shortest path



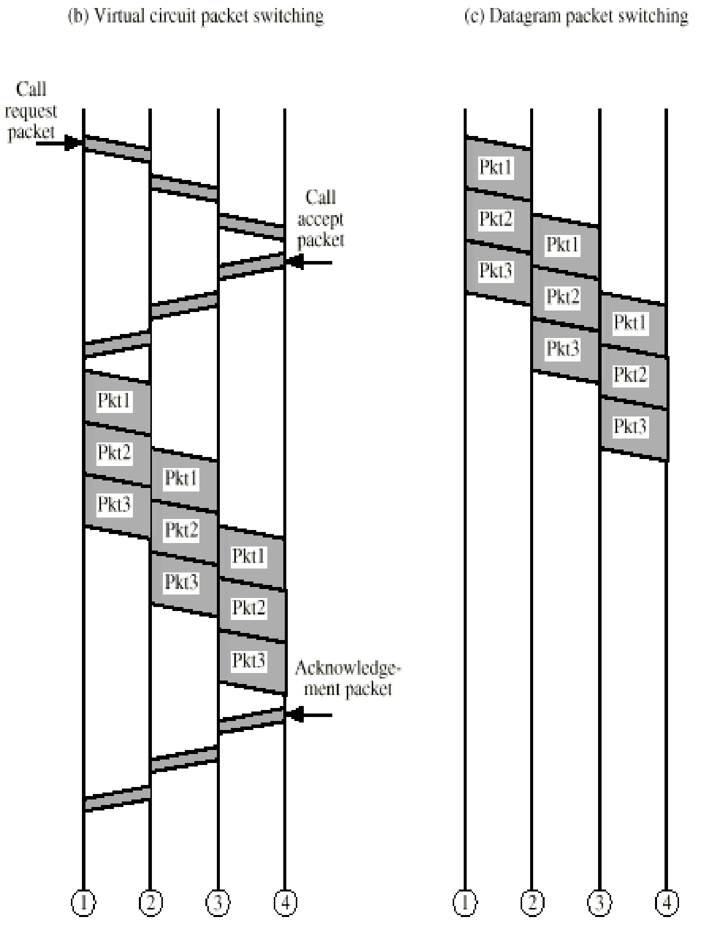
Virtual Circuits and Datagram Networks

Network Layer – ***Connection*** *and* ***Connectionless*** *Service*

Services provided by network layer

- » Datagram network
  ➔ connectionless service

- » Virtual Circuit network
  ➔ connection oriented service

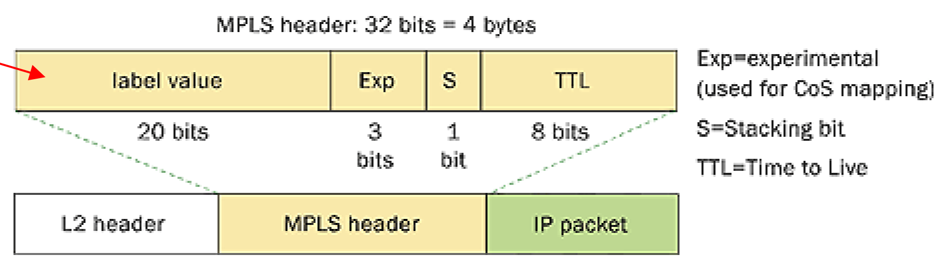


Virtual Circuit (VC)

- Phases

  circuit establishment ➔ data transference ➔ circuit termination

- Packet carries identifier of Virtual Circuit

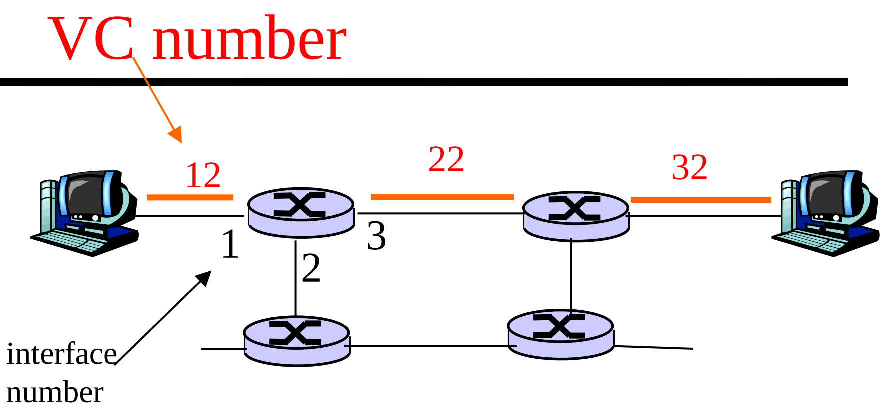


- Path defined from source to destination

  sequence of VC identifiers, one for each link along path

- Router
  - » maintains "state" for every supported circuit
  - » may allocate resources (bandwidth, buffers) per Virtual Circuit

VC - Forwarding Table



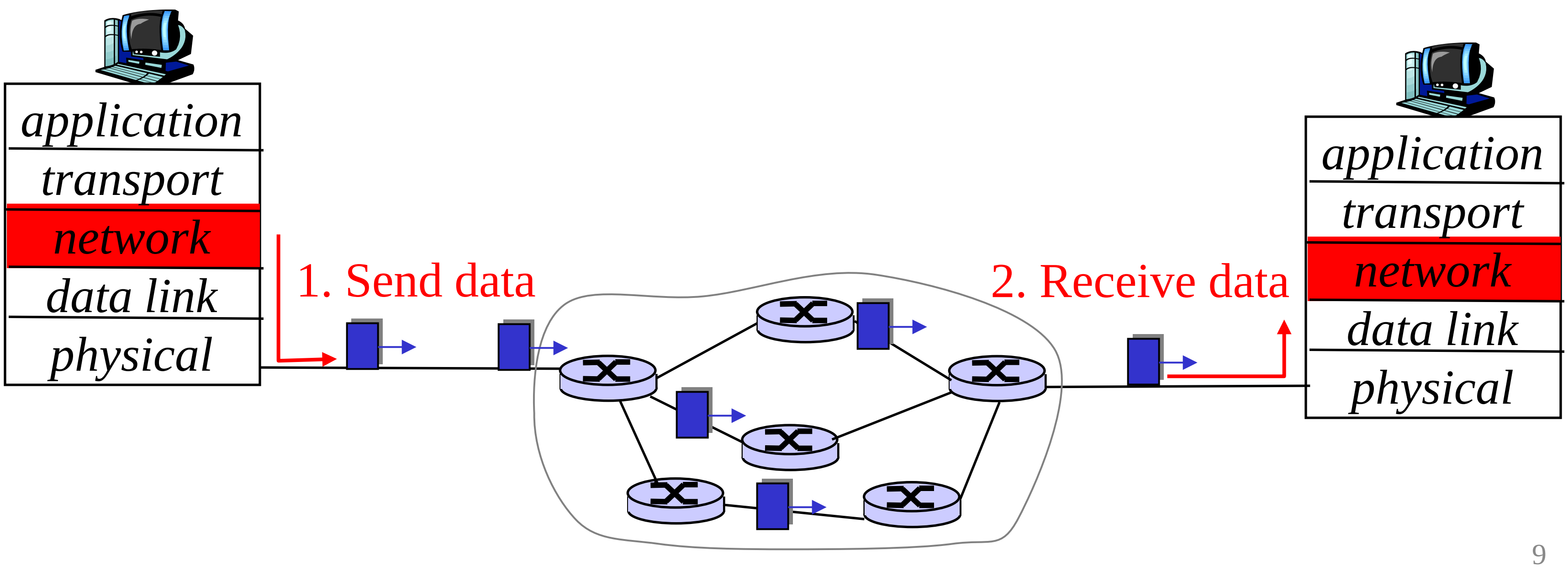
Forwarding table in
northwest router:

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

Routers maintain connection state information!

Datagram Networks

- No circuit establishment; no circuit concept
- Packets
  - » forwarded using destination host address
  - » packets between same source-destination pair may follow different paths



Forwarding Table

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

2^{32} possible entries in IPv4

- How to reduce the number of entries in the forwarding table?

Longest *Prefix Matching*

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Examples. Which Interface?

DA: 11001000 00010111 00010110 10100001 → 0

DA: 11001000 00010111 00011000 10101010 → 1,2 → 1

longest prefix

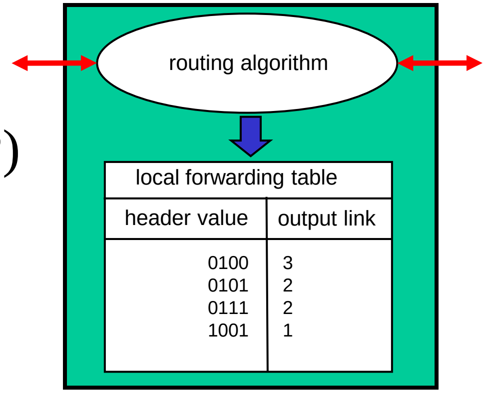
Virtual-Circuit versus Datagram Networks

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

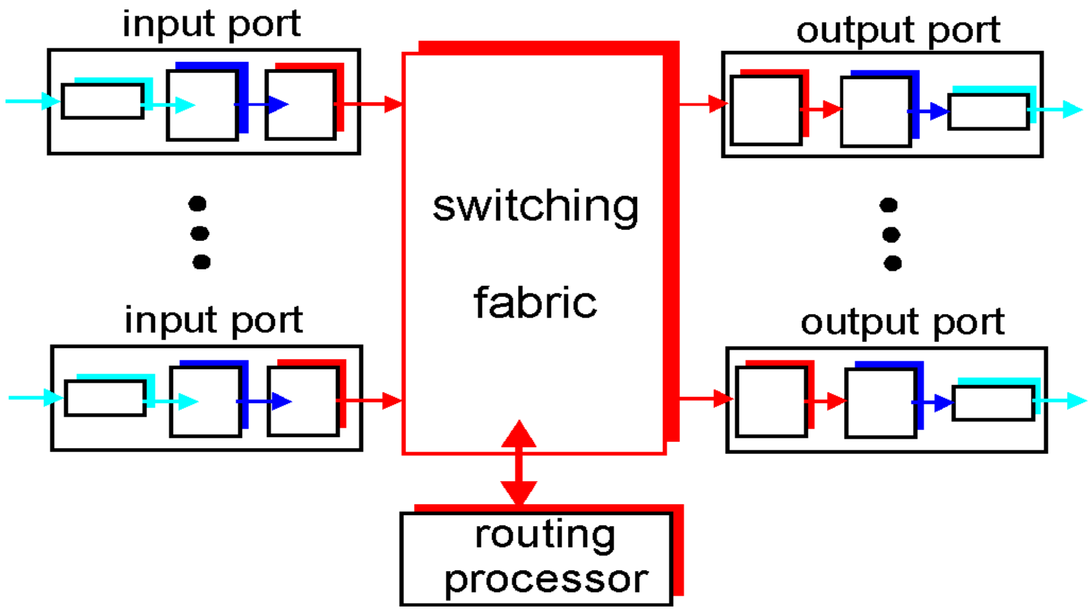
Router Architecture

Router Architecture - Overview

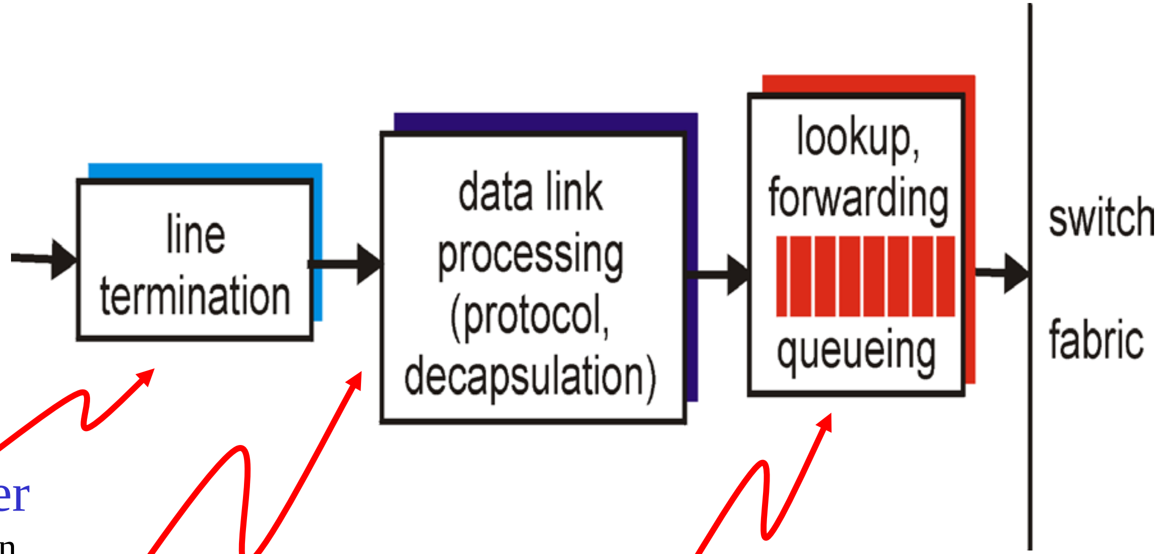
- Router main functions
  - » Run routing algorithms and protocols (RIP, OSPF, BGP)
  - » Forward packets from input link to output link



- Main components
  - » input port
  - » output port
  - » switching fabric
  - » routing processor



Input Port Functions



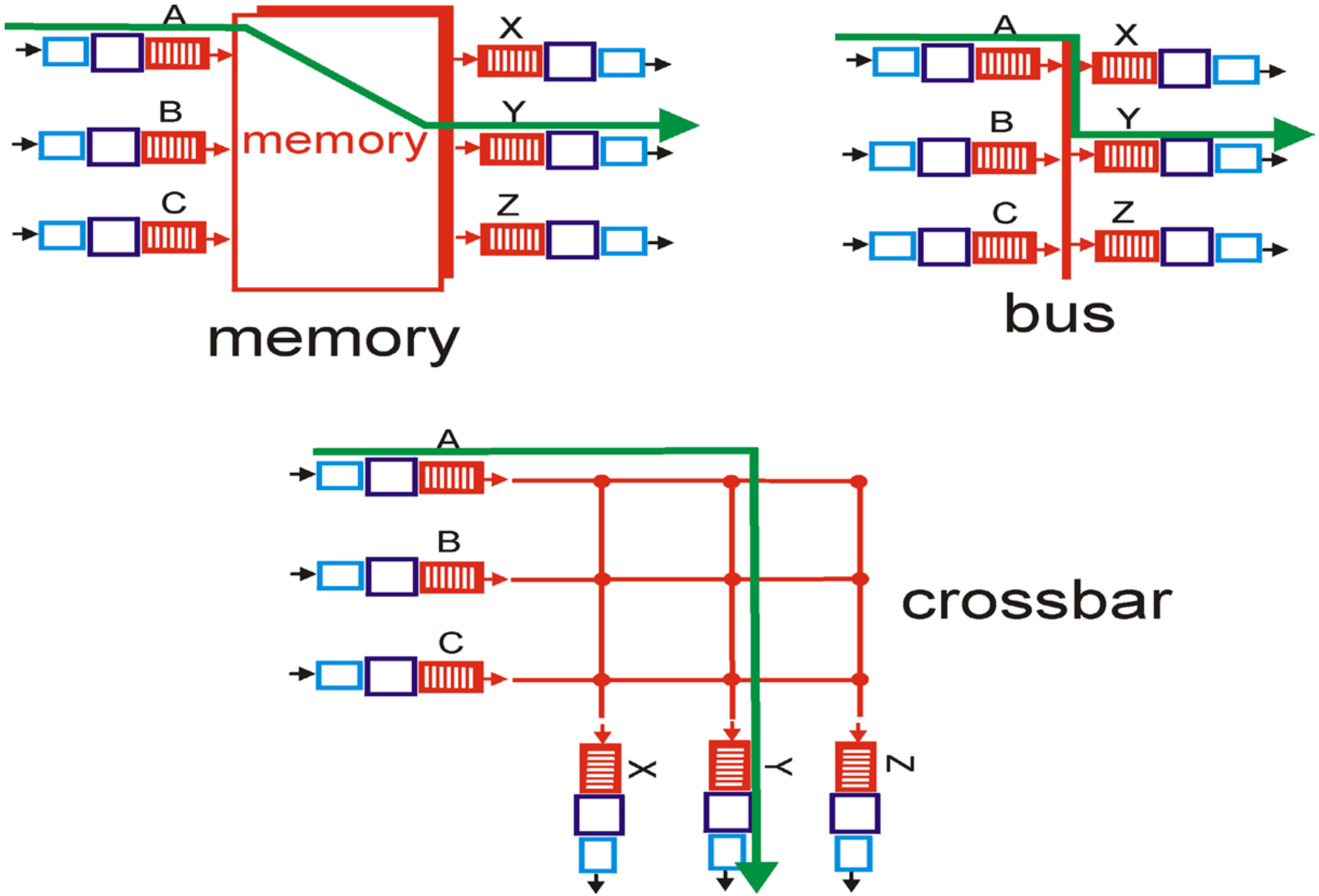
Physical layer
bit-level reception

Data link layer
e.g., Ethernet

Decentralized switching

- Lookup output port ,
  using forwarding table in input port memory
- Goal ➔ processing at 'line speed'
- Queuing,
  if packets arrive faster than forwarding rate

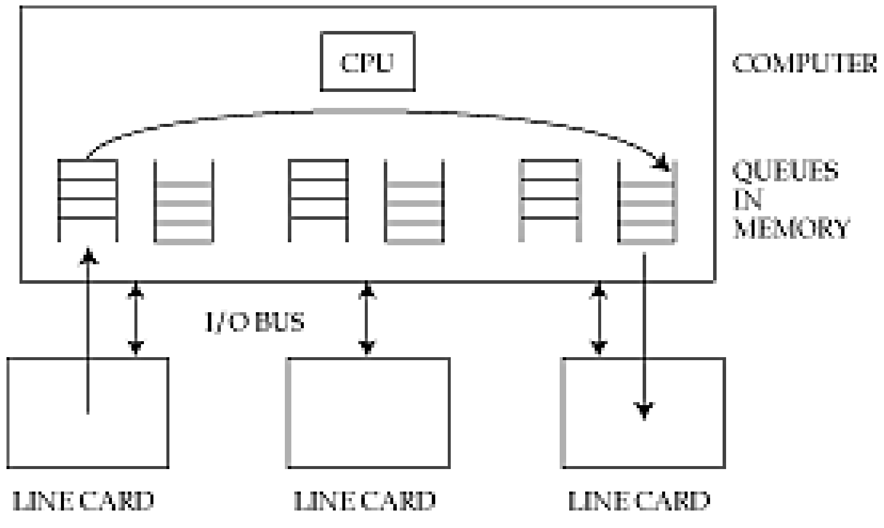
Switching Fabrics



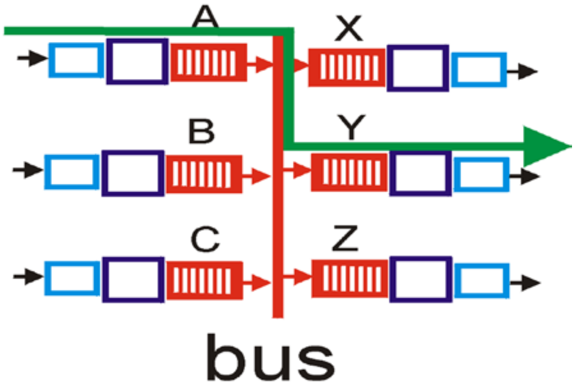
Switching Via Computer Memory

First generation router

- Traditional computers; switching controlled by CPU
- Each packet
  - » copied into system's memory
  - » transferred twice through the bus



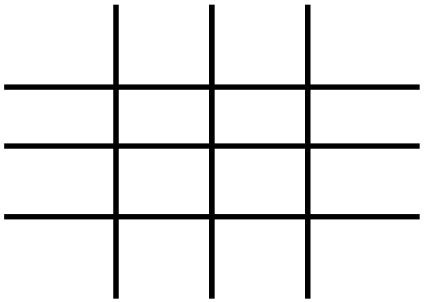
Switching Via a Bus



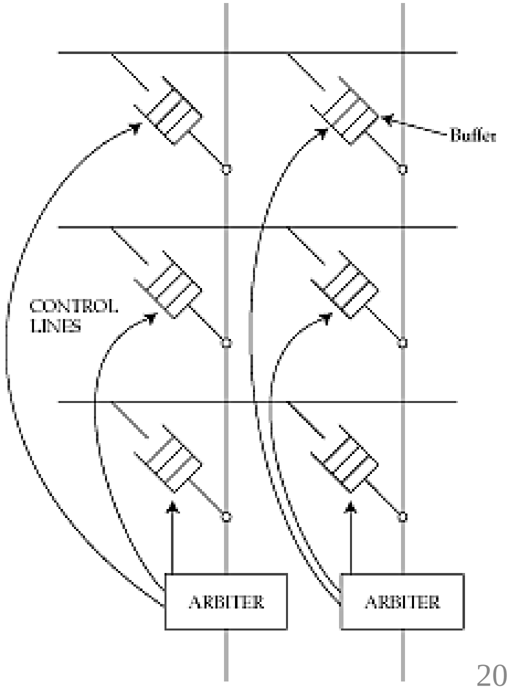
- Packet switched through a shared bus
- Direct transference of packets from input to output lines
- Bus contention ➔ switching rate limited by bus bandwidth

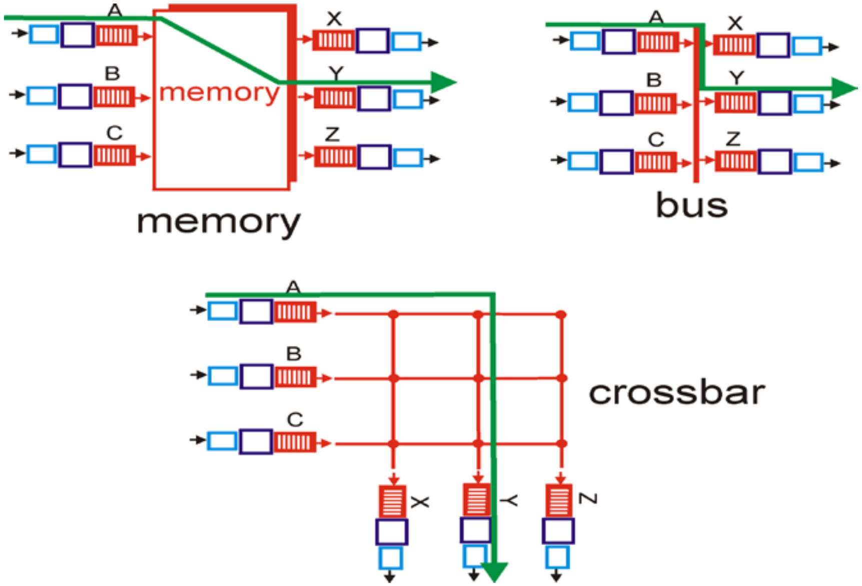
Switching Via a Crossbar

- 2N buses
- Enables simultaneous packet transferences



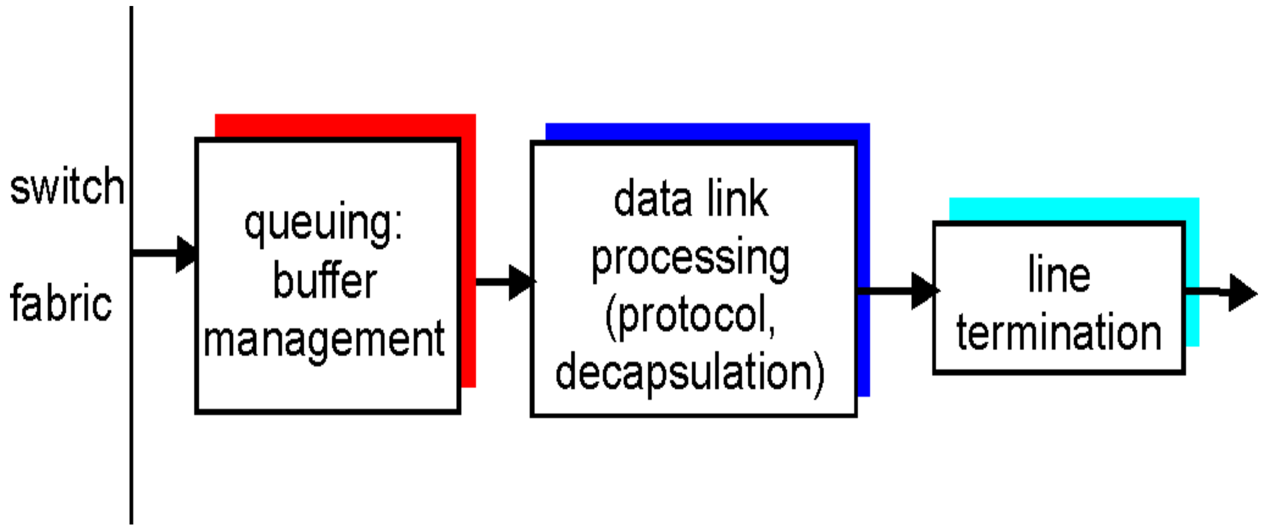
- Crossbar may contain internal buffers
- Overcomes bus bandwidth limitations





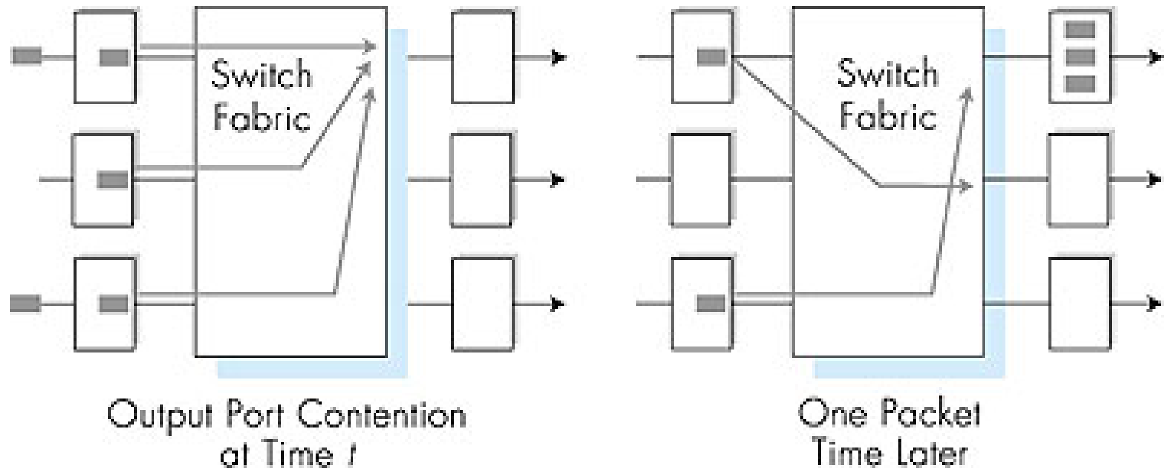
- What is the best switching fabrics? Why?

Output Port



- Buffering
- Scheduling discipline
  chooses among queued datagrams for transmission

Output Port Queuing

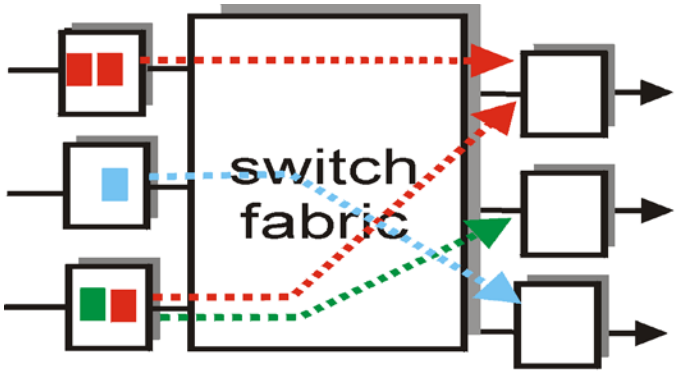


- Buffering when
  arrival rate via switch
  exceeds speed of output line

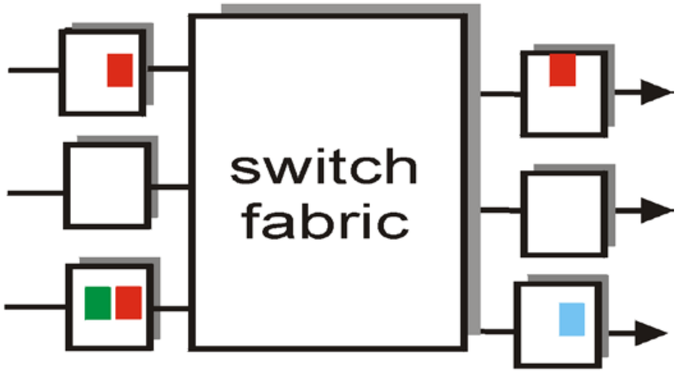
- Queuing delay and loss
  » due to overflow of the buffer in output port

Input Port Queuing

- Queuing may occur at input queues if
  - fabric slower than input ports combined
- Head-of-the-Line (HOL) blocking
  - datagram at head of queue prevents other datagrams to move forward



output port contention at time t - only one red packet can be transferred

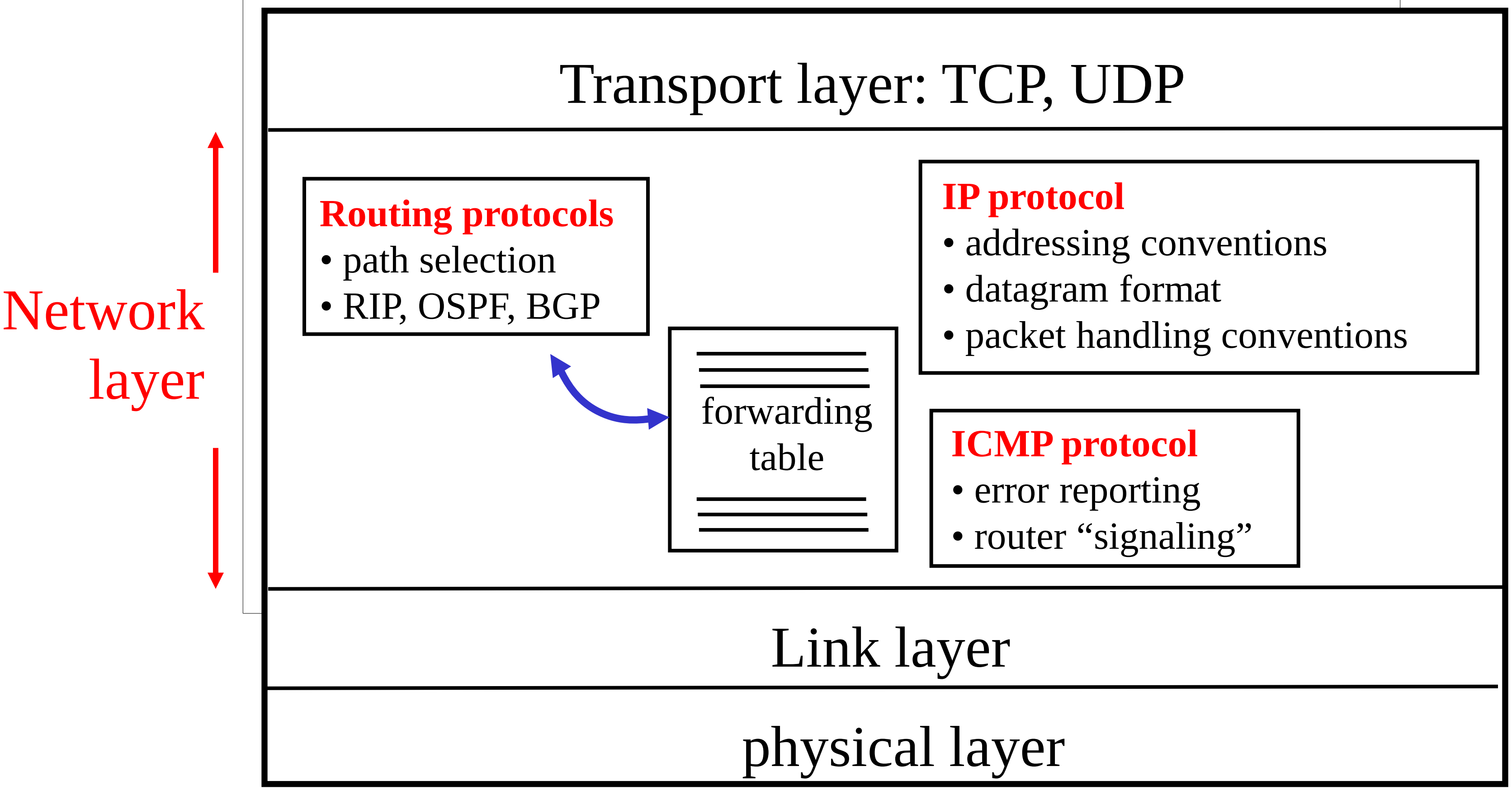


green packet experiences HOL blocking

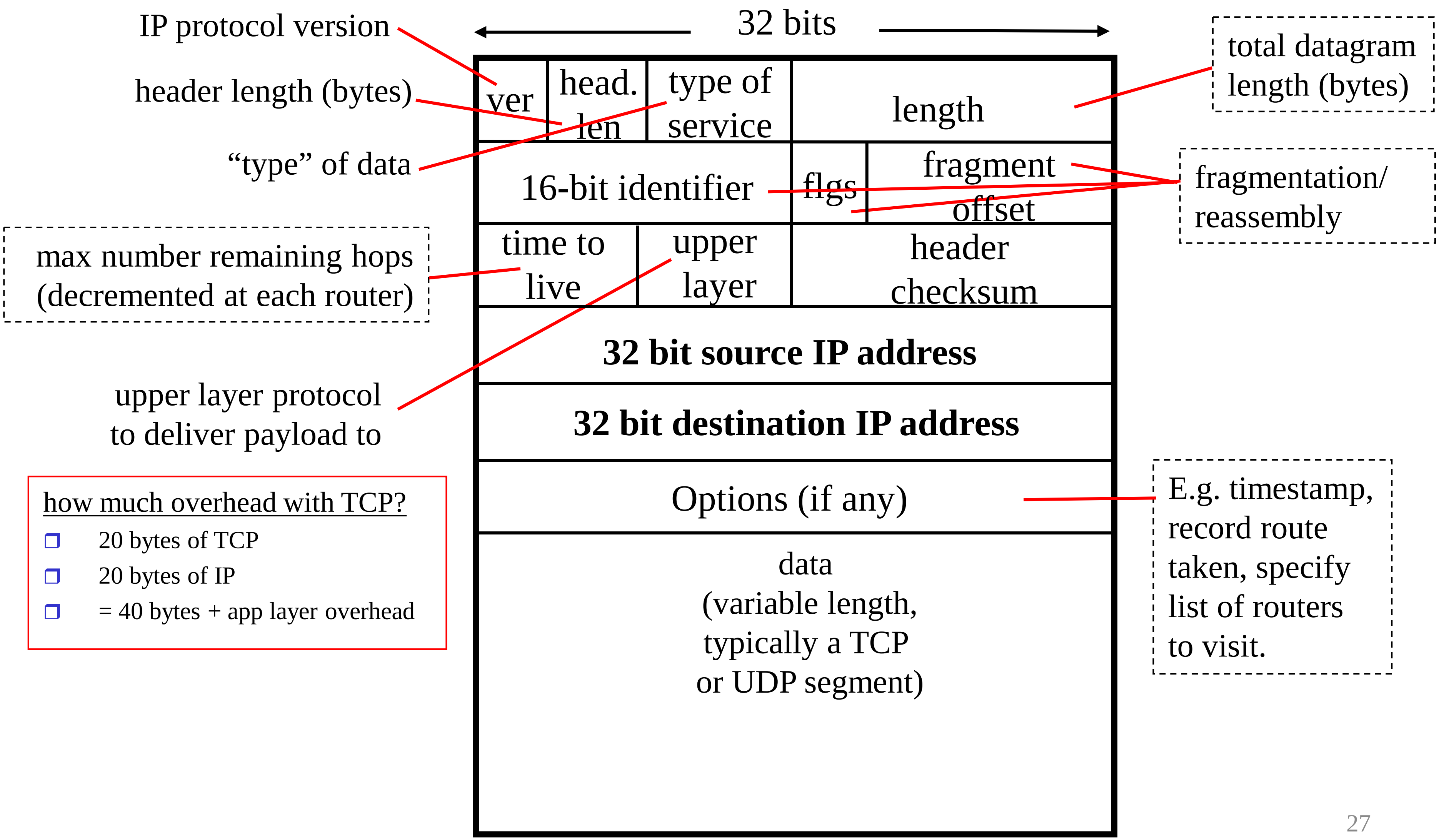
Internet Protocol

The Internet Network layer

Host, router network layer functions



IP Datagram Format



<u>how much overhead with TCP?</u>

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

Internet Checksum

- The Internet (not layer 2) uses a checksum
  - » easily implementable in software
  - » 1's complement sum of 16 bit words
  - » Performance: d=2

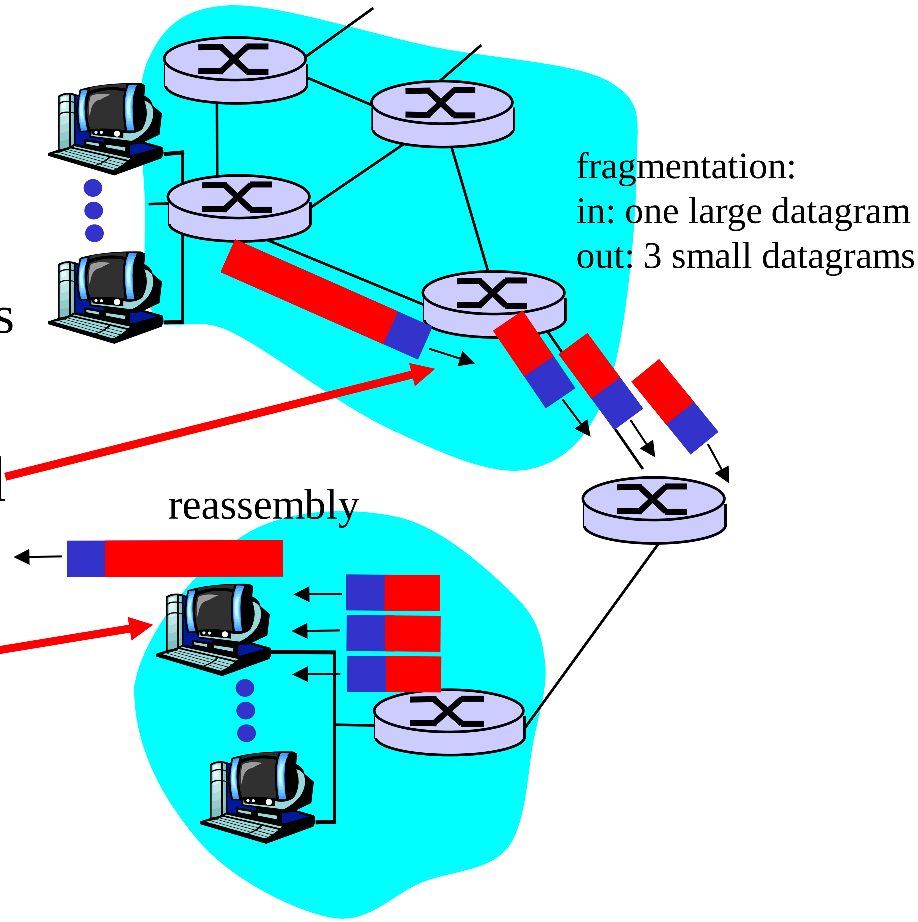
```
u_short
cksum(u_short *buf, int count)
{
    register u_long sum = 0;

    while (count--)
    {
        sum += *buf++;
        if (sum & 0xFFFF0000)
        {
            /* carry occurred,
               so wrap around */
            sum &= 0xFFFF;
            sum++;
        }
    }
    return ~(sum & 0xFFFF);
}
```

- One's complement sum
  - » Mod-2 addition with carry-out
  - » Carry-out in the most-significant-bit is added to the least-significant bit
  - » Get one's complement of "one's complement sum"

	1010011
	0110110
carry-out 1	**0001001**
Carry wrap-around	**0000001**
	0001010
One's complement =	**1110101**

IP Fragmentation and Reassembly

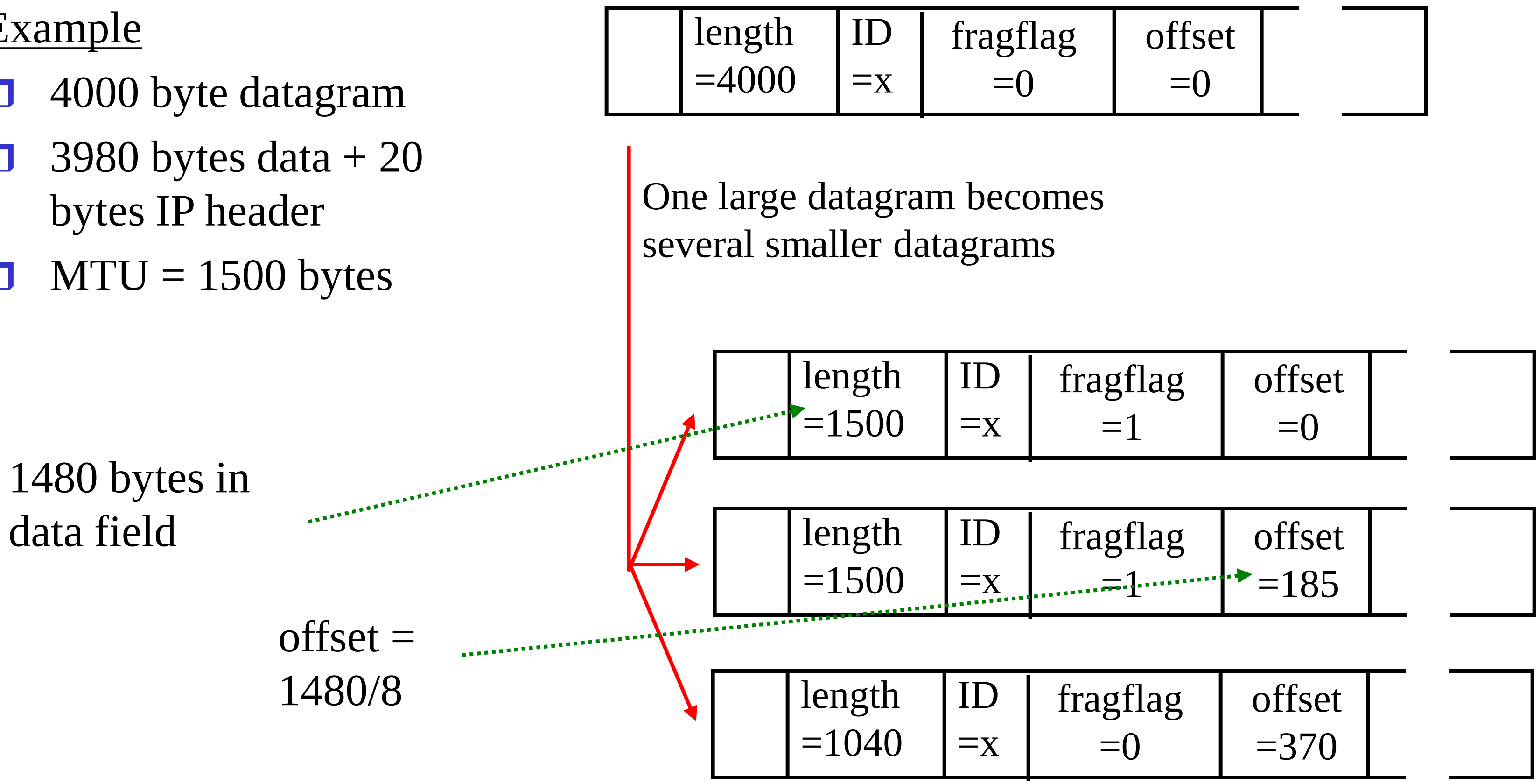
- Network links have MTU
  - » MTU - max. transfer size
  - » largest possible link-level frame
  - » different link types, different MTUs
- Large IP datagram is fragmented
  - » one datagram ➔ n datagrams
  - » “reassembled” at final destination
  - » IP header bits used to identify, order related fragments



IP Fragmentation and Reassembly

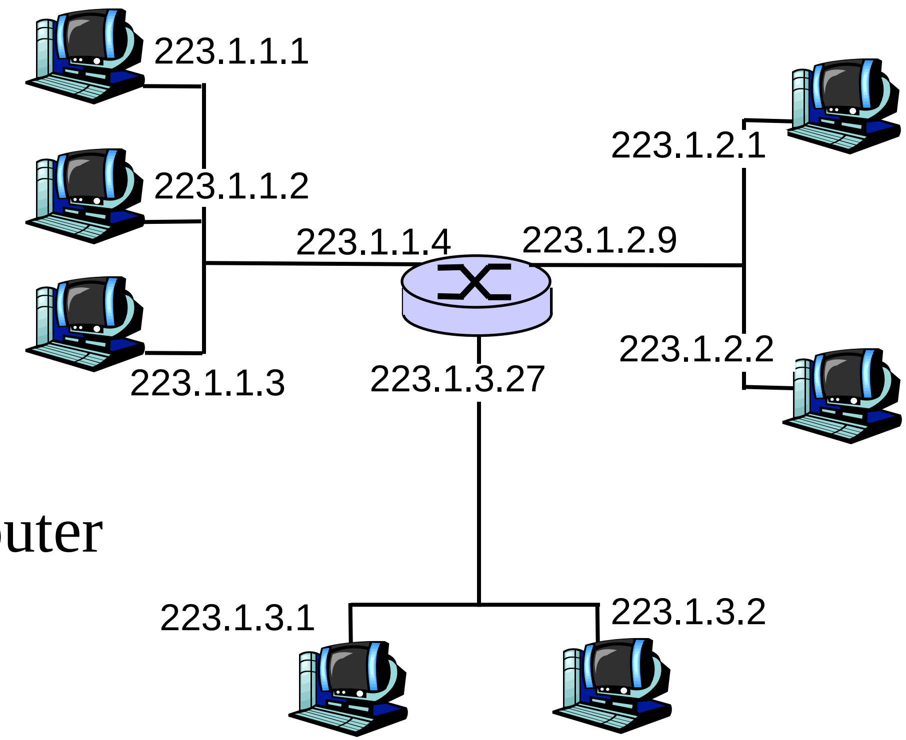
<u>Example</u>

- 4000 byte datagram
- 3980 bytes data + 20 bytes IP header
- MTU = 1500 bytes



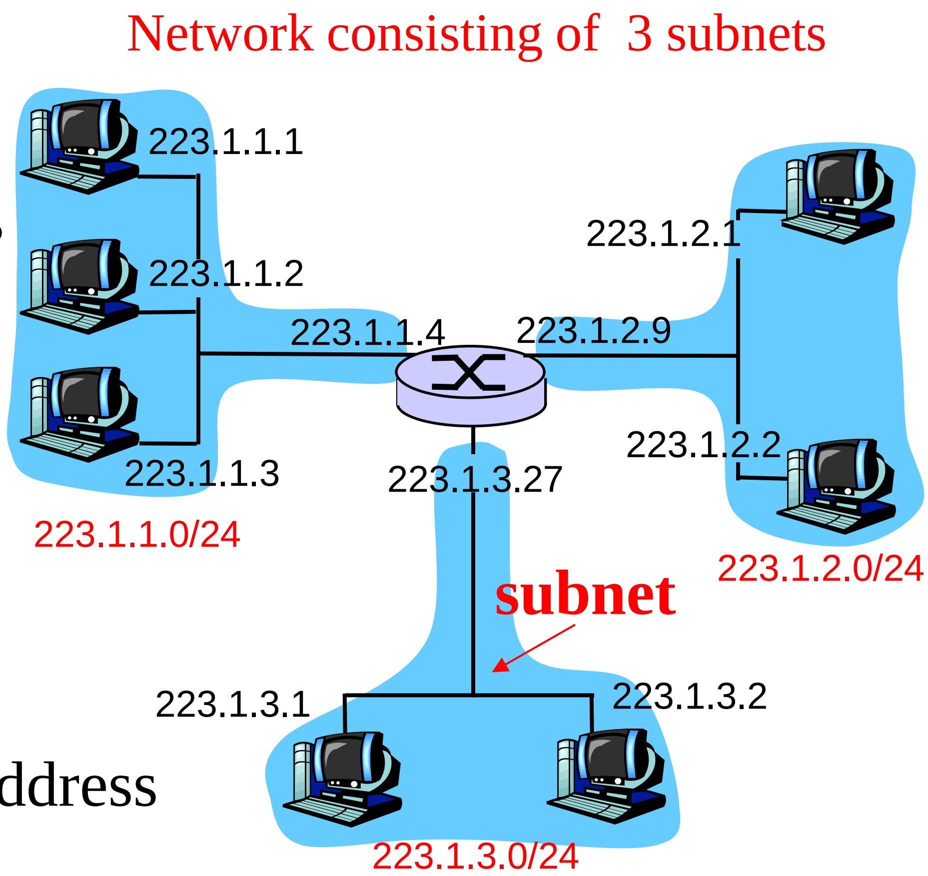
IP Addressing - Introduction

- IP address
  - » 32-bit identifier for host/router interface
- Interface
  - » connection between host/router and physical link
  - » Routers have multiple interfaces
  - » IP addresses associated with interface



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

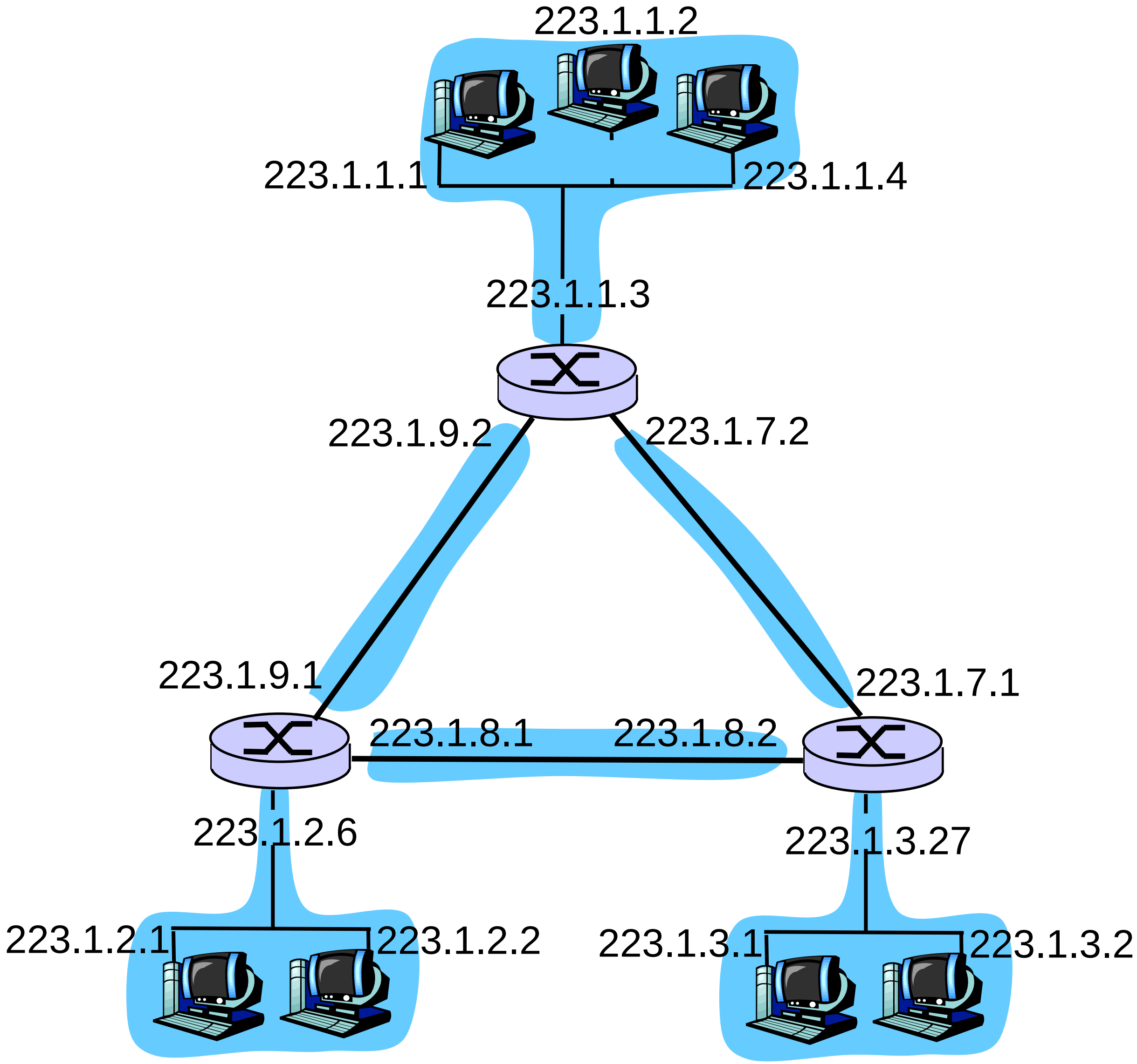
Subnets



- IP address
  - » subnet part ➔ high order bits
  - » host part ➔ low order bits
- Subnet ➔set of interfaces
  - » with same subnet part of IP address
  - » can reach each other

    without router intervention

6 Subnets



223.1.1.2
223.1.1.1
223.1.1.4
223.1.1.3
223.1.9.2
223.1.7.2
223.1.9.1
223.1.7.1
223.1.8.1
223.1.8.2
223.1.2.6
223.1.3.27
223.1.2.1
223.1.2.2
223.1.3.1
223.1.3.2

IP Addressing - CIDR

CIDR: Classless InterDomain Routing

» subnet portion of address has arbitrary length

» address format ➔ a.b.c.d/x

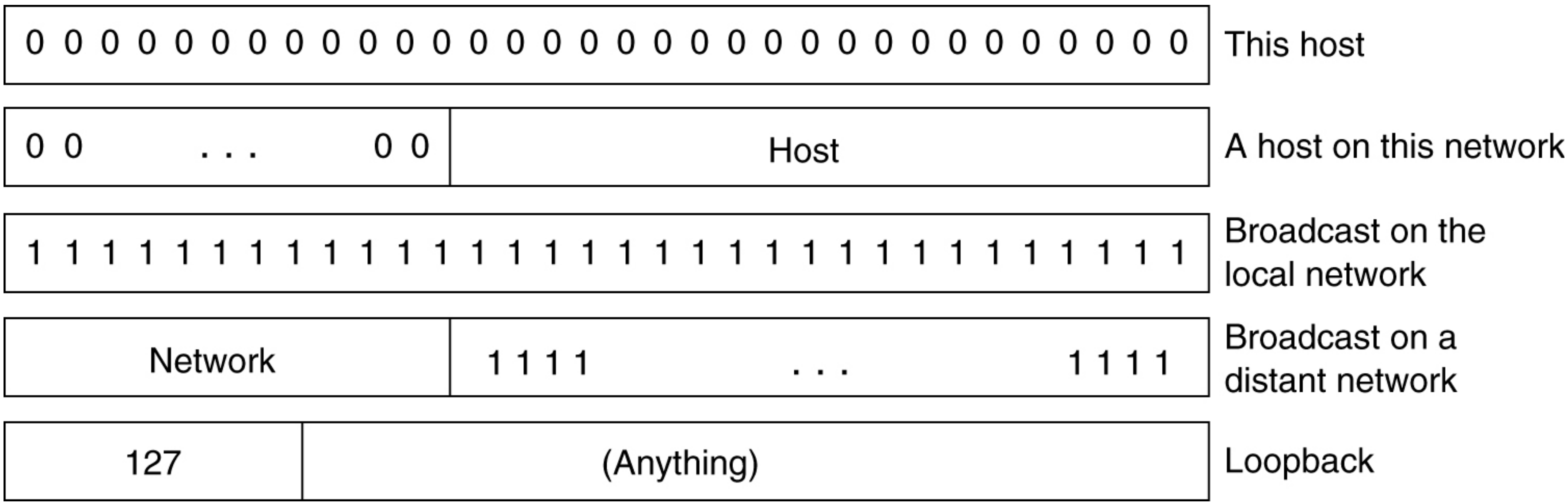
– where x is # bits in subnet portion of address

subnet part　　　　host part

11001000 00010111 00010000 00000000

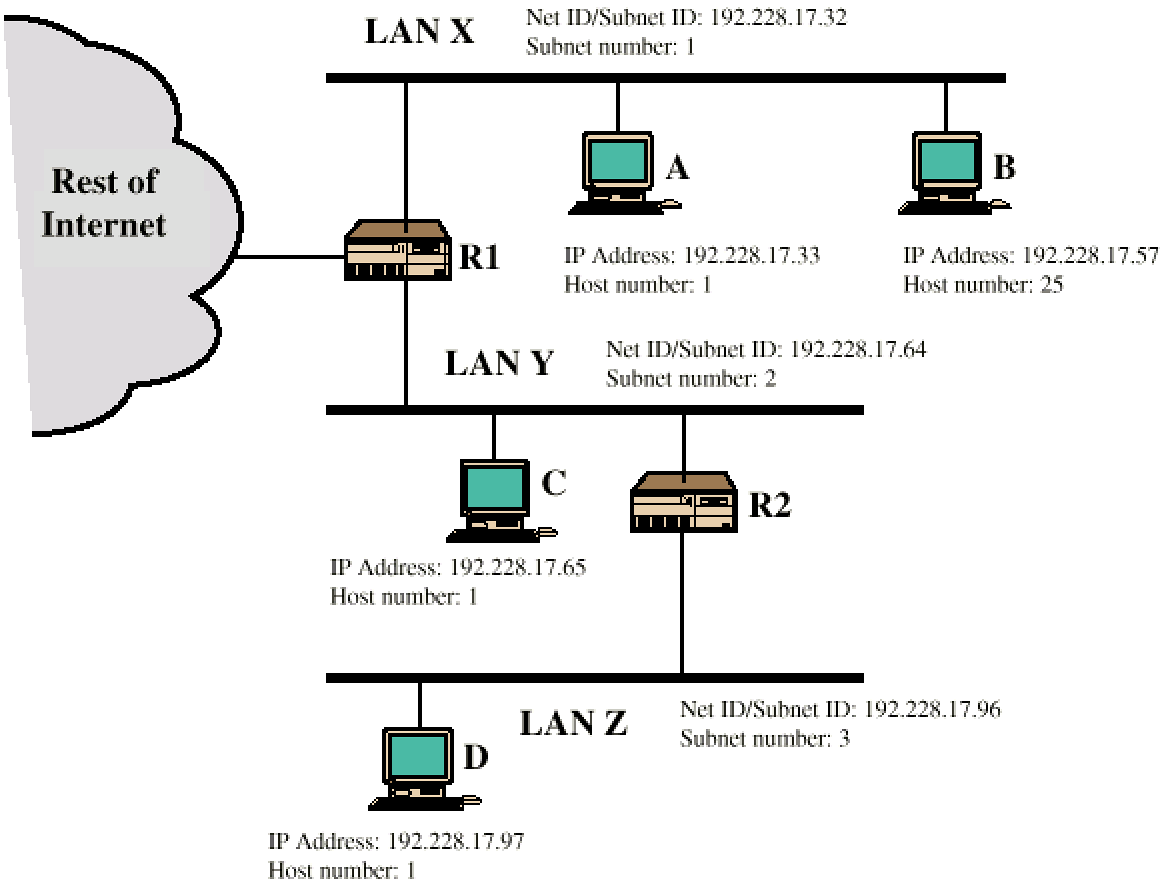
200.23.16.0/23

Special IP Addresses



Forming Sub-Networks

Network **192.228.17.0/24** is divided in 8 subnetworks ➔ masks of 27 bits



Subnetwork mask – 27 bits

subnetid – 3 bits (8 subnetworks)

11000000 11100100 00010001 01100000

192.228.17.96/27

hostid – 5 bits

30 hosts per subnet supported

all 0 – identifies subnet

all 1 – broadcast address

Example of subnetworks

192.228.17.0/27 **(.00000000)**

192.228.17.32/27 **(.00100000)**

192.228.17.64/27 **(.01000000)**

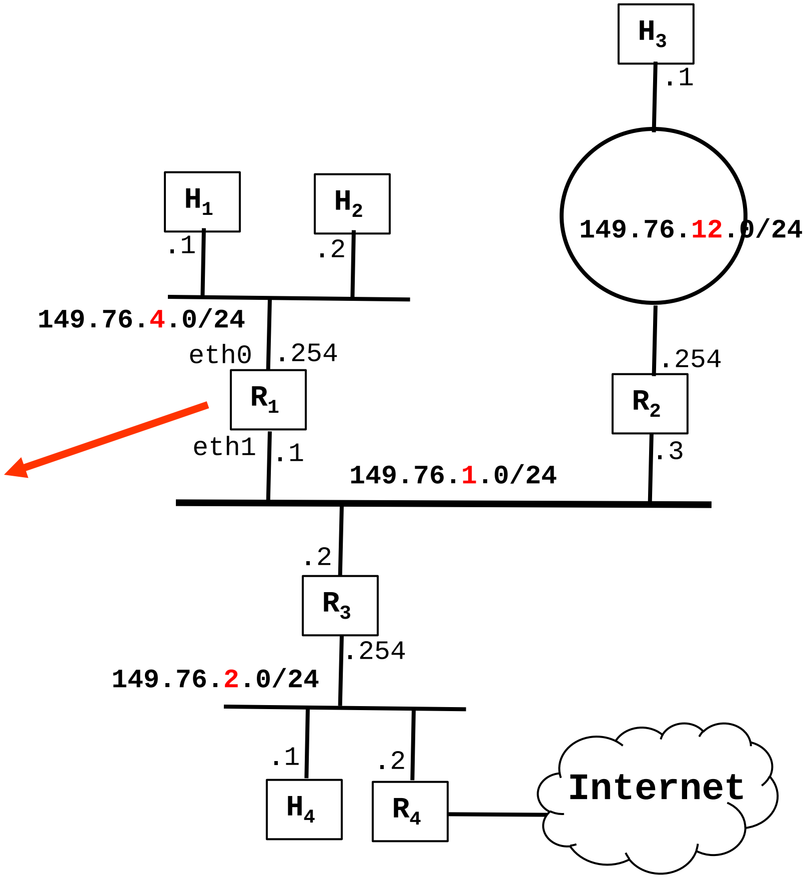
192.228.17.96/27 **(.01100000)**

…..

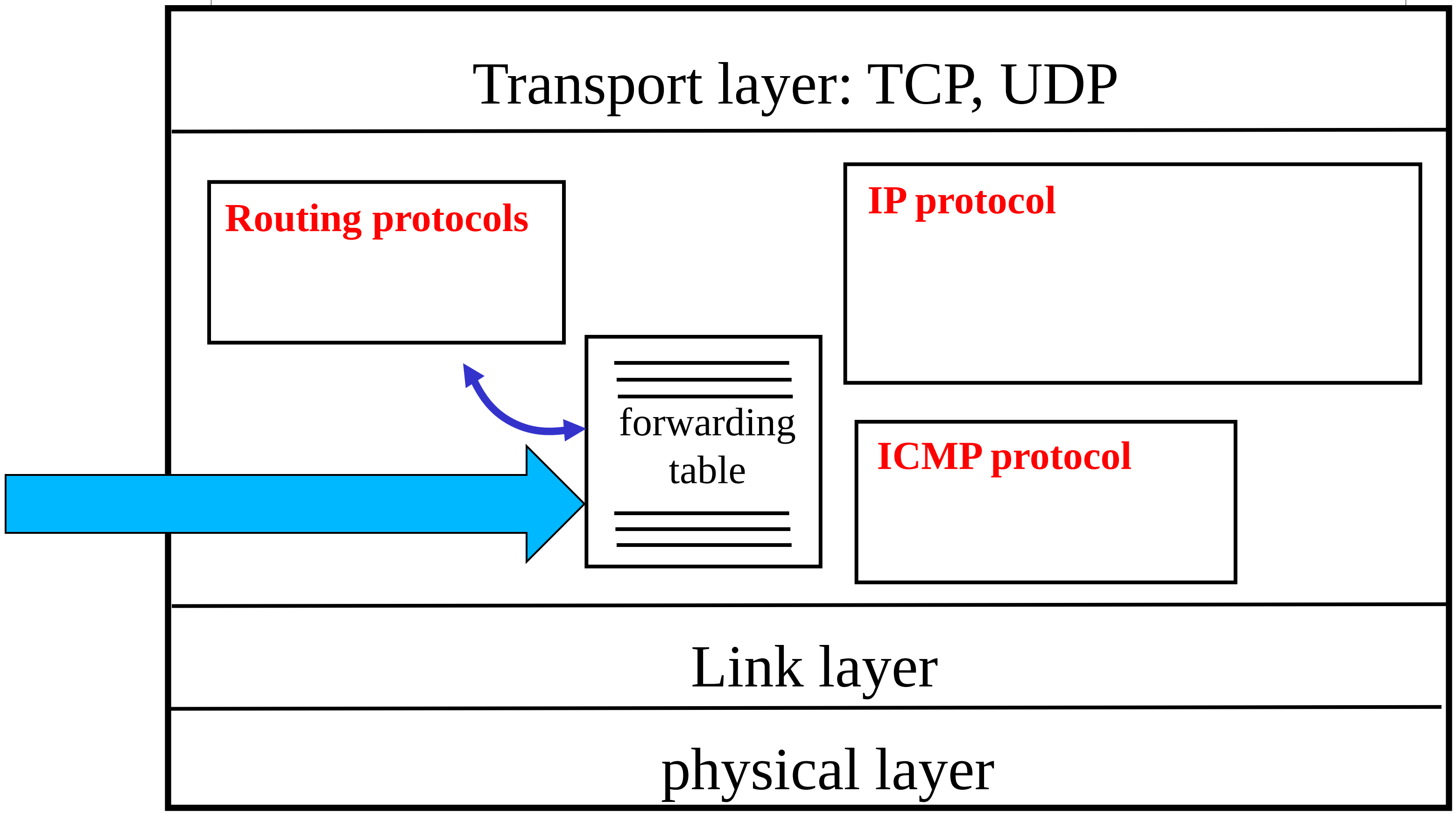
192.228.17.224/27 **(.11100000)**

Forwarding Table at R1

Destination	Gateway	Interface
149.76.1.0/24	-	eth1
149.76.2.0/24	149.76.1.2	eth1
149.76.4.0/24	-	eth0
149.76.12.0/24	149.76.1.3	eth1
0/0	149.76.1.2	eth1



Forwarding Table at R1



- What is a loopback interface? What is its IP address?

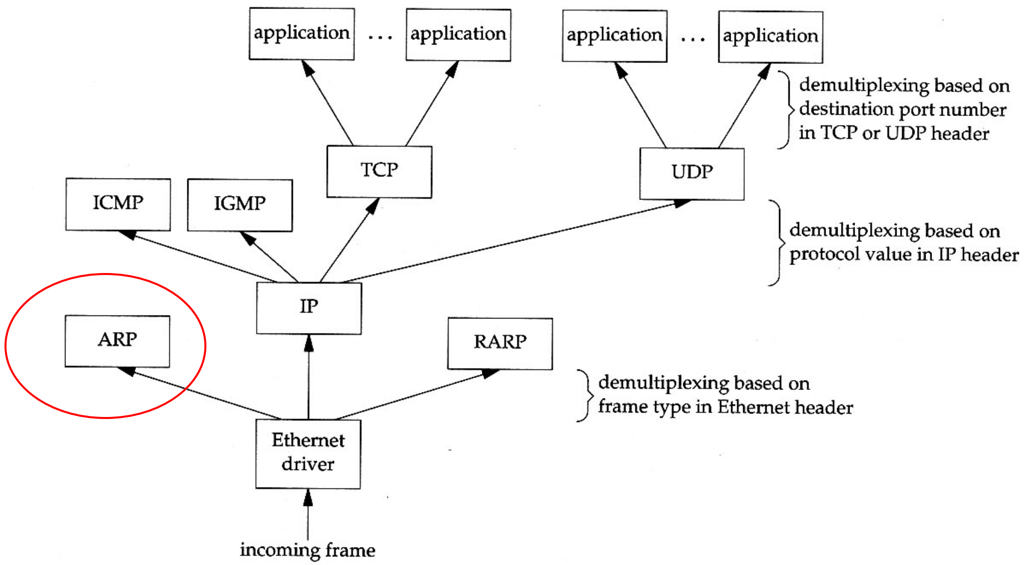
IP Forwarding Function

- Forwarding table has entries in format
  `<networkAddress/mask, port>`
- Forwarding function
  - » When a datagram arrives with destination address A, then
    - – For each entry of the forwarding table
      - `val= A & mask*` (e.g., mask=8, mask*=255.0.0.0)
      - if (`val == networkAddress` & mask*)
        - – add corresponding output port to the set of candidate ports
    - – Select the port with the largest mask → most specific route
  - » Example
    - – `frdTbl={<128.32.1.5/16,1>, <128.32.225.0/18,3>, <128.0.0.0/8,5>}`
    - – Datagram with destination address `A=128.32.195.1`
    - – Set of candidate output ports → `{1,3,5}.`
    - – Selected port → **3** ← largest mask, 18 bits

Address Resolution Protocol

Demultiplexing

- Ethernet header (type)
  - » IP - 0x0800
  - » ARP - 0x0806
  - » RARP - 0x8035
  - » IPX- 0x8037
  - » IPv6 - 0x86DD
  - » MPLS - 0x8847
- IP header (protocol)
  - » ICMP - 1
  - » IGMP - 2
  - » TCP - 6
  - » UDP – 17
- TCP/UDP header (port)
  - » FTP – 21
  - » Telnet - 23
  - » HTTP - 80
  - » SMTP - 25



ARP – Address Resolution Protocol

- A network interface has
  - one MAC address
  - one (or more) IP address(es)
- ARP: Address Resolution Protocol
  - Protocol used to

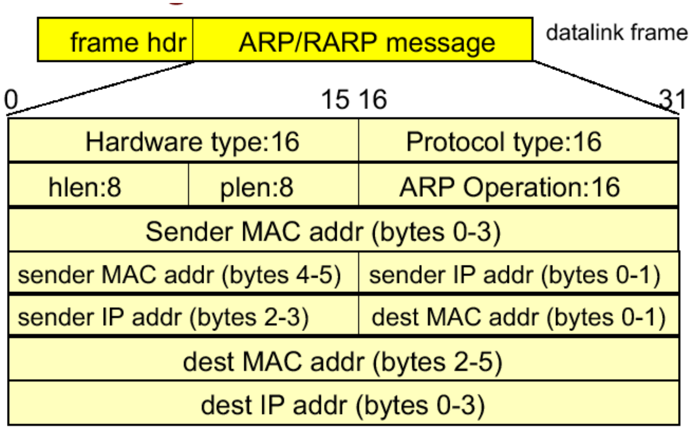
    obtain the MAC address

    associated to a given IP address
- RARP – Reverse ARP
  - Protocol used to

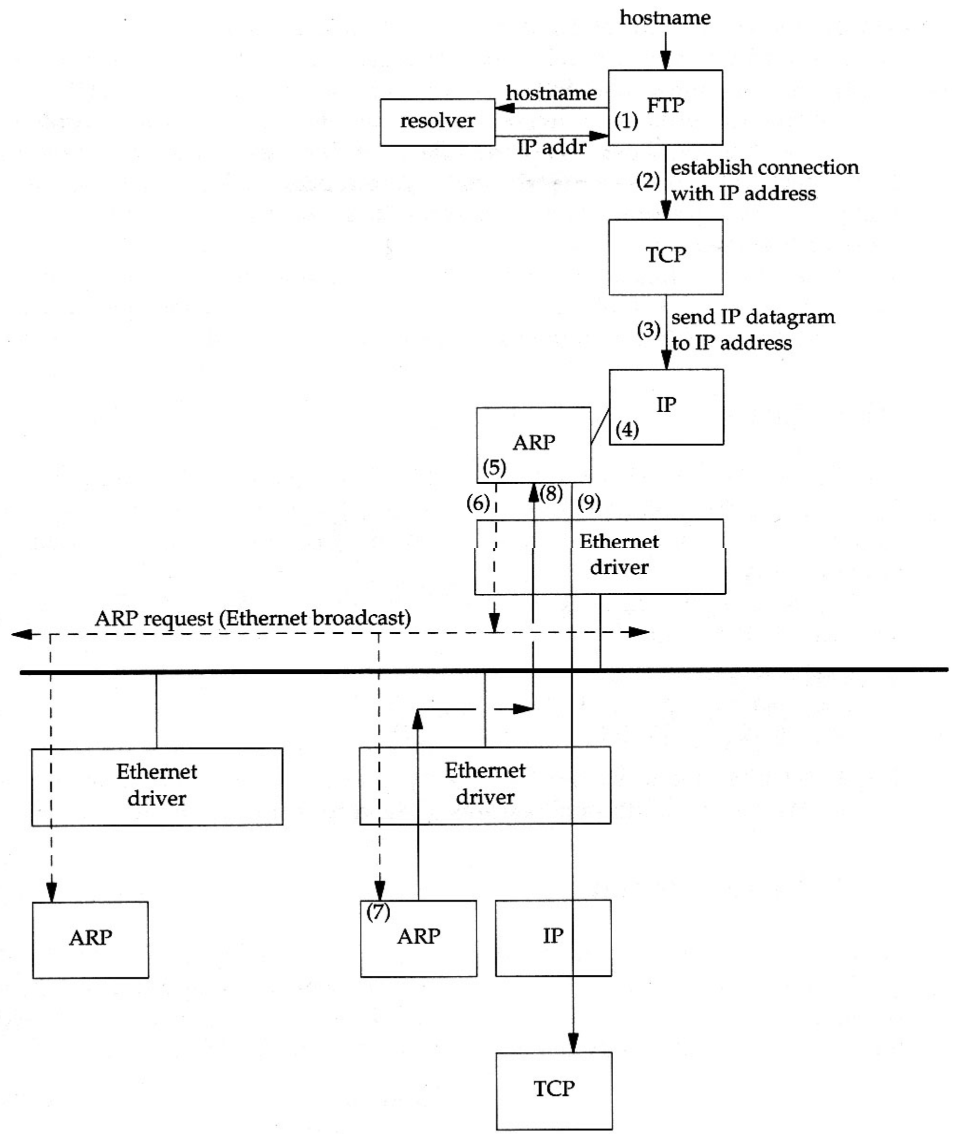
    obtain the IP address

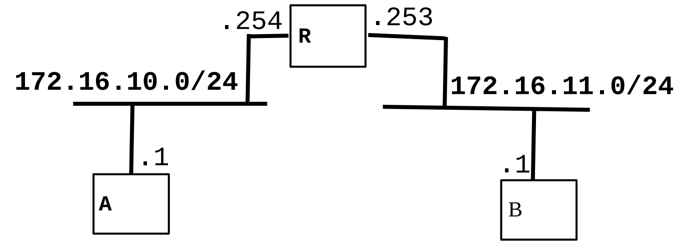
    associated to a MAC address

ARP Example



- **hardware type** : **Ethernet=1 ARCNET=7, localtalk=11**
- **protocol type** : **IP=0x800**
- **hlen** : **length of hardware address, Ethernet=6 bytes**
- **plen** : **length of protocol address, IP=4 bytes**
- **ARP operation** : **ARP request = 1, ARP reply = 2**
  RARP request = 3, RARP reply = 4





- Assume host A sends an IP packet to host B and that this packet is forwarded by router R. What are the MAC and IP addresses (source and destination) observed?

- What roles does ARP play in this scenario?

Obtaining IP Addresses

How to Obtain IP Addresses

How does network get subnet part of IP addresss?

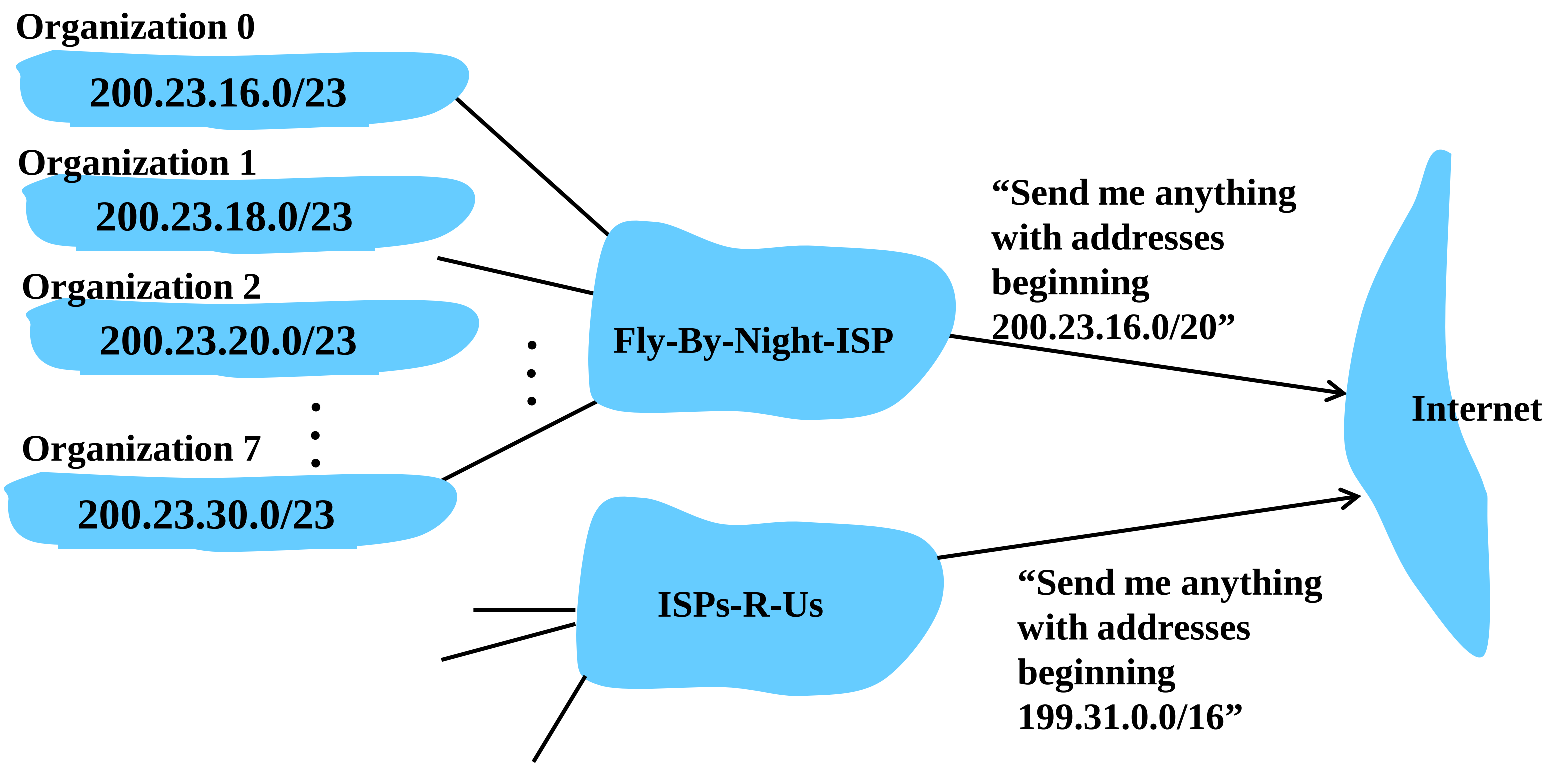
» Gets allocated portion of its provider ISP’s address space

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

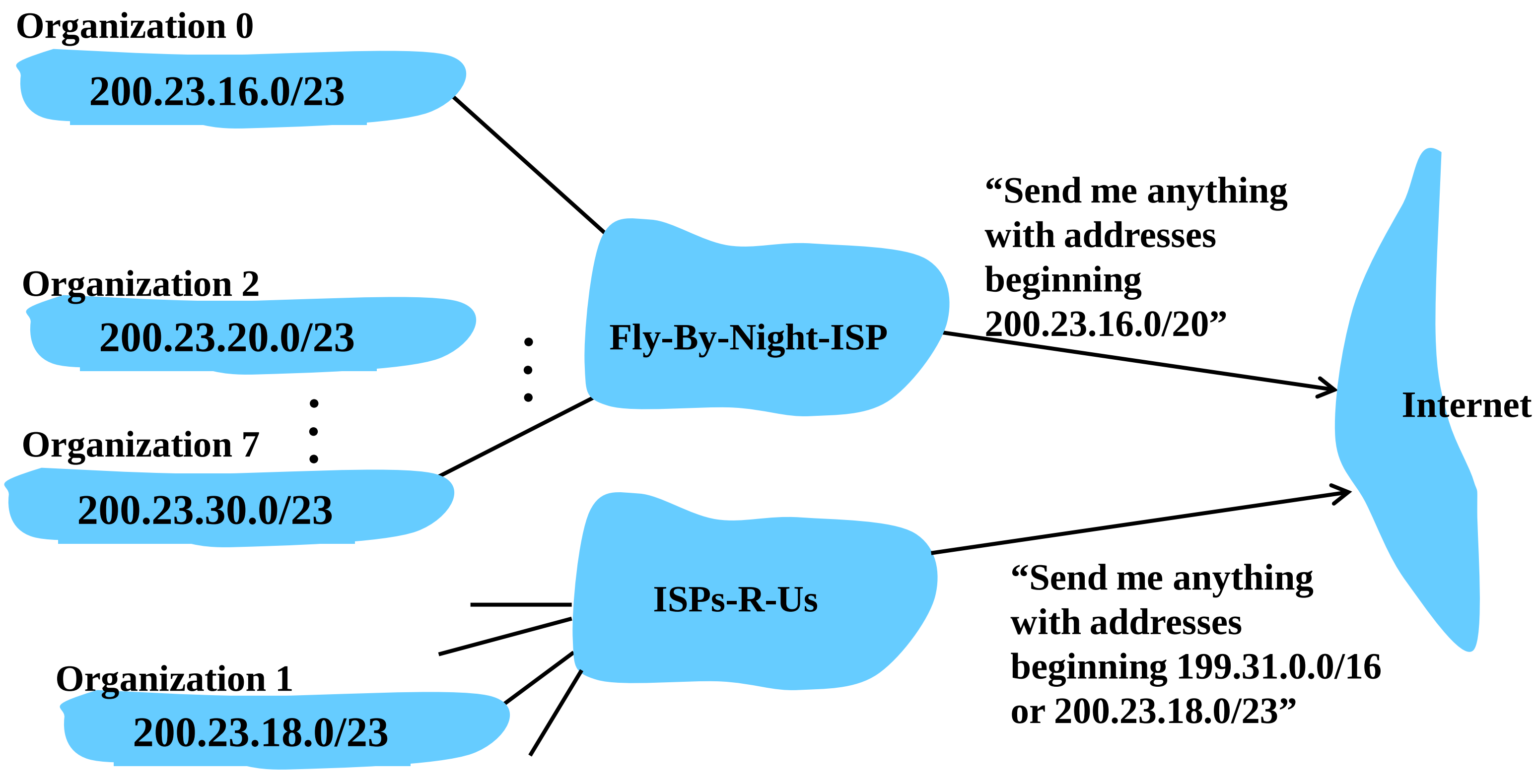
- Hierarchical addressing

  allows efficient advertisement of routing information



Hierarchical Addressing - More Specific Routes

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



IP Addressing

How does an ISP get block of addresses?

- » From ICANN: Internet Corporation for Assigned Names and Numbers
- » ICANN
  - – allocates addresses
  - – manages Domain Name Service (DNS)
  - – assigns domain names, resolves disputes

IP Addresses

How does a host obtain an IP address?

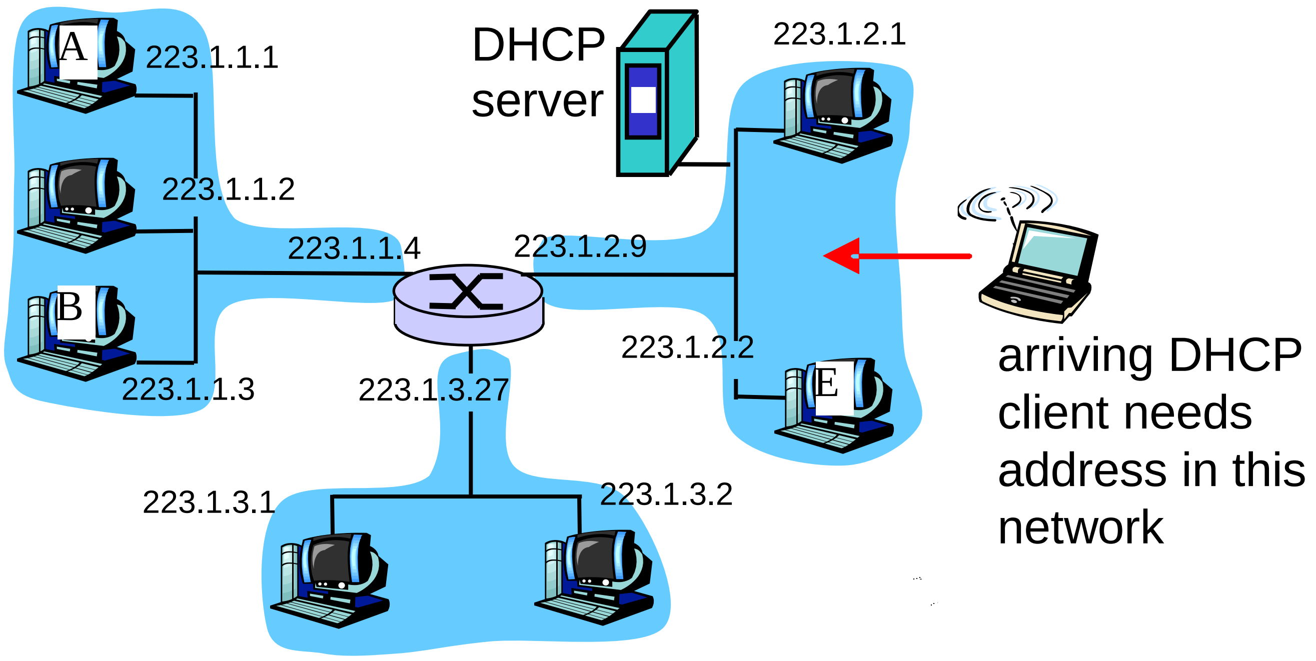
- » Hard-coded by system admin in a file
  - – Windows: control-panel → network → configuration → tcp/ip → properties
  - – UNIX: /etc/rc.config, ifconfig
- » DHCP: Dynamic Host Configuration Protocol
  - – dynamically get address from a server
  - – “plug-and-play”

DHCP - Dynamic Host Configuration Protocol

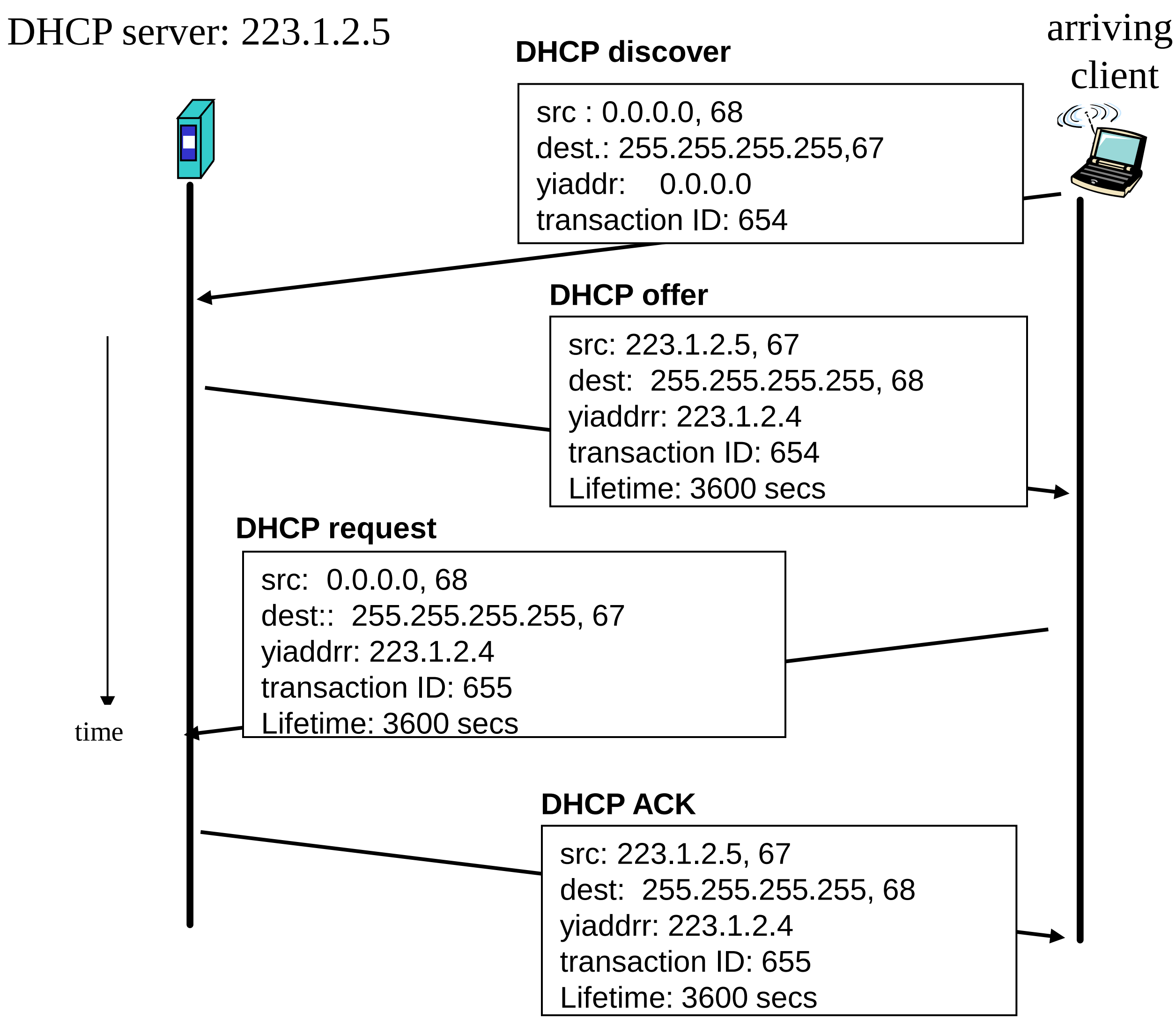
- DHCP allows
  - host to dynamically obtain its IP address from network server
    - when it joins network
  - reuse of addresses

- 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



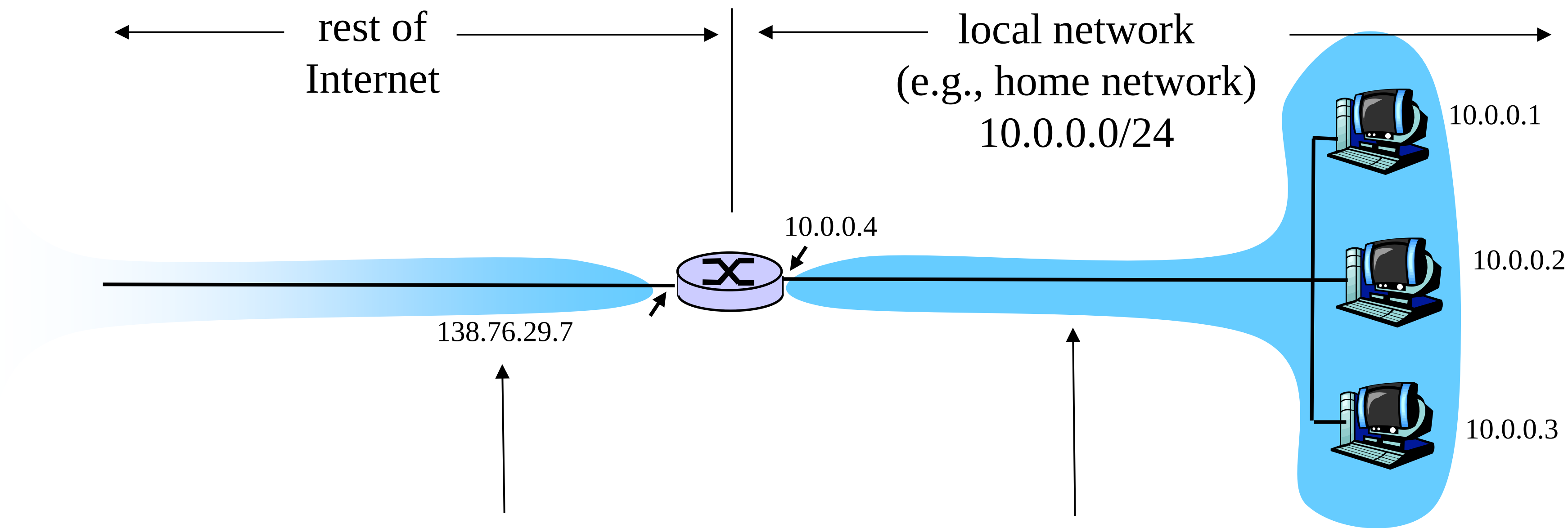
DHCP Client-server



- Is it sufficient for an arriving client to acquire an IP address? What other relevant information shall this client obtain in order to start working with full functionality?

Network Address Translation

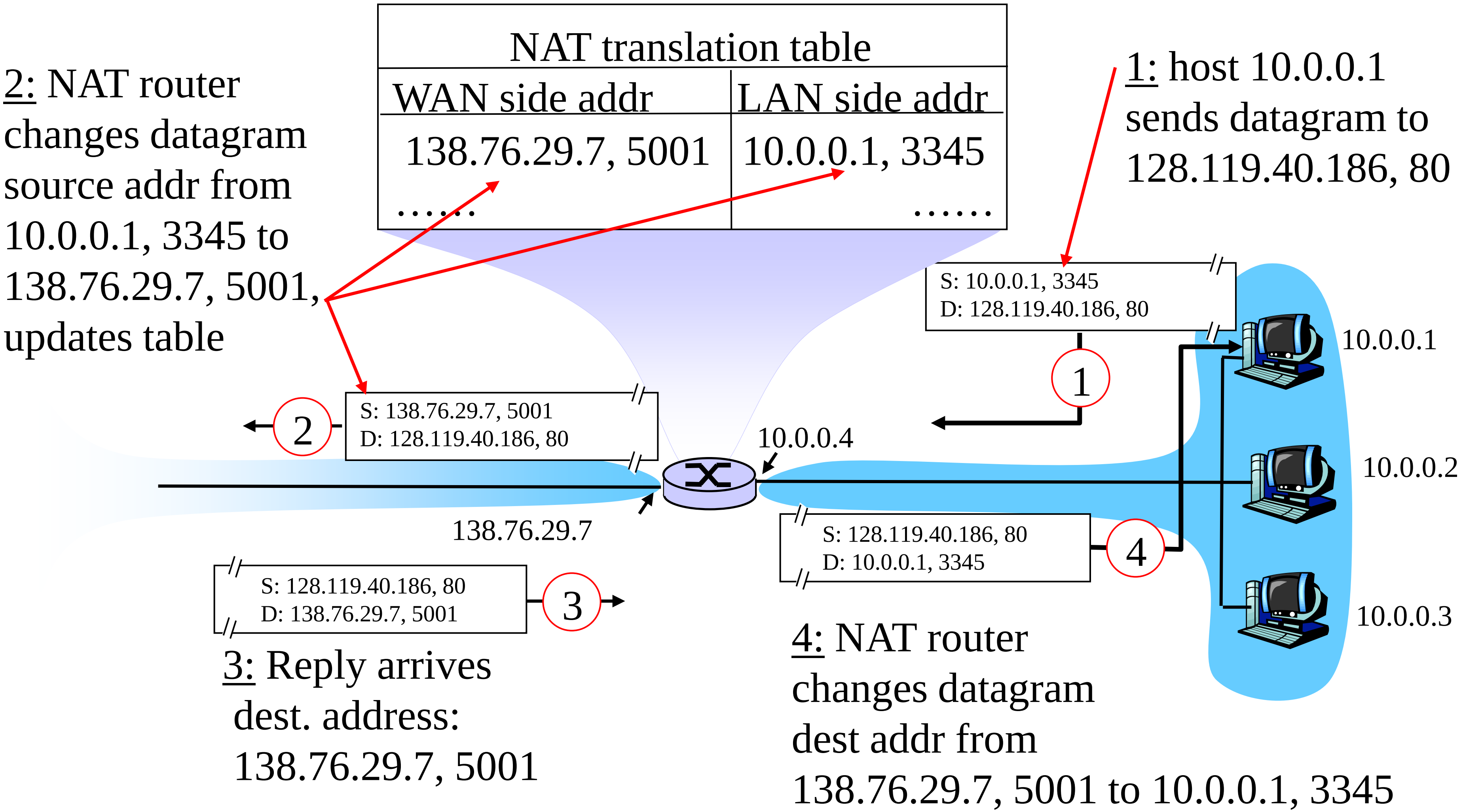
NAT - Network Address Translation



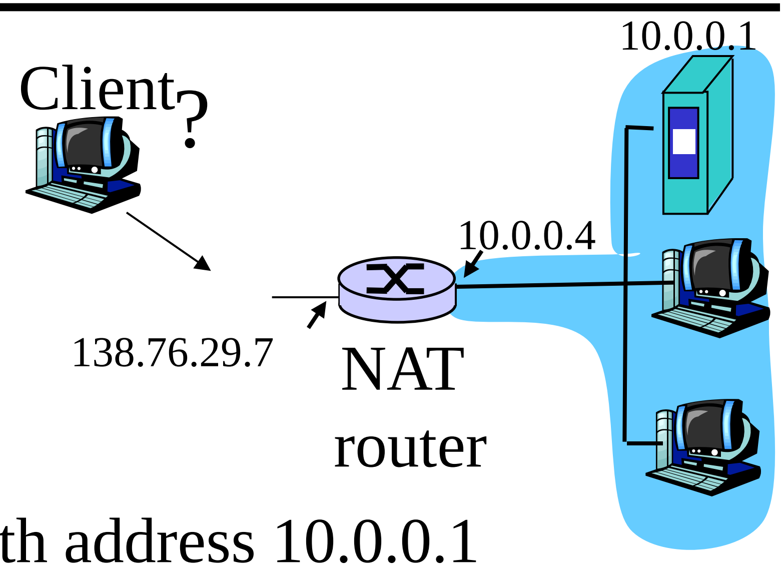
All datagrams *leaving* local network have same single source NAT IP address: 138.76.29.7, different source port numbers

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

NAT - Network Address Translation



NAT Traversal



- Client wants to connect to server with address 10.0.0.1
  - » but server address 10.0.0.1 is private
  - » only one externally visible NATed address: 138.76.29.7
- Possible solution – Port forwarding
  - » statically configure NAT
    to forward incoming connection requests at given port to server
  - » e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

- How to enable two hosts behind NAT boxes to communicate? How does Skype enable voice calls between these two clients?

Internet Message Control Protocol

ICMP - Internet Control Message Protocol

- Used by router or host
  - » to send layer 3 error or control messages
  - » to other hosts or routers
- Carried in IP datagrams

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Unused		
IP Header + 64 bits of original datagram		

(a) Destination Unreachable; Time Exceeded; Source Quench

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Pointer	Unused	
IP Header + 64 bits of original datagram		

(b) Parameter Problem

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Gateway Internet Address		
IP Header + 64 bits of original datagram		

(c) Redirect

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Identifier		Sequence Number
Optional data		

(d) Echo, Echo Reply

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Identifier		Sequence Number
Originate Timestamp		

(e) Timestamp

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Identifier		Sequence Number
Originate Timestamp		
Receive Timestamp		
Transmit Timestamp		

(f) Timestamp Reply

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Identifier		Sequence Number

(g) Address Mask Request

0 – 7	8 – 15	16 – 31
Type	Code	Checksum
Identifier		Sequence Number
Address Mask		

(h) Address Mask Reply

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)
5		Redirect
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Ping – Echo Request, Echo Reply

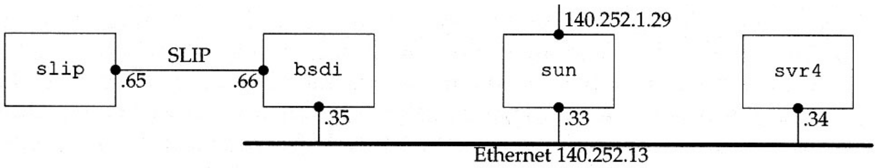
```
sun% ping gemini
PING gemini: 56 data bytes
64 bytes from gemini (140.252.1.11): icmp_seq=0. time=373. ms
64 bytes from gemini (140.252.1.11): icmp_seq=l. time=360. ms
64 bytes from gemini (140.252.1.11): icmp_seq=2. time=340. ms
64 bytes from gemini (140.252.1.11): icmp_seq=3. time-320. ms
64 bytes from gemini (140.252.1.11): icmp_seq=4. time=330. ms
64 bytes from gemini (140.252.1.11): icmp seq=5. time=310. ms
64 bytes from gemini (140.252.1.11): icmp seq=6. time==290. ms
64 bytes from gemini (140.252.1.11): icmp_seq-7. time=300. ms
64 bytes from gemini (140.252.1.11): icmp_seq=8. time=280. ms
64 bytes from gemini (140.252.1.11): icmp_seq=9. time=290. ms
64 bytes from gemini (140.252.1.11): icmp_seq=10. time=300. ms
64 bytes from gemini (140.252.1.11): icmp_seq=ll. time=280. ms
--gemini PING Statistics--
12 packets transmitted, 12 packets received, 0% packet loss
round-trip (ms) min/avg/max = 280/314/373
```

Traceroute and ICMP

- Source sends series of UDP segments to destination
  - first segment has TTL =1
  - second segment has TTL=2, …
  - unlikely port number
- When nth datagram arrives

  to nth router
  - router discards datagram
  - sends to source: ICMP TTL expired
  - message includes router name & IP address

```
svr4% traceroute slip
traceroute to slip (140.252.13.65), 30 hops max. 40 byte packets
1 bsdi (140.252.13.35) 20 ms 10 ms 10 ms
2 slip (140.252.13.65) 120 ms 120 ms 120 ms

slip% traceroute svr4
traceroute to svr4 (140.252.13.34), 30 hops max, 40 byte packets
1 bsdi (140.252.13.66) 110 ms 110 ms 110 ms
2 svr4 (140.252.13.34) 110 ms 120 ms 110 ms
```



- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times for each TTL
- Stop criterion
  - UDP segment eventually arrives at destination host
  - Destination returns ICMP “dest port unreachable” packet
  - source stops

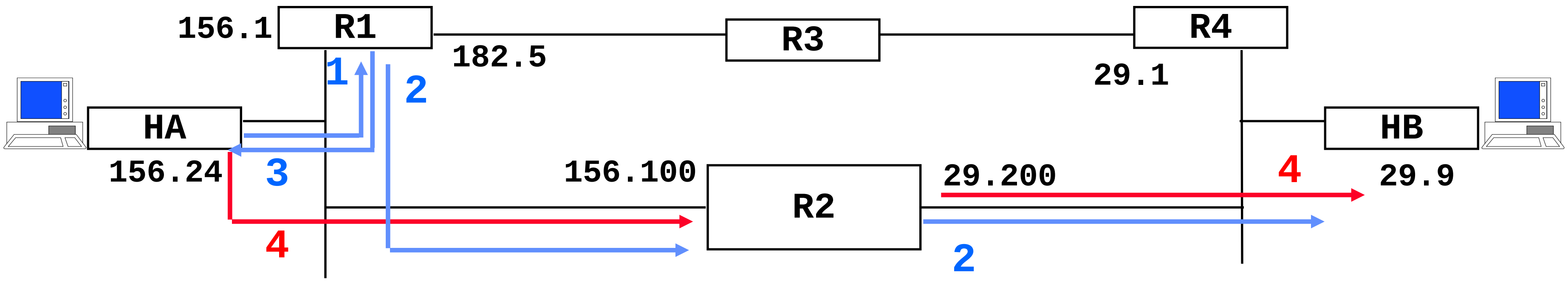
ICMP Redirect

- General routing principle of the TCP/IP architecture
  - » routers have extensive knowledge of routes
  - » hosts have minimal routing information ➔ learn routes also from ICMP redirects
- ICMP redirect message
  - » Sent by router R1 to source host A
    - – when R1 receives a packet from A with destination = B, and R1
      - ◆ finds that the next hop is R2 and
      - ◆ A is on-link with R2
  - » R1 sends ICMP redirect to A saying next hop for destination B is R2
  - » A updates its forwarding table with a host route

ICMP Redirect Format

```
/                                                                 /
|                  IP datagram header  (prot = ICMP)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type=5       |     code       |           checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Router IP address that should be preferred         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        IP header plus 8 bytes of original datagram data         |
/                                                                 /
```

ICMP Redirect Example



	dest IP addr	srce IP addr	prot	data part
1:	193.154.29.9	193.154.156.24	udp	xxxxxxx
2:	193.154.29.9	193.154.156.24	udp	xxxxxxx
3:	193.154.156.24	193.154.156.1	icmp	type=redir code=host cksum 193.154.156.100 xxxxxxx (28 bytes of 1)
4:	193.154.29.9	193.154.156.24	udp	....

ICMP Redirect Example

After 4

```
HA$ netstat -nr
Routing Table:
  Destination                Gateway               Flags  Interface
-------------------- -------------------- ----- ---------
127.0.0.1            127.0.0.1             UH      lo0
193.154.29.9         193.154.156.100       UGH     eth0
193.154.156.0        193.154.156.24        U       eth0
224.0.0.0            193.154.156.24        U       eth0
default              193.154.156.1         UG      eth0
```

Flags:
- **U** - *route Up*
- **G** - *route to a Gateway (next hop router)*
- **H** - *route to a Host*

IPv6

The Need of a New IP

- IPv4
  - Small addressing space (32 bits)
  - Non-continuous usage
  - Some solutions used to overcome these problems
    private networks (NAT), classless networks (CDIR)

- IETF developed new IP version: **IPv6**
  - Same principles of IPv4
  - Many improvements
  - Header re-defined

IPv6 – Improvements

- 128 bit addresses (16 octets, 8 *shorts*). No classes
- Better QoS support (native flow label)
- Native security functions (peer authentication, data encryption)
- Autoconfiguration (*Plug-n-play)*
- Routing
- Multicast

Address Representation

- 8 x 16 bit, hexadecimal. Separated by **:**

  47CD : 1234 : 3200 : 0000 : 0000 : 4325 : B792 : 0428

- Compressed format: **FF01:0:0:0:0:0:0:43 → FF01::43**

- Compatibility with IPv4: **0:0:0:0:0:0:13.1.68.3 or ::13.1.68.3**

- Loopback address: **::1**

- Network prefix described by **/** , same as IPv4
  - **FEDC:BA98:7600::/40 → network prefix = 40 bits**

Reserved Addresses

Allocation	Prefix (binary)	Fraction of Address Space
Unassigned	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128
Unassigned	0000 01	1/64
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Global Unicast	001	1/8
Unassigned	010	1/8
Unassigned	011	1/8
Unassigned	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512
Link-Local Unicast Addresses	1111 1110 10	1/1024
Site-Local Unicast Addresses	1111 1110 11	1/1024
Multicast Addresses	1111 1111	1/256

Adresses – Link-Local, Global Unicast, Anycast, Multicast

» *Link-Local*

- Used for communication between hosts in the same LAN/link
- Address built from MAC address
- Routers do not foward packets having *Link-Local* destination addresses

» *Global Unicast*

- Global addresses
- Address: network prefix + computer identifier
- Structured prefixes

  Network aggregation; less entries in the router forwarding tables

» *Anycast*

- Group address; packet is received by any (only one) member of the group

» *Multicast*

- Group address; packet received by all the members of the group

Address Formats

```
|         n bits         |   m bits  |       128-n-m bits         |
+------------------------+-----------+----------------------------+
|001 global rout prefix  | subnet ID |       interface ID         |
+------------------------+-----------+----------------------------+
```

Global Unicast Address (2000::/3)

```
|   10     |
|  bits    |         54 bits         |         64 bits            |
+----------+-------------------------+----------------------------+
|1111111010|           0             |       interface ID         |
+----------+-------------------------+----------------------------+
```

Link-Local Unicast address (fe80::/10)

```
|   10     |
|  bits    |         54 bits         |         64 bits            |
+----------+-------------------------+----------------------------+
|1111111011|        subnet ID        |       interface ID         |
+----------+-------------------------+----------------------------+
```

Site-Local Unicast address (fec0::/10) (not used)

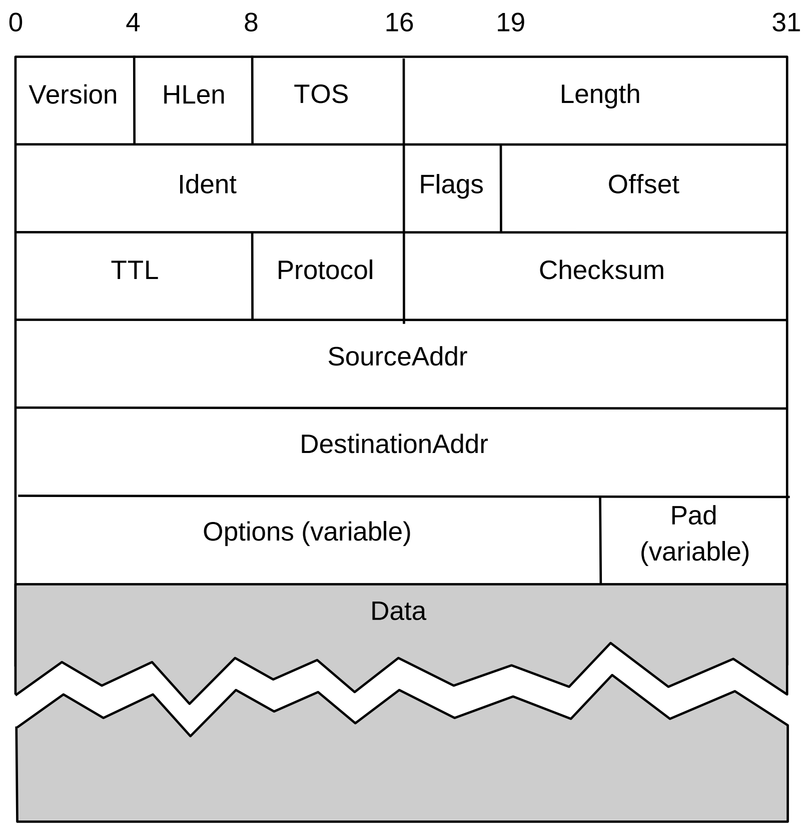
```
|                          n bits                 |   128-n bits   |
+-------------------------------------------------+----------------+
|                    subnet prefix                | 00000000000000 |
+-------------------------------------------------+----------------+
```

Anycast address

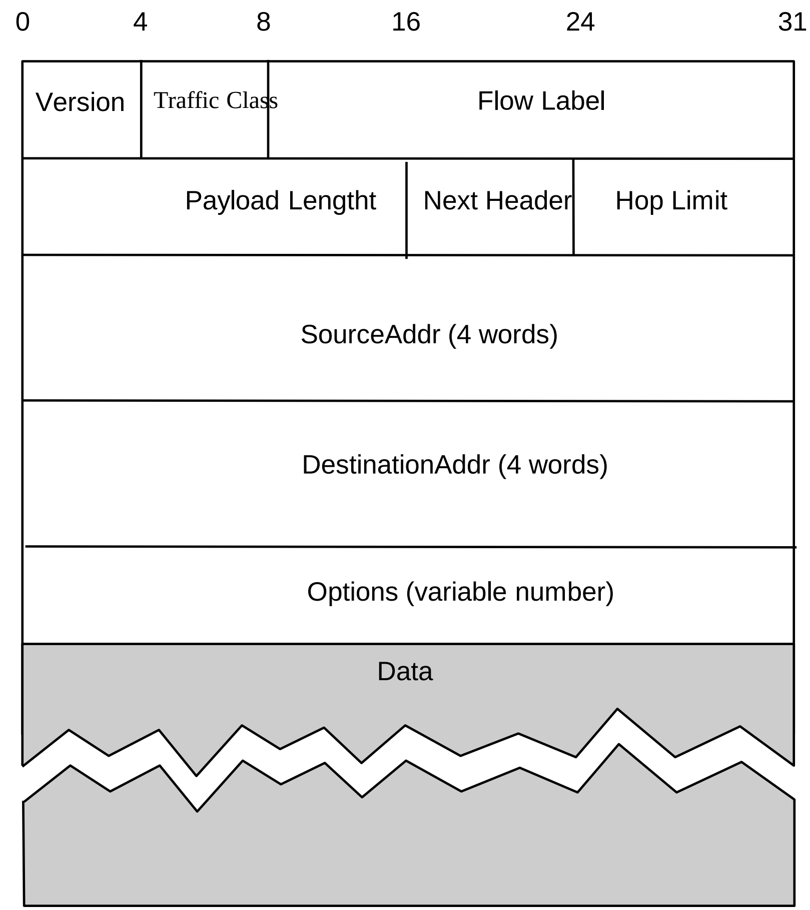
```
|   8    |  4 |  4 |                  112 bits                     |
+------ -+----+----+-----------------------------------------------+
|11111111|flgs|scop|                  group ID                     |
+--------+----+----+-----------------------------------------------+
```

Multicast address
Scope – link, site, global, ...
(ff::/8)

Headers IPv4 and IPv6



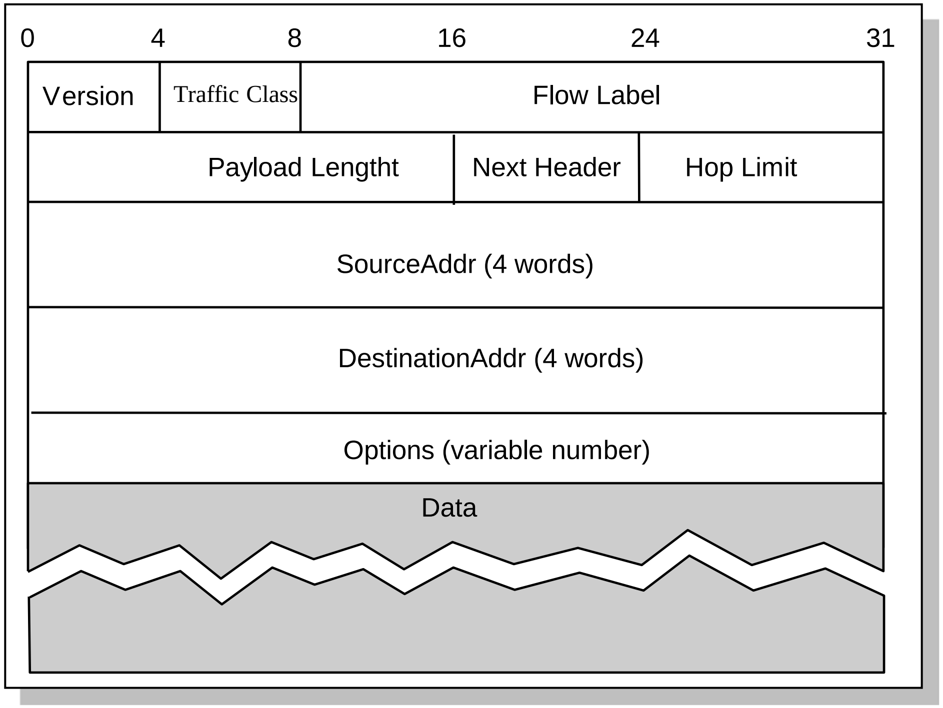
IPv4



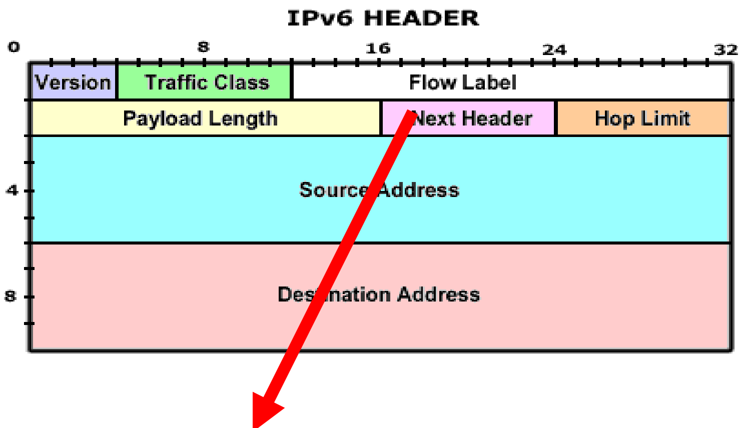
IPv6

IPv6 Header

- Flow label → identifies packet flow
  - » QoS, resource reservation
  - » Packets receive same service
- Payload length
  - » Header not included
- Hop limit = TTL (v4)
- Next header
  - » Identifies next header/extension
- Options → included as extension headers



Extension Headers



IPv6 Header Next Header = TCP	TCP header + data

IPv6 Header Next Header = Routing	Routing Header Next Header = TCP	TCP header + data

IPv6 Header Next Header = Routing	Routing Header Next Header = Fragment	Fragment Header Next Header = TCP	Fragment of TCP header + data

IPv6	Hop-by-hop	Destination	Routing	Fragment	Authenticate.	ESP	TCP

Extension Headers

- Hop-by-hop

  additional information, inspected by every node traversed by the packet

  Other headers are inspected only at the destination or at pre-defined nodes

- Destination: Information for the destination node
- Routing: List of nodes to be visited by the packet
- Fragmentation: Made by the source; it shall find MPU
- Authentication: Authentication (signature) of packet header
- ESP: Data encryption

Example of Lab Network

```
quadro
                                                  porta

banc_3                                            banc_6
pc3---[HUB]---pc2----+            +----pc2---[HUB]---pc3
2000:0:0:3::/64      |            |      2000:0:0:6::/64
                     |            |
banc_2               |            |               banc_5
pc3---[HUB]---pc2--[HUB]-+    +-[HUB]--pc2---[HUB]---pc3
2000:0:0:2::/64      |   |    |   |      2000:0:0:5::/64
                     |   |    |   |
banc_1               |   |    |   |               banc_4
pc3---[HUB]---pc2----+   |    |   +----pc2---[HUB]---pc3
2000:0:0:1::/64          |    |          2000:0:0:4::/64
                         |    |
          2000:0:0:e::/64|    |2000:0:0:d::/64
                         |    |
                       [routerv6]
```

2000:0:0:1::1

2000:0:0:1::aa

2000:0:0:e::1

Configuration examples in Linux

```
tux13:~# /sbin/ifconfig eth0 inet6 add 2000:0:0:1::1/64
tux13:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:C0:DF:08:D5:99
          inet addr:172.16.1.13  Bcast:172.16.1.255  Mask:255.255.255.0
          inet6 addr: 2000:0:0:1::1/64 Scope:Global
          inet6 addr: fe80::2c0:dfff:fe08:d599/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:81403 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2429 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:4981344 (4.7 MiB)  TX bytes:260692 (254.5 KiB)
          Interrupt:5
tux13:~# /sbin/route -A inet6 add 2000::/3  gw 2000:0:0:1::aa
tux13:~# route -A inet6
Kernel IPv6 routing table
Destination                    NextHop        Flags Metric Ref Use Iface
::1/128                        ::             U     0      0   0    lo
2000:0:0:1::1/128              ::             U     0      0   0    lo
2000:0:0:1::/64                ::             UA    256    0   0    eth0
2000::/3                       2000:0:0:1::aa UG    1      0   0    eth0
fe80::2c0:dfff:fe08:d599/128 ::               U     0      0   0    lo
fe80::/10                      ::             UA    256    0   0    eth0
ff00::/8                       ::             UA    256    0   0    eth0
::/0                           ::             UDA   256    0   0    eth0
```

Identifier IEEE EUI-64

**Method to create a IEEE EUI-64 identifier from an IEEE 48bit MAC identifier.
This is to insert two octets, with hexadecimal values of 0xFF and 0xFE,
in the middle of the 48 bit MAC (between the company_id and vendor supplied id).
For example, the 48 bit IEEE MAC with global scope:**

```
|0              1|1              3|3              4|
|0              5|6              1|2              7|
+----------------+----------------+----------------+
|cccccc0gcccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+  00:C0:DF:08:D5:99
```

**where "c" are the bits of the assigned company_id, "0" is the value of the
universal/local bit to indicate global scope, "g" is individual/group bit,
and "m" are the bits of the manufacturer-selected extension identifier.
The interface identifier would be of the form:**

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc1gcccccccc|cccccccc11111111|11111110mmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+----------------+
```

fe80::2c0:dfff:fe08:d599

Protocol Neighbor Discovery (ND)

- IPv6 node uses ND for
  - » Find other nodes in the same link /LAN
  - » Find a node MAC address
    - ND substitutes ARP
  - » Find router(s) in its network
  - » Mantaining information about neighbour nodes

- ND similar to the IPv4 functions
  - » ARP IPv4
  - » ICMP Router Discovery
  - » ICMP Redirect

ND Messages

- ICMP messages (over IP); using *Link Local* addresses

- ***Neighbor Solicitation***

  Sent by a host to obtain MAC address of a neighbour / to verify its presence

- ***Neighbor Advertisement:*** Answer to the request

- ***Router Advertisement***

  Information about the network prefix; periodic or under request

  Sent by router to IP address *Link Local multicast*

- ***Router Solicitation:*** host solicits from router a *Router Advertisment* message

- ***Redirect***: Used by a router to inform a host about the best route to a destination

Homework

1. Review slides

2. Read from Kurose&Ross
   » Chapter 4 – The Network Layer
   (this set of slides follows mainly Kurose&Ross)

3. Or, from Tanenbaum,
   » Chapter 5 – The Network Layer

4. Answer questions at moodle