

Network Layer



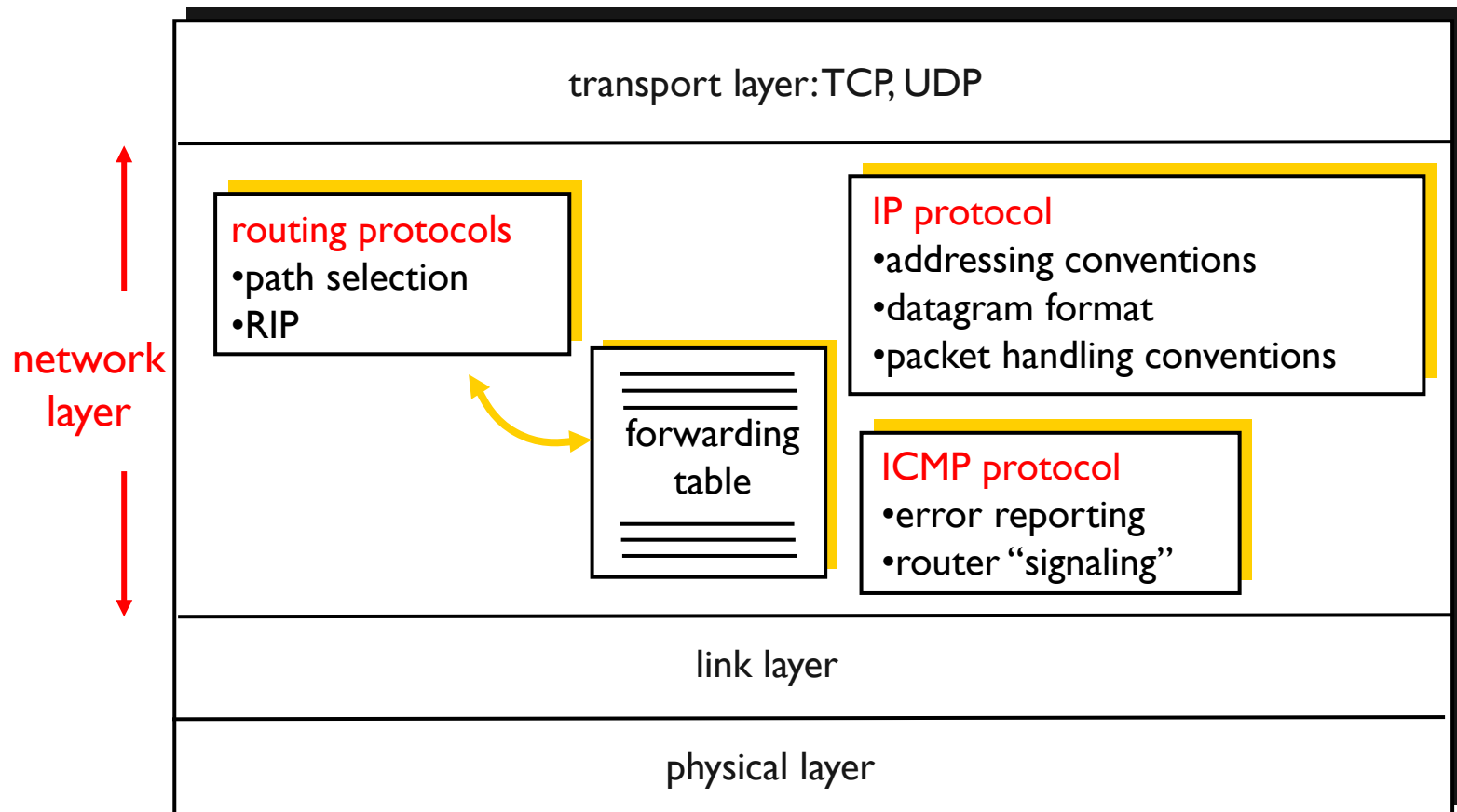
Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 14

puc@marshall.edu

The Internet Network Layer: Host, Router Network Layer Functions

focus on how **addressing** and **forwarding** are done in the Internet!



IP Datagram Format

IP protocol version
number

header length
(bytes)

“type” of data

max number
remaining hops
(decremented at
each router)

upper layer protocol
to deliver payload to

how much overhead with
TCP?

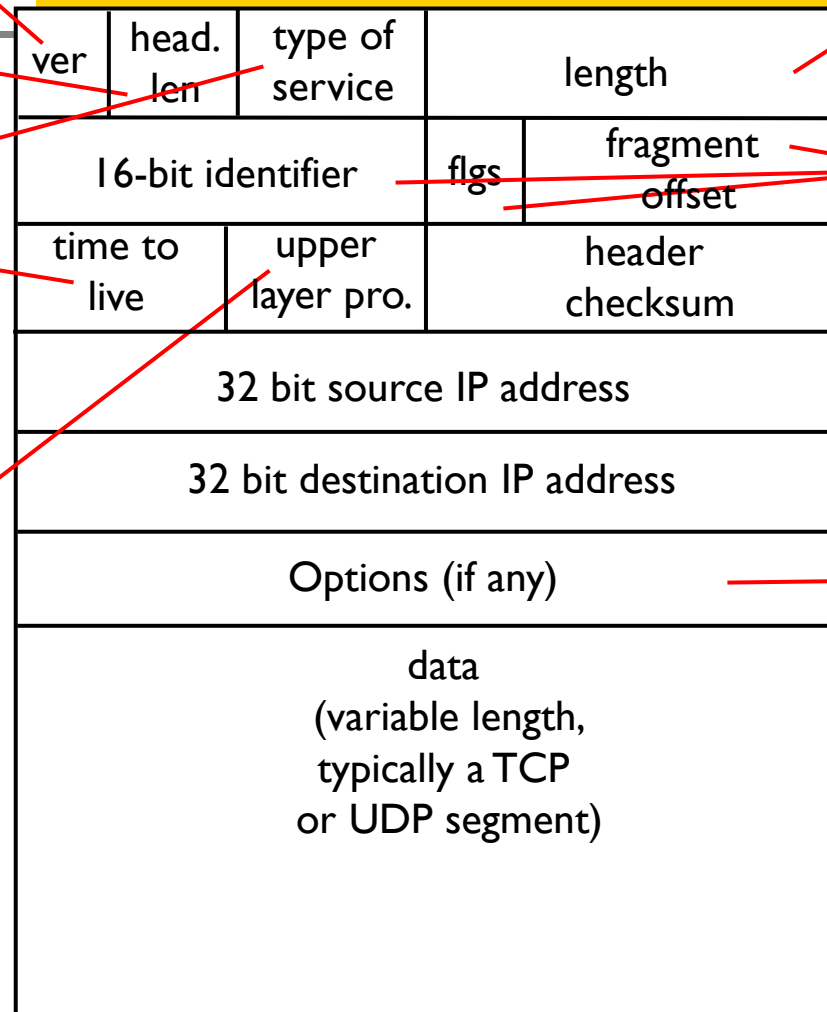
- ☐ 20 bytes of TCP
- ☐ 20 bytes of IP
- ☐ = 40 bytes + app layer message

32 bits

total datagram
length (bytes)

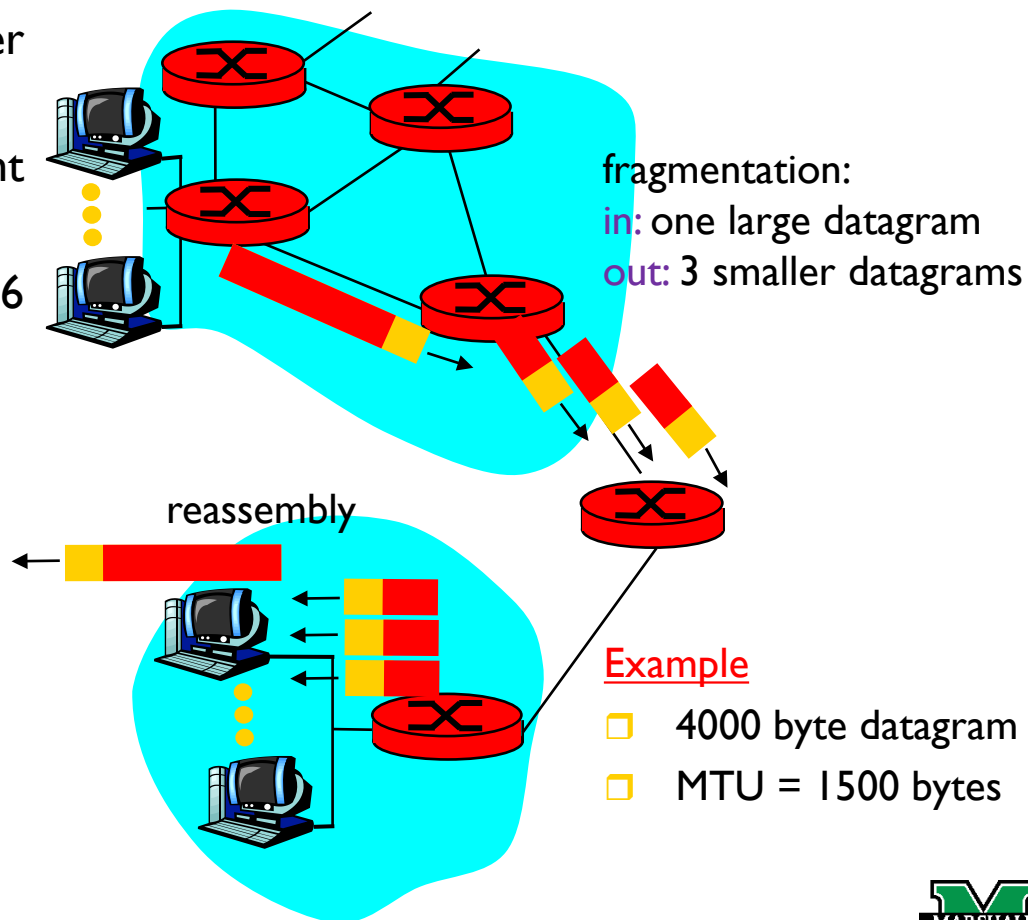
for
fragmentation/
reassembly

E.g. timestamp,
record route
taken, specify
list of routers
to visit.



IP Fragmentation & Reassembly

- network links have MTU (max. transfer unit) - largest possible link-level **frame**
 - different link types have different MTUs
 - e.g., some wide-area link – 576 bytes
- large IP datagram divided (“fragmented”) within network
 - one datagram becomes several datagrams
 - “reassembled” only at **final destination**
 - IP header bits used to identify order related fragments



IP Fragmentation & Reassembly (cont.)

Example

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

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

one large datagram becomes
several smaller datagrams

1480 bytes in data field + 20 bytes of IP header

$$\text{offset} = 185 = 1480 / 8$$

$$\text{offset} = 370 = 2960 / 8$$

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

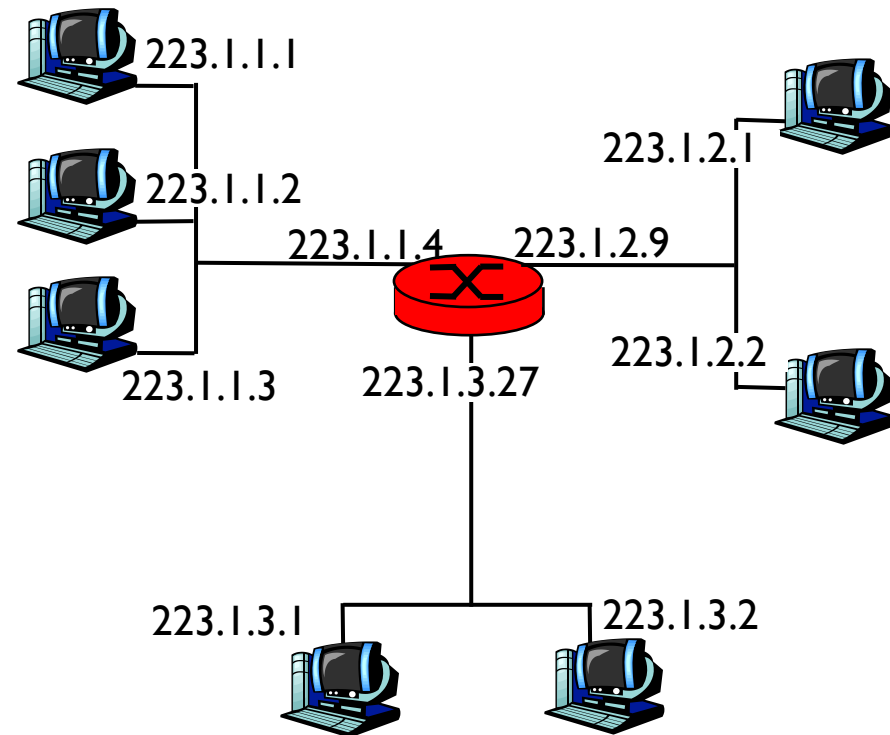
	length = 1500	ID = x	fragflag = 1	offset = 185	
--	------------------	-----------	-----------------	-----------------	--

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

offset is measured in terms of 8 bytes

IP address is technically associated with an interface, rather than with the host of router containing that interface!

- **IP address:** 32-bit identifier for host and 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***


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

dotted-decimal notation: each byte is written in decimal form and is separated by a dot from other bytes

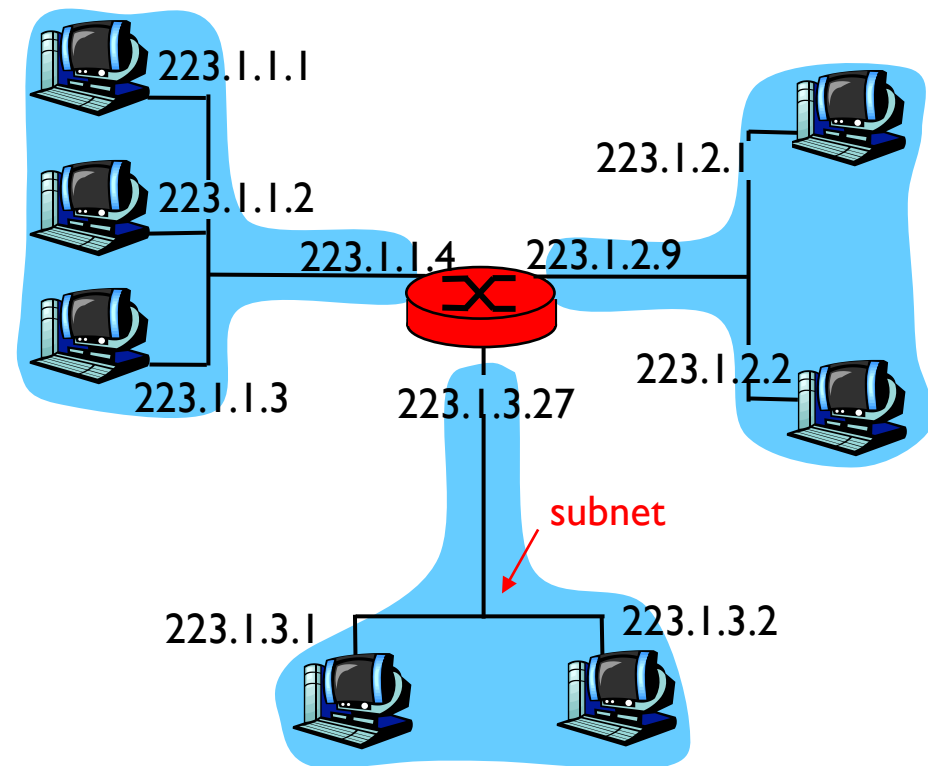
Subnets

■ What's a subnet ?

- the network interconnecting several hosts and routers
- device interfaces with **same** subnet part of IP address
- can physically reach each other without intervening router

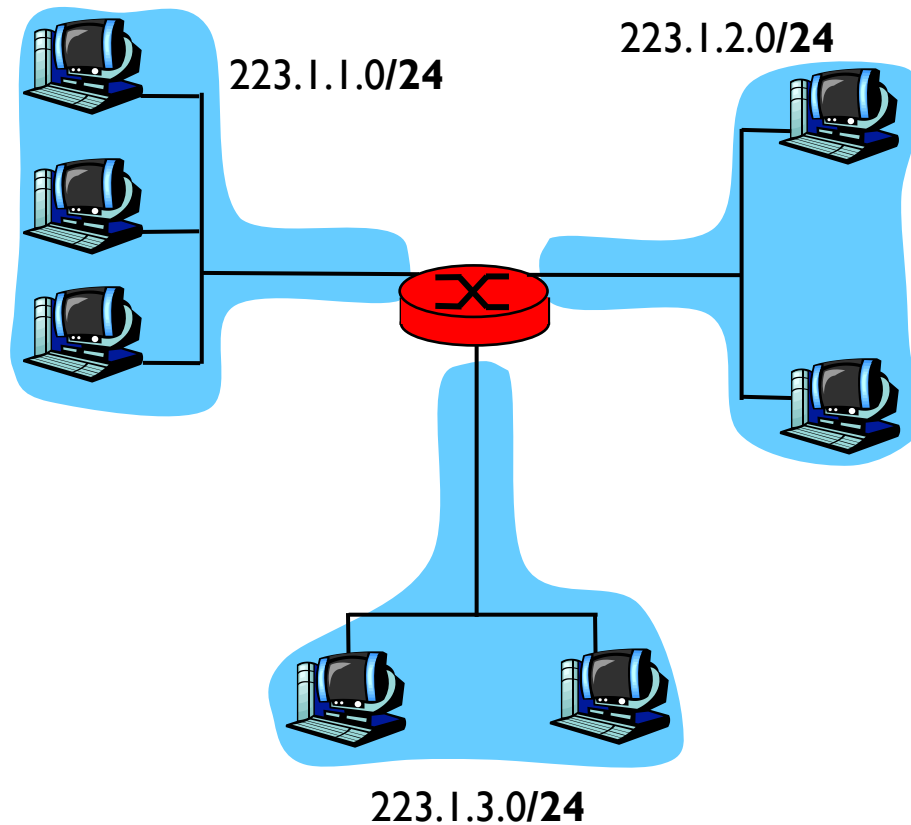
■ IP address:

- subnet part (high order bits)
- host part (low order bits)



network consisting of 3 subnets

Subnets (cont.)



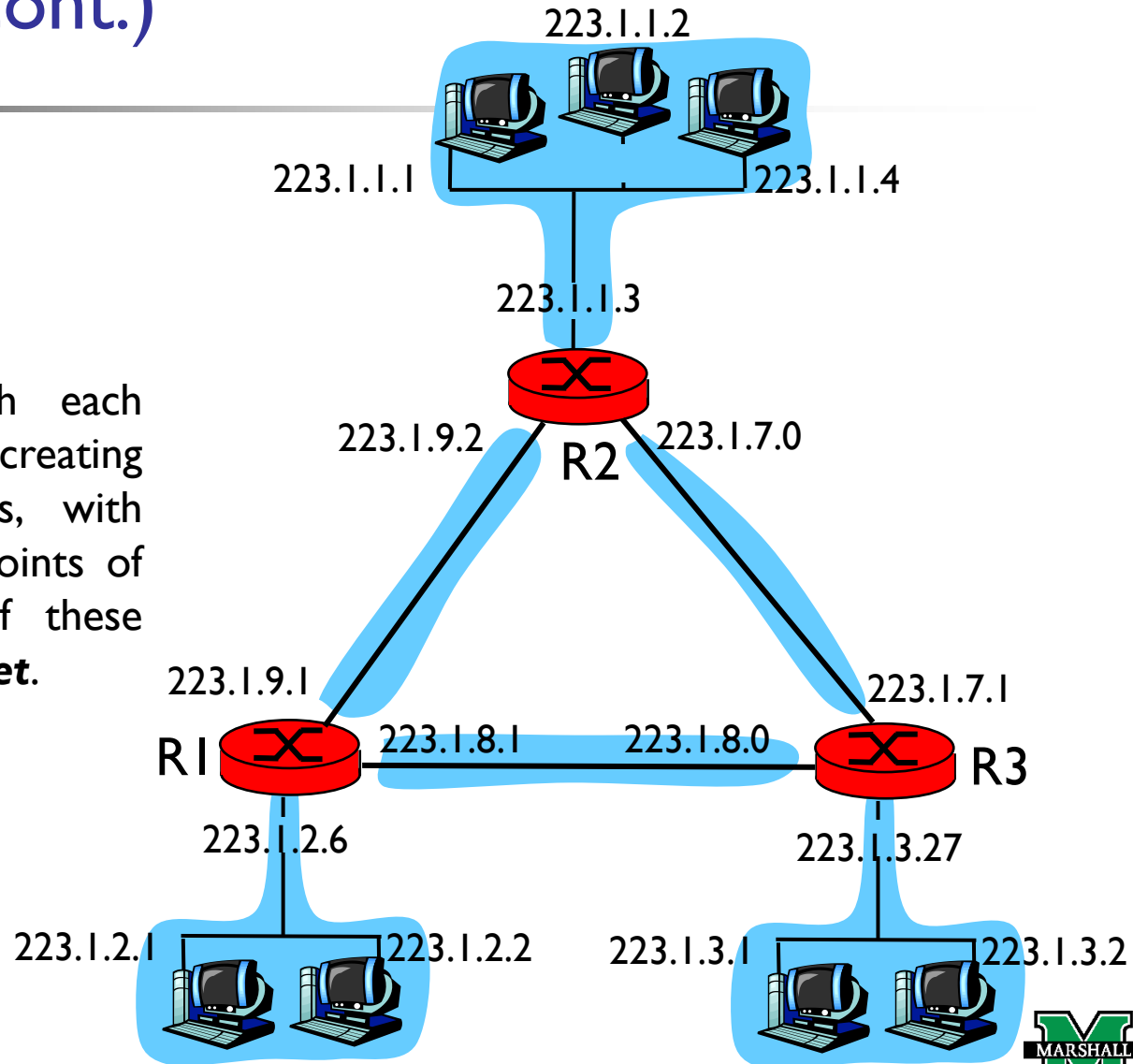
Subnet mask: */24*, indicating the leftmost 24 bits of the 32-bit quantity define the subnet address.

Subnets (cont.)

Q: How many subnets?

A: 6

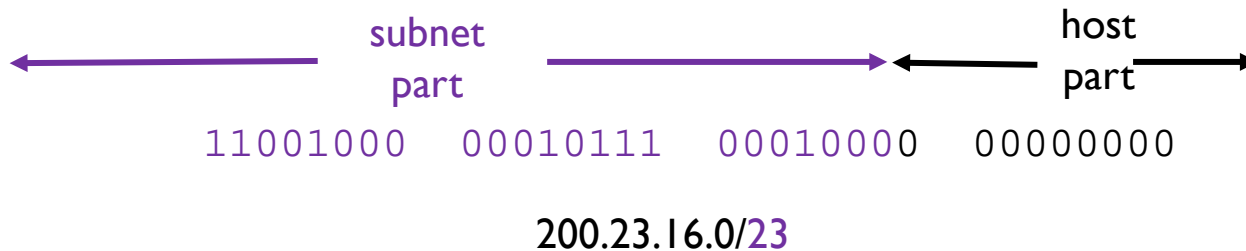
To determine the subnets, detach each interface from its host or router, creating islands of isolated networks, with interfaces terminating the end points of the isolated networks. Each of these isolated networks is called a **subnet**.





IP Addressing: CIDR

- Internet's address assignment strategy
- **CIDR: Classless InterDomain Routing**
 - generalizes the notion of subnet addressing
 - address format: **a.b.c.d/x**, where **x** is # bits in subnet portion of address





IP Addresses: How to get one?

- Q: How does **network** get subnet part of IP addr?
 - A: get allocated portion of its provider ISP's address space

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



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



IP Addresses: How to get one? (cont.)

- once an organization has obtained a block of address,
 - assign individual IP addresses to the host and router interfaces
- Q: How does a **host** get IP address?
 - hard-coded by system admin in a file
 - **DHCP**: **D**ynamic **H**ost **C**onfiguration **P**rotocol: dynamically get address from a server
 - “plug-and-play”

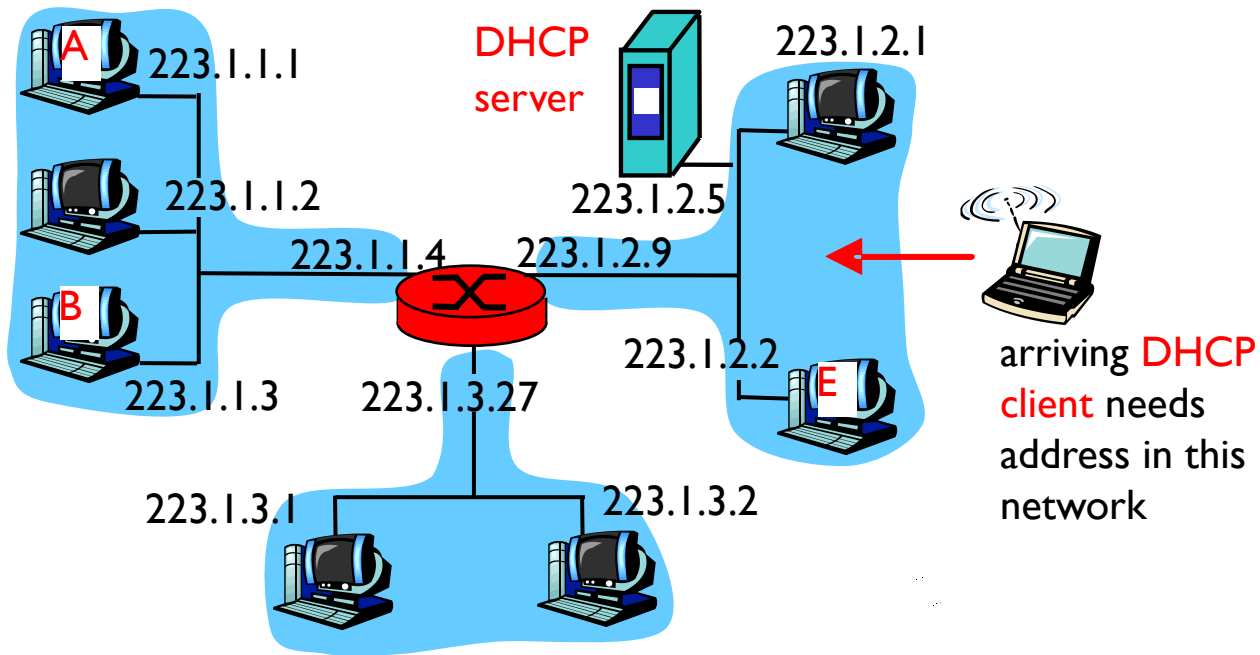


DHCP:

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
- DHCP overview:
 - host broadcasts “**DHCP server discover**” msg
 - DHCP server responds with “**DHCP server offer**” msg
 - host requests IP address: “**DHCP request**” msg
 - DHCP server sends address: “**DHCP ack**” msg

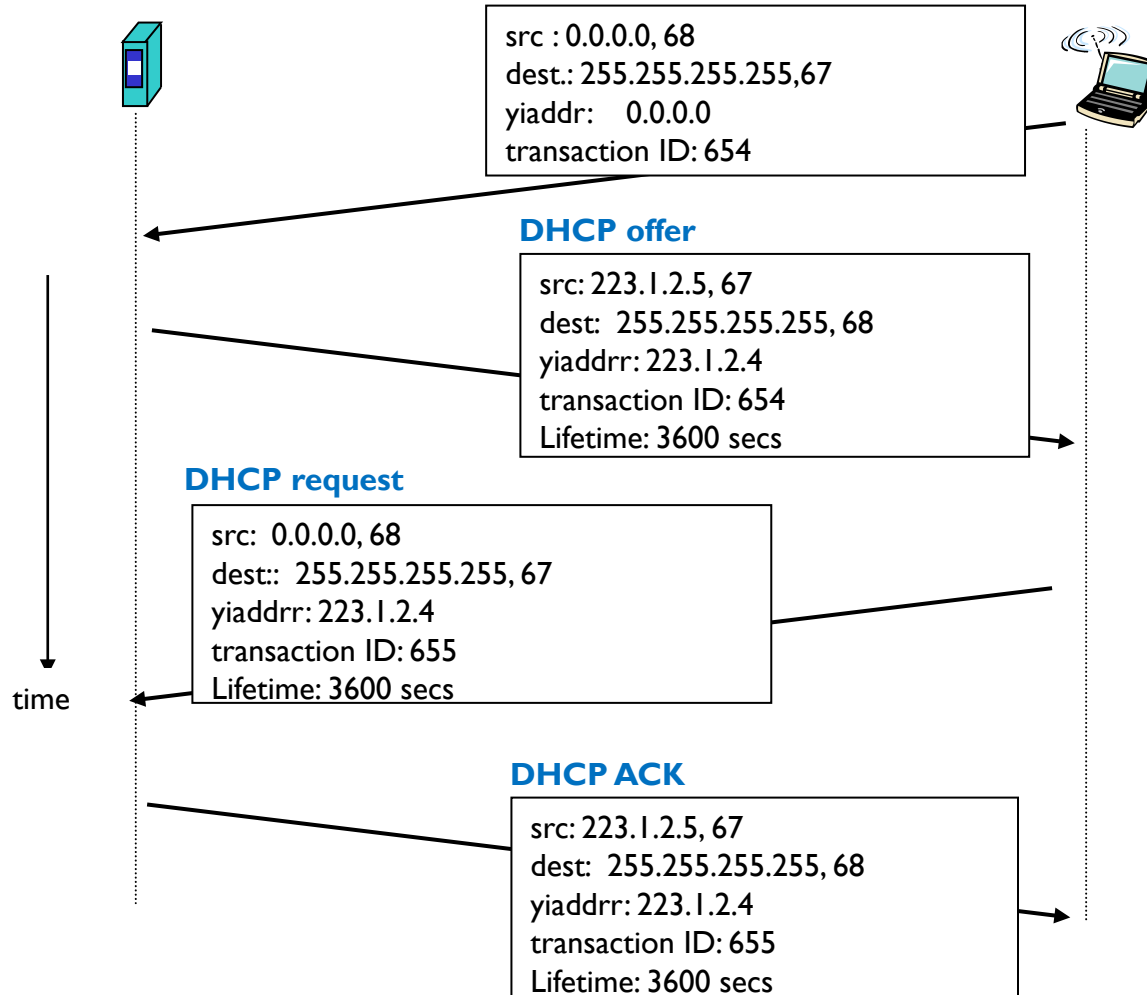
DHCP Client-Server Scenario



DHCP Client-Server Scenario

DHCP server: 223.1.2.5

arriving
client

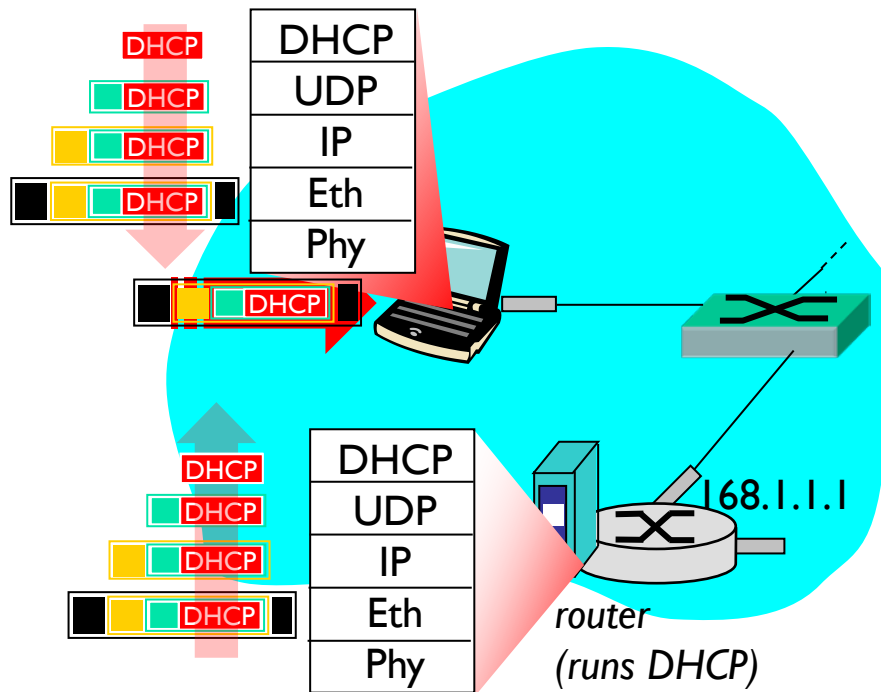




DHCP: More Than IP Address

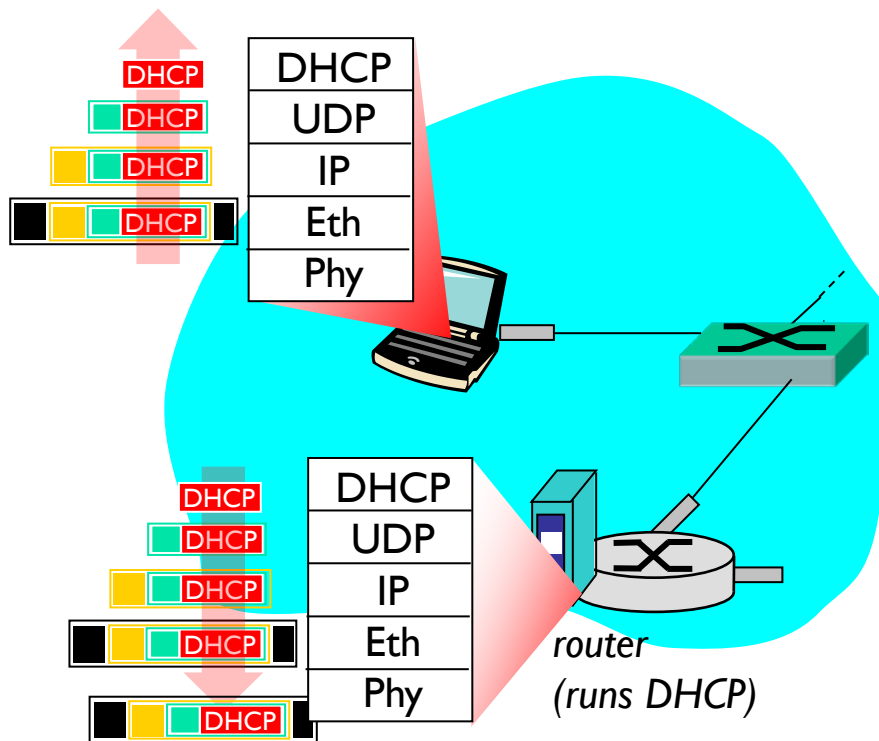
- DHCP can return more than just allocated IP address on subnet:
 - address of first-hop router for client
 - name and IP address of DNS sever
 - network mask (indicating network versus host portion of address)

DHCP: Example



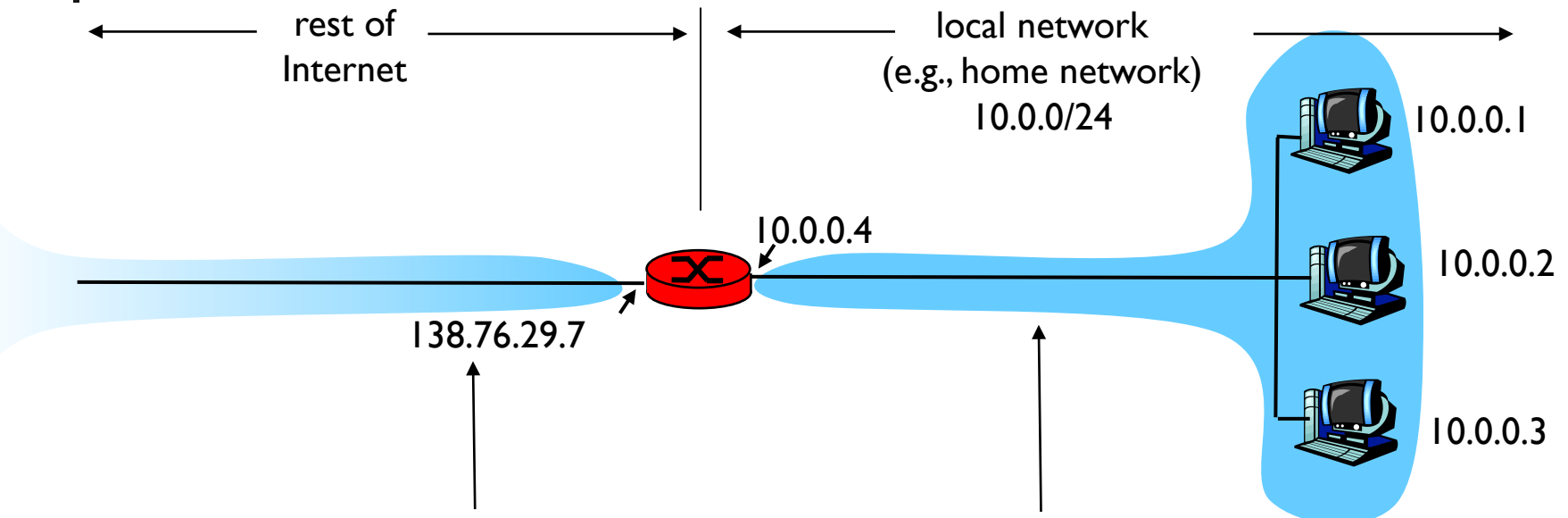
- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in **UDP**, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFFFFFF) on LAN, received at router running DHCP server

DHCP: Example (cont.)



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

NAT: Network Address Translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7.

all traffic *entering* local network have *same* destination address: 138.76.29.7.

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

The address space **10.0.0.0/8** is reserved for a private networks or a realm with private address

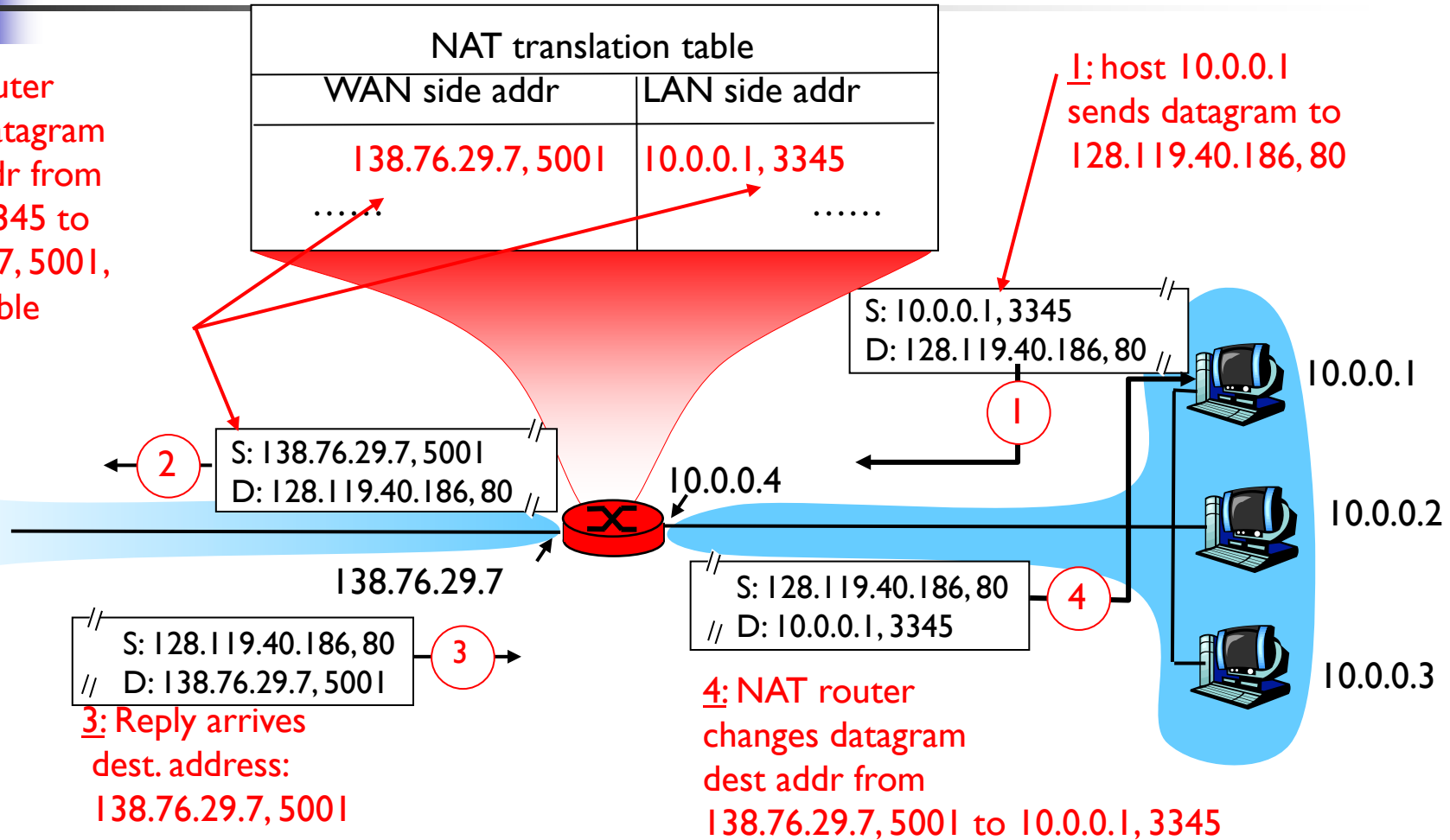


NAT: Network Address Translation (cont.)

- **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, e.g., NAT-enabled router
 - 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 network not explicitly addressable and visible by outside world

NAT: Network Address Translation (cont.)

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table





NAT: Network Address Translation (cont.)

- **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



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 and reassembly allowed at intermediate routers
 - these operations can be performed only by the **source** and **destination**

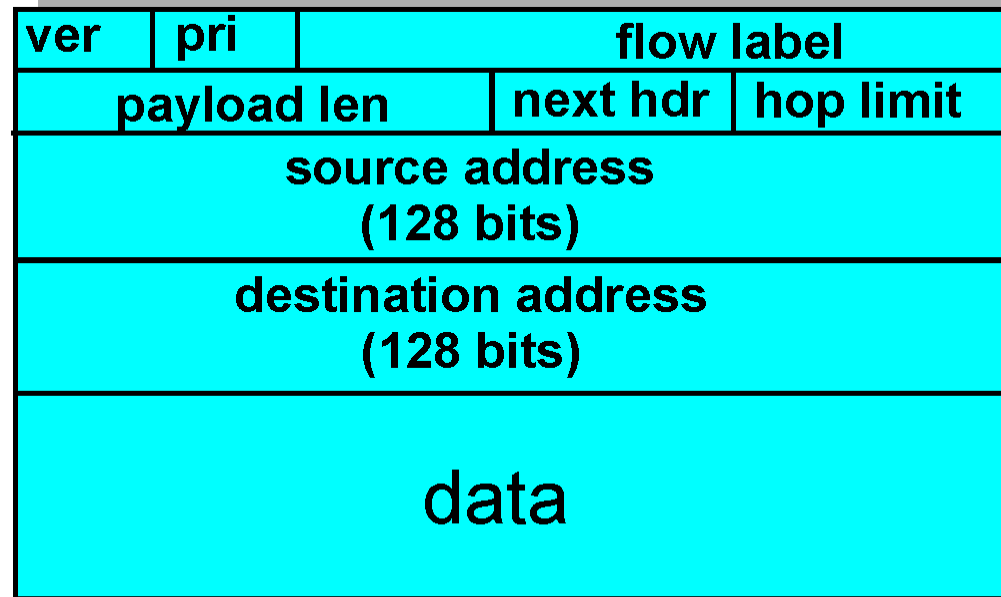


IPv6 Header

Priority: identify priority among datagrams in flow

Flow Label: identify a flow of datagrams

Next header: identify upper layer protocol for data (e.g., TCP or UDP)



← 32 bits →



Other Changes from IPv4

- **Checksum:** removed entirely to reduce processing time at each hop
 - the transport & link layers performs checksum
- **ICMPv6:** new version of ICMP
 - additional message types, e.g., “Packet Too Big” due to no fragmentation / reassembly
 - If an IPv6 datagram is too big to forward?
 - simply drop & send a “Packet Too Big” ICMP error message back
 - the sender will send a smaller IP datagram
 - → speed up IP forwarding within the network