CS5283 – Week 6

IP Addressing

Outline

- Possible option to discuss for remaining class schedule
  - Could have no class in 2 weeks (October 14), Vanderbilt fall break, not technically scheduled this way for this class, as for some reason, last day of this program is 12/7
  - To have 14 synchronous sessions, our last class would be 12/9 instead of 12/2
- Project discussion: proposal due October 14
- Quiz 2 available, due next class start October 7, covers mostly weeks 3/4, some week 5; self assessment posted
- Q/A and demos on homework 2

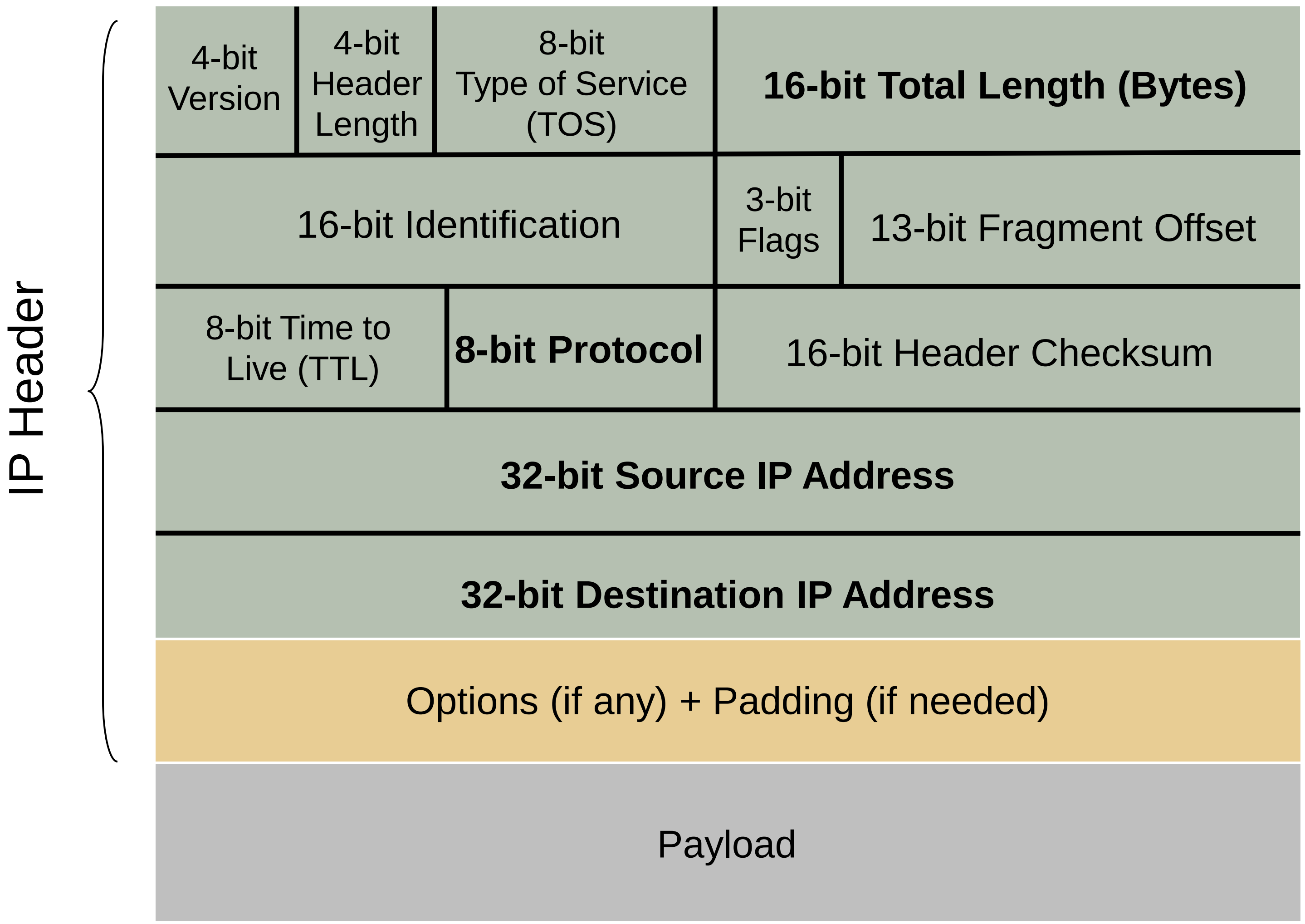
- Q/A and breakouts on asynchronous content
  - IP Addressing: packet headers, fragmentation, hierarchical addressing
  - IP Routing: classful routing, classless (CIDR), routers/forwarding tables, IP address allocation and scalability, network address translation (NAT)

IP Packet Headers

IP Addressing: Overview

- IP fragmentation
- IP (IPv4) addressing
  - Classless Inter-Domain Routing (CIDR)
  - Network address translation (NAT)

IP Packet Structure



IP Packet Structure

4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	**16-bit Total Length (Bytes)**
16-bit Identification		3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)	**8-bit Protocol**	16-bit Header Checksum	
32-bit Source IP Address			
32-bit Destination IP Address			
Options (if any) + Padding (if needed)			
Payload			

IP Packet Header Fields

- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically “4” (for IPv4), and sometimes “6” (for IPv6)
- Header length (4 bits)
  - Number of 32-bit words in the header
  - Typically “5” (for a 20-byte IPv4 header)
  - Can be more when IP **options** are used
- Type of service (8 bits)
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer

IP Packet Structure

<table>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type of Service (TOS)</td><td colspan="2">16-bit Total Length (Bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8-bit Time to Live (TTL)</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any) + Padding (if needed)</td></tr>
<tr><td colspan="5">Payload</td></tr>
</table>

IP Packet Headers

The End

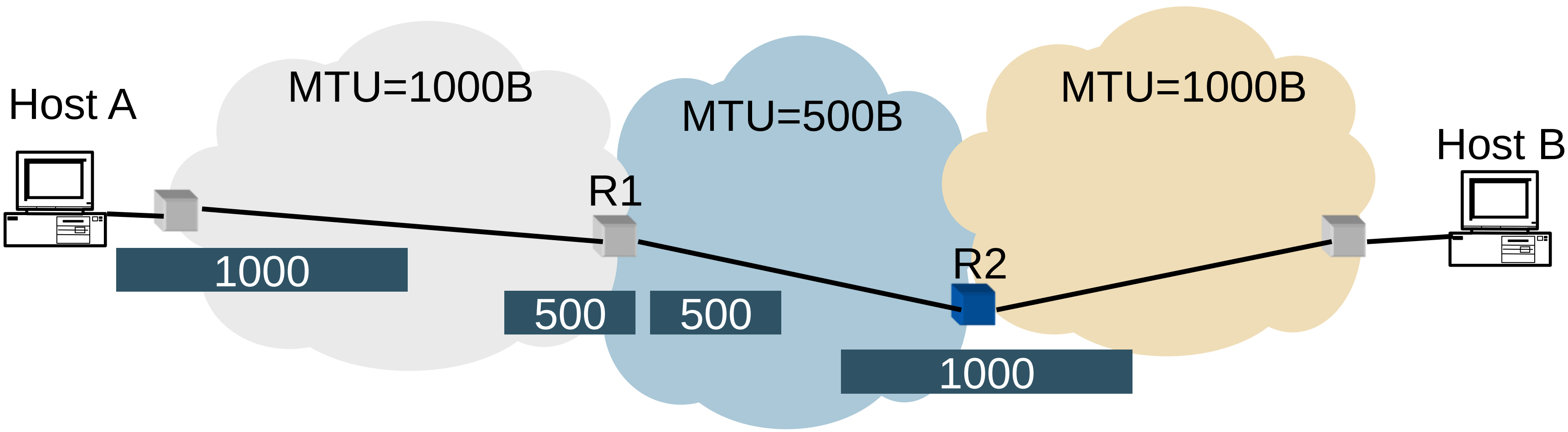
IP Fragmentation

IP Packet Header Fields (cont.)

- Total length (16 bits)
  - Number of bytes in the packet
  - Maximum size is 65,535 bytes (2^{16} -1)
  - … though underlying links may impose smaller limits
- Fragmentation: when forwarding a packet, an Internet router can **split** it into multiple pieces (“fragments”) if too big for the next hop link
- Fragmentation information (32 bits)
  - Packet **identifier**, **flags**, and fragment **offset**

Where Does Reassemble Happen?

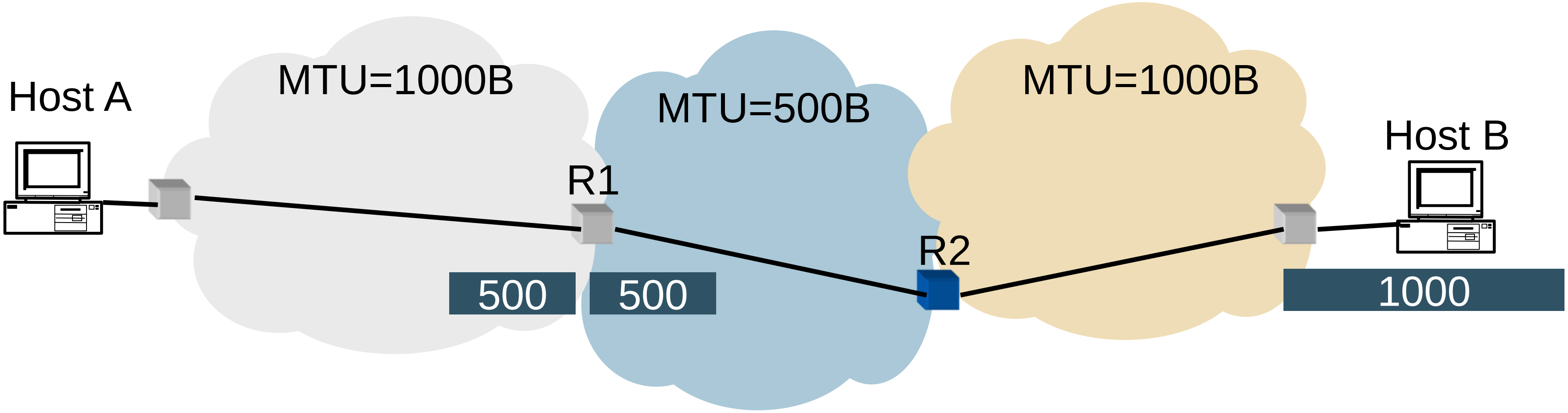
A1: router R2



MTU (maximum transfer unit) = Maximum packet size handled by the network

Where Does Reassemble Happen?

A2: end-host B (receiver)

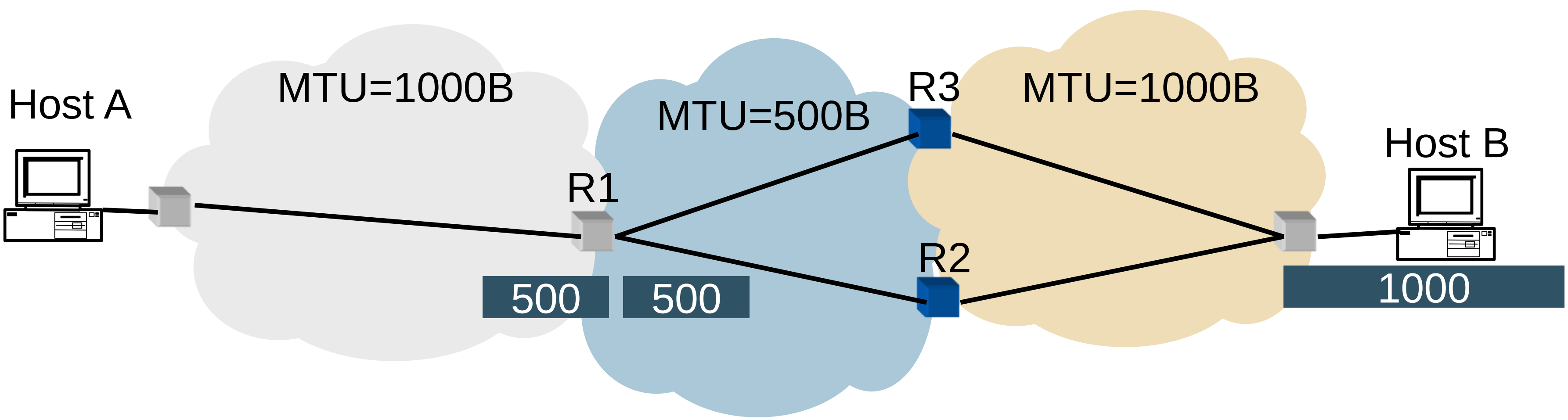


MTU (maximum transfer unit) = Maximum packet size handled by the network

Where Does Reassemble Happen?

A2: correct answer

- Fragments can travel across different paths!
- Why reassemble in network, if it could be further fragmented later?



MTU (maximum transfer unit) = Maximum packet size handled by the network

IP Packet Structure

<table>
<tr><td>4-bit
Version</td><td>4-bit
Header
Length</td><td>8-bit
Type of Service
(TOS)</td><td colspan="2">16-bit Total Length (Bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>RF/DF/
MF</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8-bit Time to
Live (TTL)</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any) + Padding (if needed)</td></tr>
<tr><td colspan="5">Payload</td></tr>
</table>

Fragmentation

- Identifier (16 bits): used to tell which fragments belong together
- Flags (3 bits):
  - Reserved **(RF):** unused bit (why “reserved”?)
  - Don’t fragment **(DF):** instruct routers to **not** fragment the packet even if it won’t fit
    - Instead, they ***drop*** the packet and send back a “Too Large” ICMP (Internet Control Message Protocol) control message
    - Forms the basis for “Path MTU Discovery,” covered later
  - More (**MF**): this fragment is not the last one

Fragmentation (cont.)

- Offset (13 bits): what part of datagram this fragment covers ***in 8-byte units?***
- How can a receiver differentiate between the last fragment of a packet and an unfragmented packet?
  - Neither of them has the MF flag set …
  - If the offset is 0, it’s an unfragmented packet, otherwise it must be the last fragment

Example of Fragmentation (Part I)

- Suppose we have a 4,000-byte datagram sent from host 1.2.3.4 to host 3.4.5.6 …

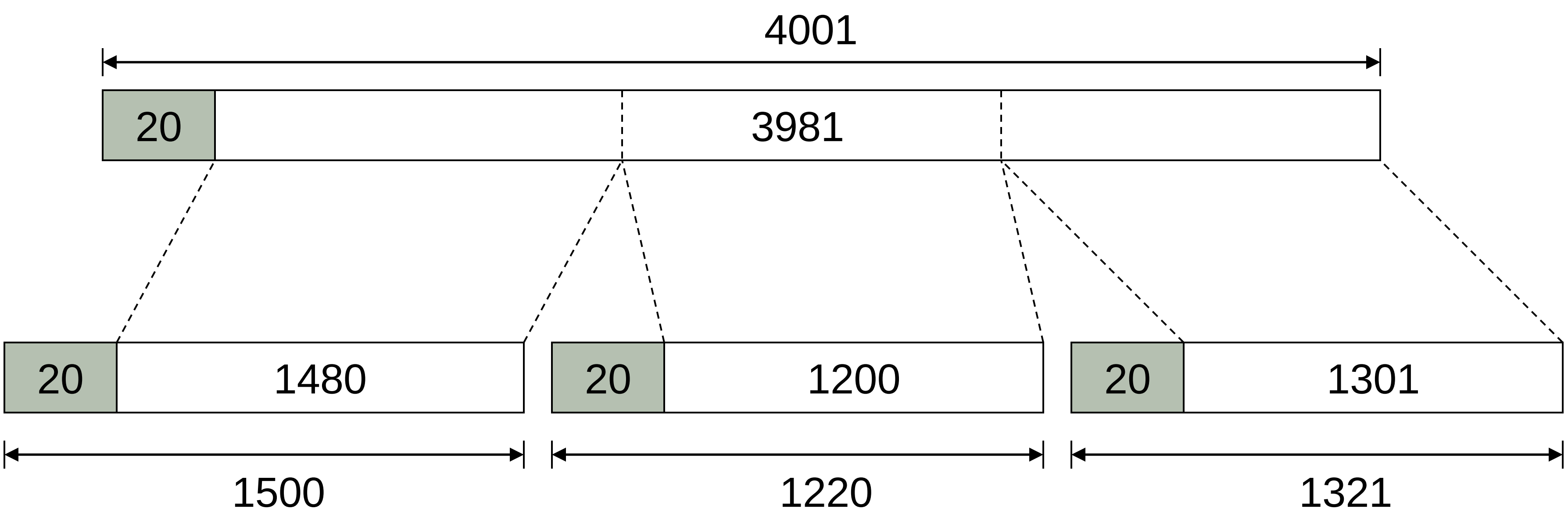
<table>
<tr><td>Version
4</td><td>Header Length
5</td><td>Type of Service
0</td><td colspan="2">**Total Length: *4000***</td></tr>
<tr><td colspan="3">Identification: ***56273***</td><td>R/D/M
0/0/0</td><td>Fragment Offset: ***0***</td></tr>
<tr><td colspan="2">TTL
127</td><td>**Protocol**
6</td><td colspan="2">Checksum: ***44019***</td></tr>
<tr><td colspan="5">**Source Address: *1.2.3.4***</td></tr>
<tr><td colspan="5">**Destination Address: *3.4.5.6***</td></tr>
</table>

(3980 more bytes here)

- … and it traverses a link that limits datagrams to 1,500 bytes

Example of Fragmentation (Part II)

- Datagram split into three pieces
- Example:



Example of Fragmentation (Part III)

- Possible first piece:

<table>
<tr><td>Version
4</td><td>Header Length
5</td><td>Type of Service
0</td><td colspan="2">Total Length: 1500
(20 + 1480 = 1500)</td></tr>
<tr><td colspan="3">Identification: 56273</td><td>R/D/M
0/0/1</td><td>Fragment Offset: 0</td></tr>
<tr><td colspan="2">TTL
127</td><td>Protocol
6</td><td colspan="2">Checksum: xxx</td></tr>
<tr><td colspan="5">Source Address: 1.2.3.4</td></tr>
<tr><td colspan="5">Destination Address: 3.4.5.6</td></tr>
</table>

Example of Fragmentation (Part IV)

- Possible second piece:

<table>
<tr><td>Version
4</td><td>Header Length
5</td><td>Type of Service
0</td><td colspan="2">**Total Length: *1220***
(20 + 1200 = 1200)</td></tr>
<tr><td colspan="3">Identification: ***56273***</td><td>R/D/M
0*/*0*/1**</td><td>Fragment Offset: ***185
(185 * 8 = 1480)</td></tr>
<tr><td>TTL
127</td><td colspan="2">**Protocol**
6</td><td colspan="2">Checksum: **yyy**</td></tr>
<tr><td colspan="5">**Source Address: *1.2.3.4***</td></tr>
<tr><td colspan="5">**Destination Address: *3.4.5.6***</td></tr>
</table>

Example of Fragmentation (Part V)

- Possible third piece:

<table>
<tr><td>Version
4</td><td>Header Length
5</td><td>Type of Service
0</td><td colspan="2">**Total Length: *1321***
(20 + 3881 – 2680 = 1321)</td></tr>
<tr><td colspan="3">Identification: ***56273***</td><td>R/D/M
0/***0***/**0**</td><td>Fragment Offset: ***335***
(335 * 8 = 2680)</td></tr>
<tr><td colspan="2">TTL
127</td><td>**Protocol**
6</td><td colspan="2">Checksum: **zzz**</td></tr>
<tr><td colspan="5">**Source Address: *1.2.3.4***</td></tr>
<tr><td colspan="5">**Destination Address: *3.4.5.6***</td></tr>
</table>

Some Fragmentation Design Decisions

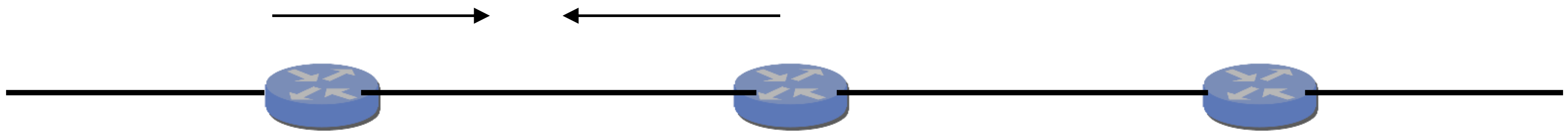
- Q: Where are the fragments reassembled?
- A: Usually at the final destination. Why?
  - Reason 1: because different fragments can take different paths through the network—the whole collection might only be available at the receiver
  - Reason 2: reassembly at any node may be premature, as subsequent low-MTU links might require fragmentation again
- Q: Why use a byte-offset for fragments rather than numbering each fragment?
- Answer 1 (more fundamental): It allows further fragmentation of fragments.
- Answer 2: With a byte offset, the receiver can lay down the bytes in memory when they arrive (avoids memcpy).

IP Packet Structure

<table>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type of Service (TOS)</td><td colspan="2">16-bit Total Length (Bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8-bit Time to Live (TTL)</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
<tr><td colspan="5">Payload</td></tr>
</table>

Time to Live (TTL) Field (8 Bits)

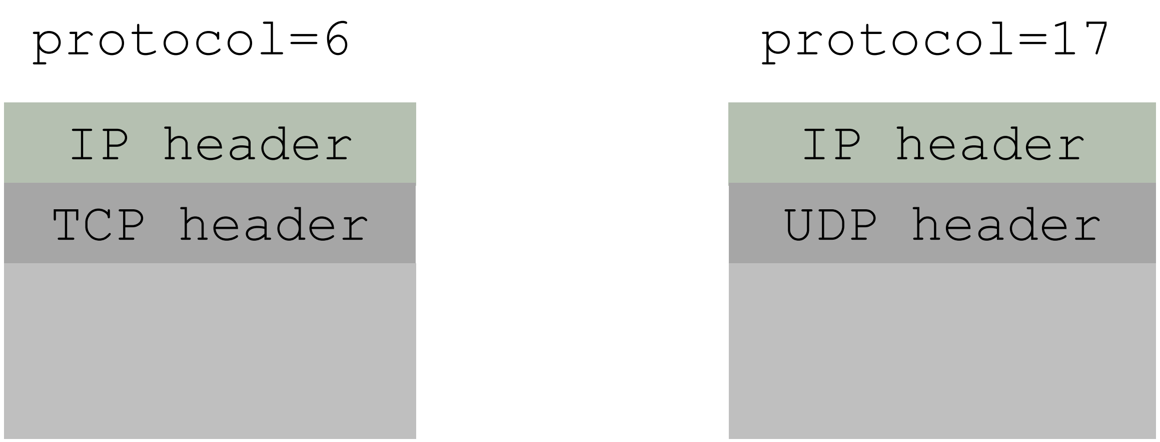
- Potentially lethal problem
  - Forwarding loops can cause packets to cycle forever
  - As these accumulate, eventually consume **all** capacity



- Time to live field in the packet header
  - Decremented at each hop, the packet is discarded if it reaches 0
  - … and “time exceeded” message is sent to the source
    - Using ICMP control message; basis for **traceroute**

IP Packet Header Fields (cont.)

- Protocol (8 bits)
  - Identifies the higher-level protocol
    - E.g., “6” for the Transmission Control Protocol (TCP)
    - E.g., “17” for the User Datagram Protocol (UDP)
  - Important for demultiplexing at receiving host
    - Indicates what kind of header to expect next



IP Packet Header Fields (cont.)

- Checksum (16 bits)
  - Complement of the *one’s-complement* sum of all 16-bit words in the IP **packet header**
- Each router computes the one’s-complement sum of the entire header *including checksum* …
  - … should get 0 (or 0xffff)
  - If not, the router **discards** the packet as corrupted
    - So it doesn’t act on bogus information

IP Packet Structure

<table>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type of Service (TOS)</td><td colspan="2">16-bit Total Length (Bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8-bit Time to Live (TTL)</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any) + Padding (if needed)</td></tr>
<tr><td colspan="5">Payload</td></tr>
</table>

IP Packet Header (cont.)

- Two IP addresses
  1. Source IP address (32 bits)
  2. Destination IP address (32 bits)
- Destination address
  - Unique **identifier/locator** for the receiving host
  - Allows each node to make forwarding decisions
- Source address
  - Unique identifier/locator for the sending host
  - The recipient can decide whether to accept the packet
  - Enables the recipient to send a reply back to source

IP Fragmentation

The End

Hierarchical Addressing

Goals

- IP addresses
  - Dotted-quad notation
  - IP prefixes for aggregation
  - Classful addresses
  - Classless Inter-Domain Routing (CIDR)
  - Special-purpose address blocks
  - Network address translation (NAT)
- Address allocation
  - Hierarchy by which address blocks are given out
  - Finding information about an allocation

IP Packet Structure

<table>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type of Service (TOS)</td><td colspan="2">16-bit Total Length (Bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8-bit Time to Live (TTL)</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any) + Padding (if needed)</td></tr>
<tr><td colspan="5">Payload</td></tr>
</table>

IP Packet Header (cont.)

- Two IP addresses
  1. Source IP address (32 bits)
  2. Destination IP address (32 bits)
- Destination address
  - Unique **identifier/locator** for the receiving host
  - Allows each node to make forwarding decisions
- Source address
  - Unique identifier/locator for the sending host
  - The recipient can decide whether to accept the packet
  - Enables the recipient to send a reply back to source

Designing IP's Addresses

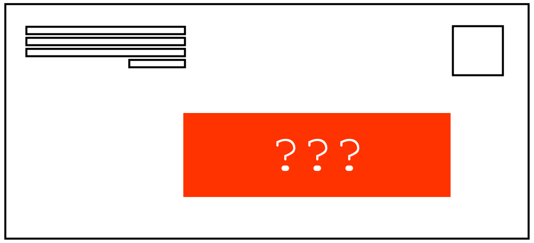
- Question 1: What should an address be associated with?
  - E.g., a telephone number is not associated with a person but with a **handset**
- Question 2: What **structure** should addresses have? What are the **implications** of different types of structure?
- Question 3: **Who** determines the particular addresses used in the global Internet? What are the implications of how this is done?

IP Addresses (IPv4)

- A unique 32-bit number
- Identifies an ***interface*** (on a host, on a router, …)
- Represented in ***dotted-quad*** notation; e.g., **12.34.158.5:**

12	34	158	5
00001100	**00100010**	**10011110**	**00000101**

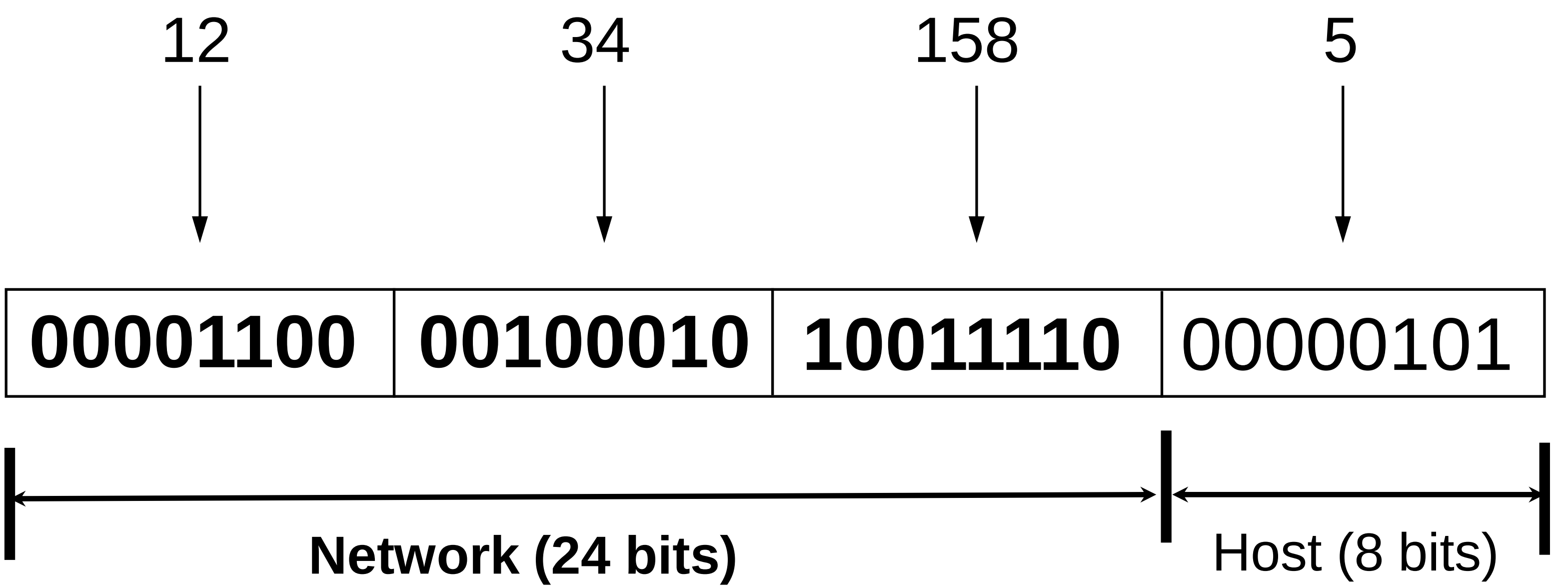
Hierarchical Addressing in U.S. Mail



- Addressing the U.S. mail
  - Zip code: 37212
  - Street: 16th Ave S
  - Building on street: 1025
  - Ste: 102
  - ATTN: Taylor Johnson
- Forwarding the U.S. mail
  - Deliver the letter to the post office in the zip code
  - Assign the letter to the mailman covering the street
  - Drop the letter into the mailbox for the building/suite
  - Give the letter to the appropriate person

Hierarchical Addressing: IP Prefixes

- Divided into network (left) and host portions (right)
- 12.34.158.0/24 is a 24-bit ***prefix*** with 2^8 addresses
  - Terminology: ***“Slash 24”***



IP Address and a 24-Bit Subnet Mask

Address

12	34	158	5
00001100	**00100010**	**10011110**	00000101
11111111	**11111111**	**11111111**	00000000
255	255	255	0

Mask

Hierarchical Addressing

The End

Classful Routing

Classful Addressing

- Class A: if first byte in [0..127], assume /8 (top bit = 0)

*0********	********	********	********

  - Very large blocks (e.g., MIT has 18.0.0.0/8)
- Class B: first byte in [128..191] → assume /16 (top bits = 10)

*10*******	********	********	********

  - Large blocks (e.g., VU has* 129.59.0.0/16)
- Class C: [192..223] → assume /24 (top bits = 110)

*110******	********	********	********

  - Small blocks
  - The “swamp” (many European networks, due to history)

Classful Addressing (cont.)

- Class D: [224..239] (top bits 1110)

1110****	************	************	************

  - Multicast groups
- Class E: [240..255] (top bits 11110)

11110***	************	************	************

  - Reserved for future use
- What problems can classful addressing lead to?
  - Only comes in three sizes
  - Routers can end up knowing about a **lot** of class Cs

Classful Routing

The End

Classless Inter-Domain Routing (CIDR)

Classless Inter-Domain Routing (CIDR)

Use **arbitrary** length prefixes

Use two 32-bit numbers to represent a network.

Network number = IP address + Mask

IP Address : 12.4.0.0 IP Mask: 255.254.0.0

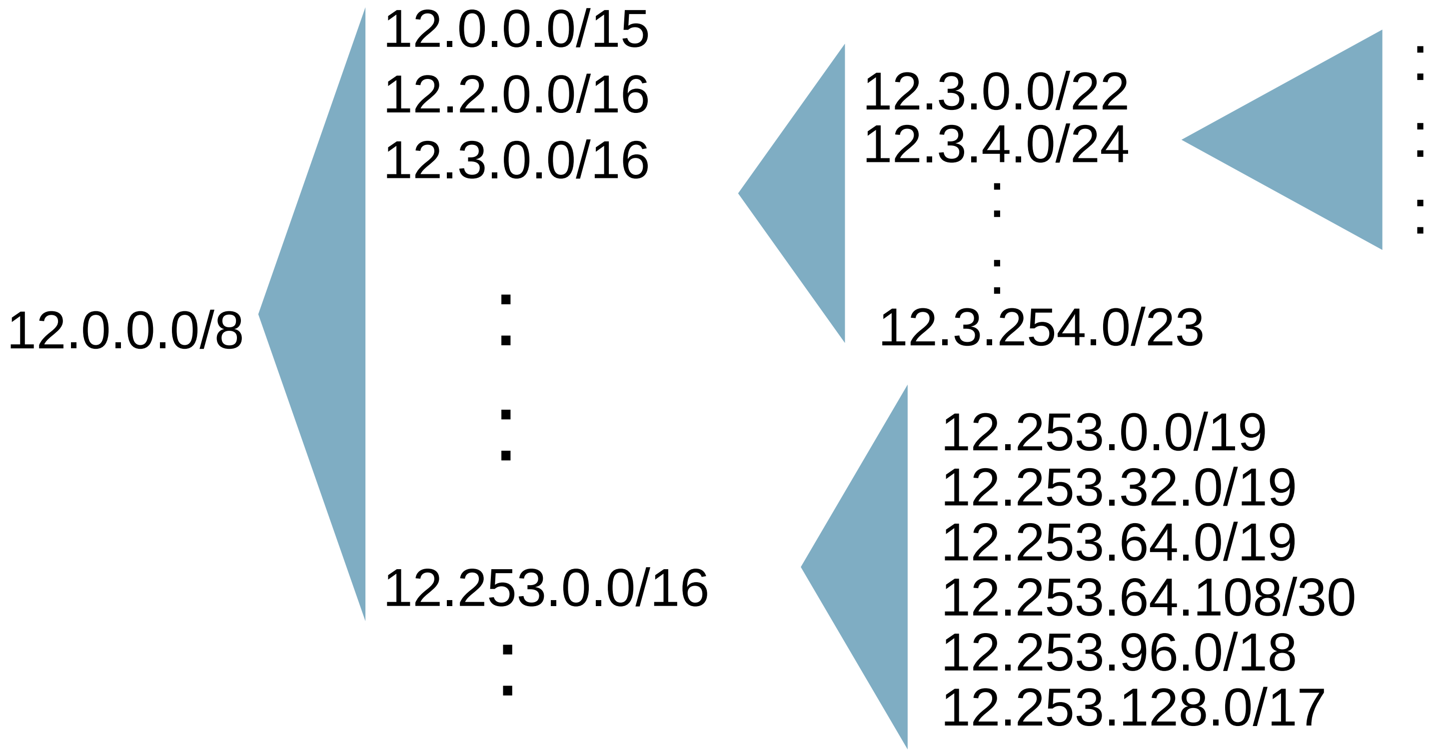
Address	00001100	00000100	00000000	00000000
Mask	11111111	11111110	00000000	00000000

Network Prefix | for hosts

Written as 12.4.0.0/15 or 12.4/15

CIDR: Hierarchal Address Allocation

- Prefixes are key to Internet scalability
  - Addresses allocated in contiguous chunks (prefixes)
  - Routing protocols and packet forwarding based on prefixes



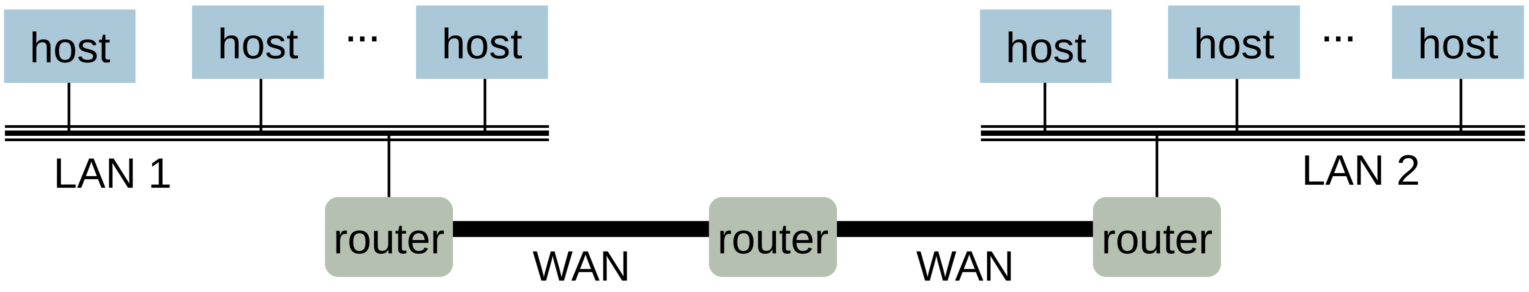
Classless Inter-Domain Routing (CIDR)

The End

Routers and Forwarding Tables

Addressing Hosts in the Internet

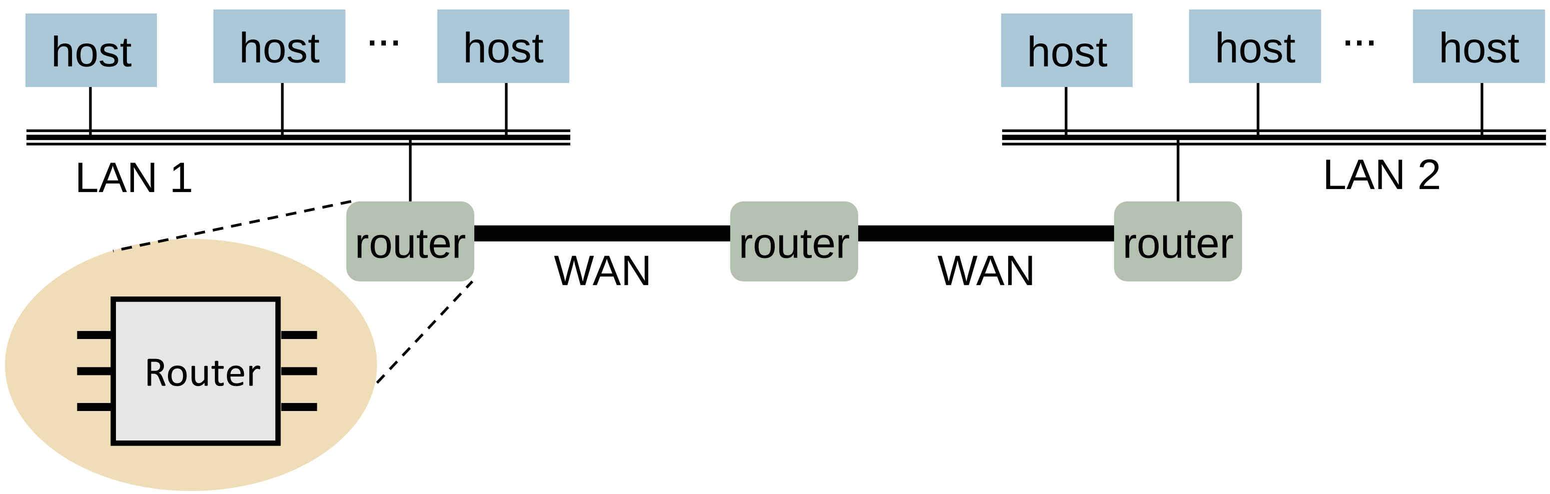
- The Internet is an “inter-network”
  - Used to connect *networks* together, not hosts
  - Needs a way to address a network (i.e., a group of hosts)



LAN: local area network
WAN: wide area network

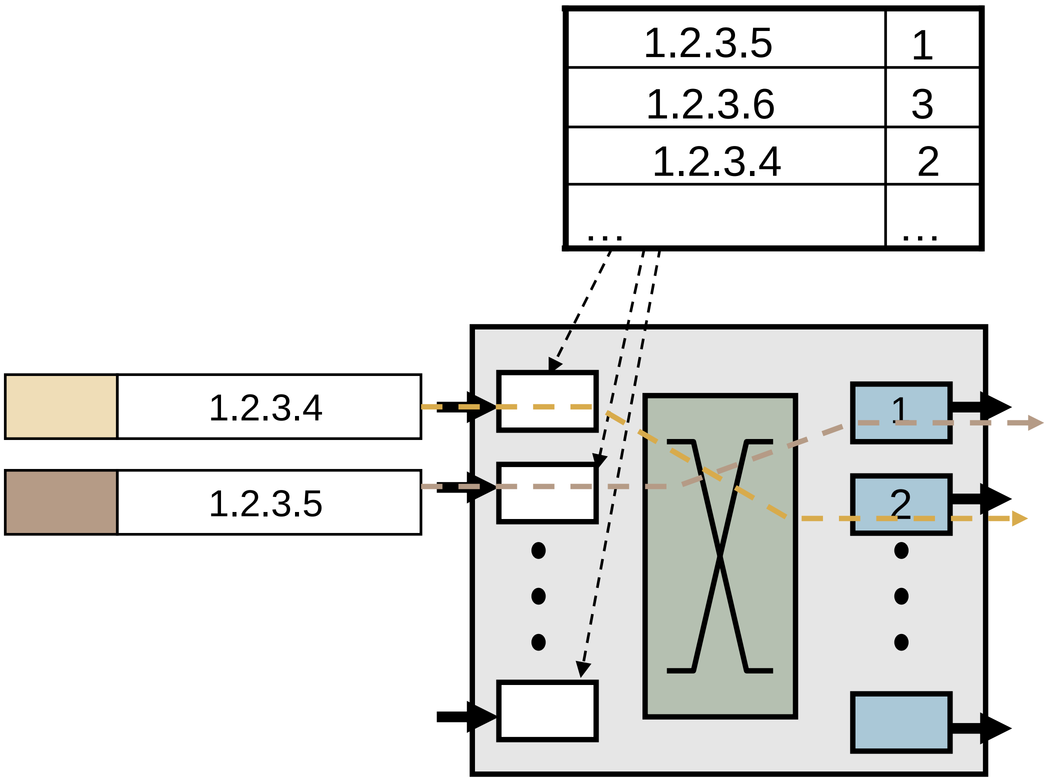
Routers

- A router consists of:
  - A set of input interfaces where packets arrive
  - A set of output interfaces from which packets depart
  - Some form of interconnect connecting inputs to outputs
- A router implements:
  - Forward packet to corresponding output interface
  - **(Manage bandwidth and buffer space resources)**



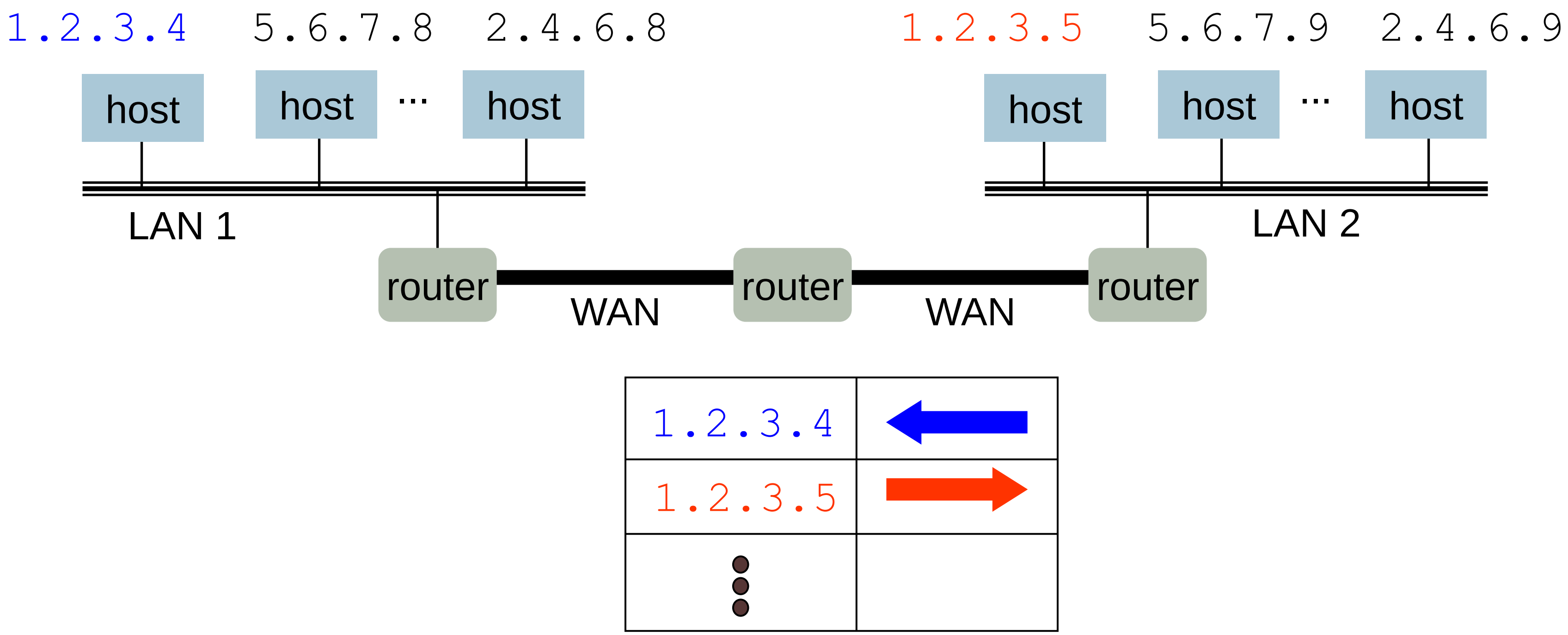
Forwarding Table

- Store a mapping between IP addresses and output interfaces
  - Forward an incoming packet based on its destination address



Scalability Challenge

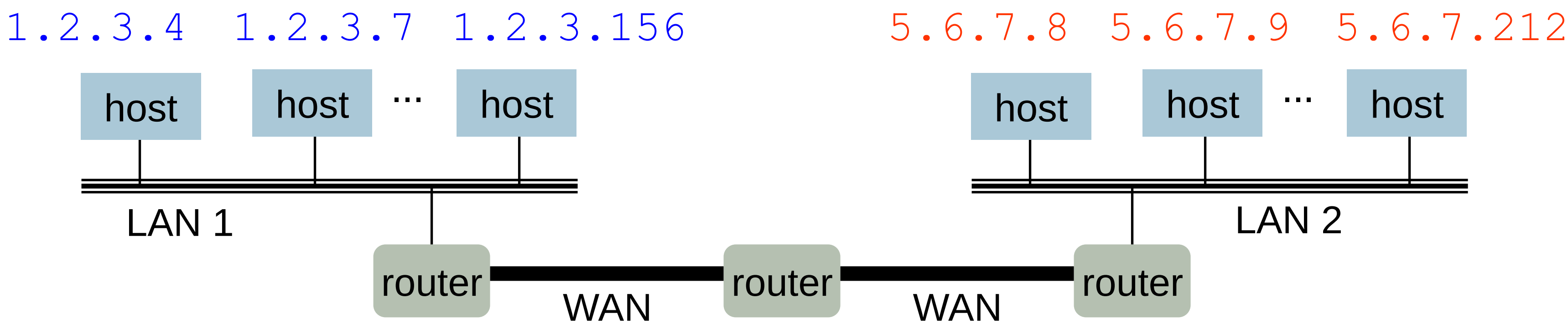
- Suppose hosts had arbitrary addresses
  - Then every router would need a lot of information …
  - … to know how to direct packets toward the host



forwarding table

Scalability Improved

- Number-related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN

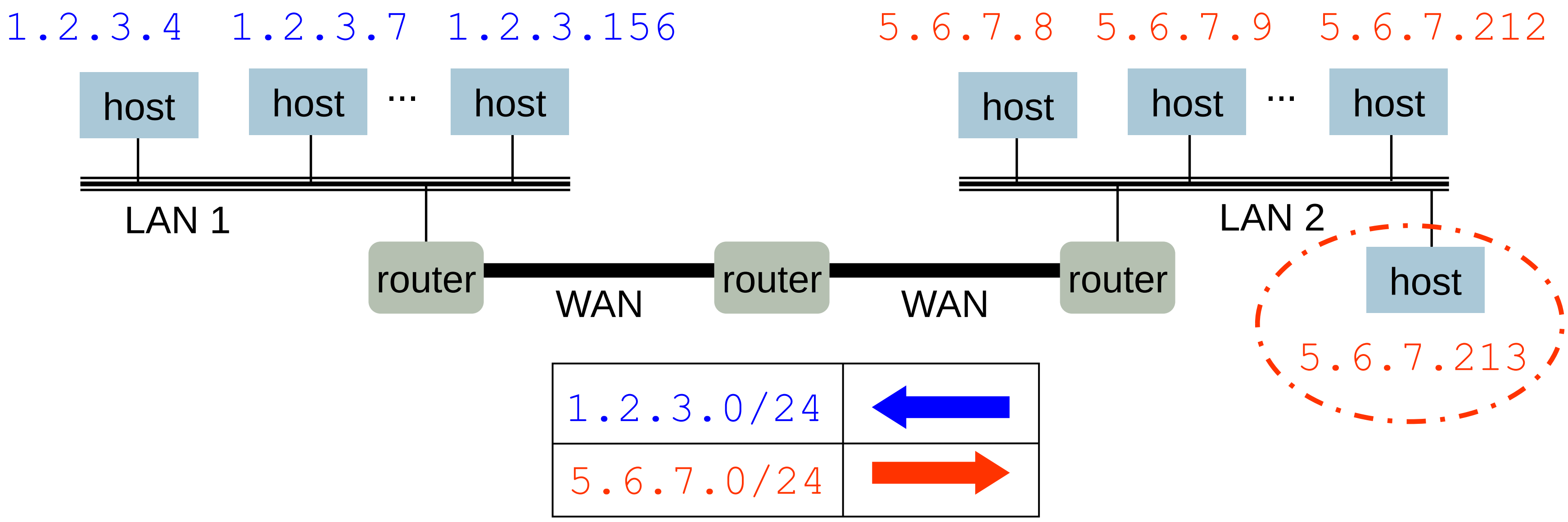


1.2.3.0/24	←
5.6.7.0/24	→

forwarding table

Easy to Add New Hosts

- No need to update the routers
  - E.g., adding a new host 5.6.7.213 on the right
  - Doesn't require adding a new forwarding entry



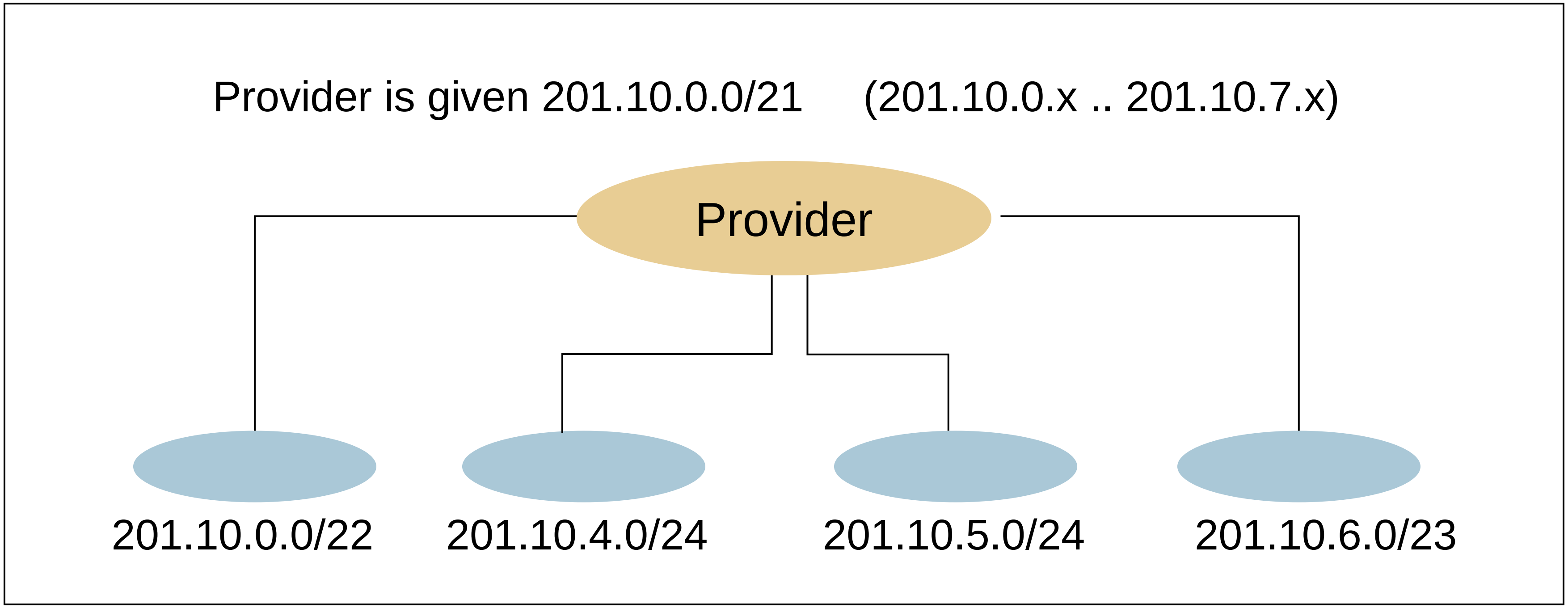
forwarding table

Routers and Forwarding Tables

The End

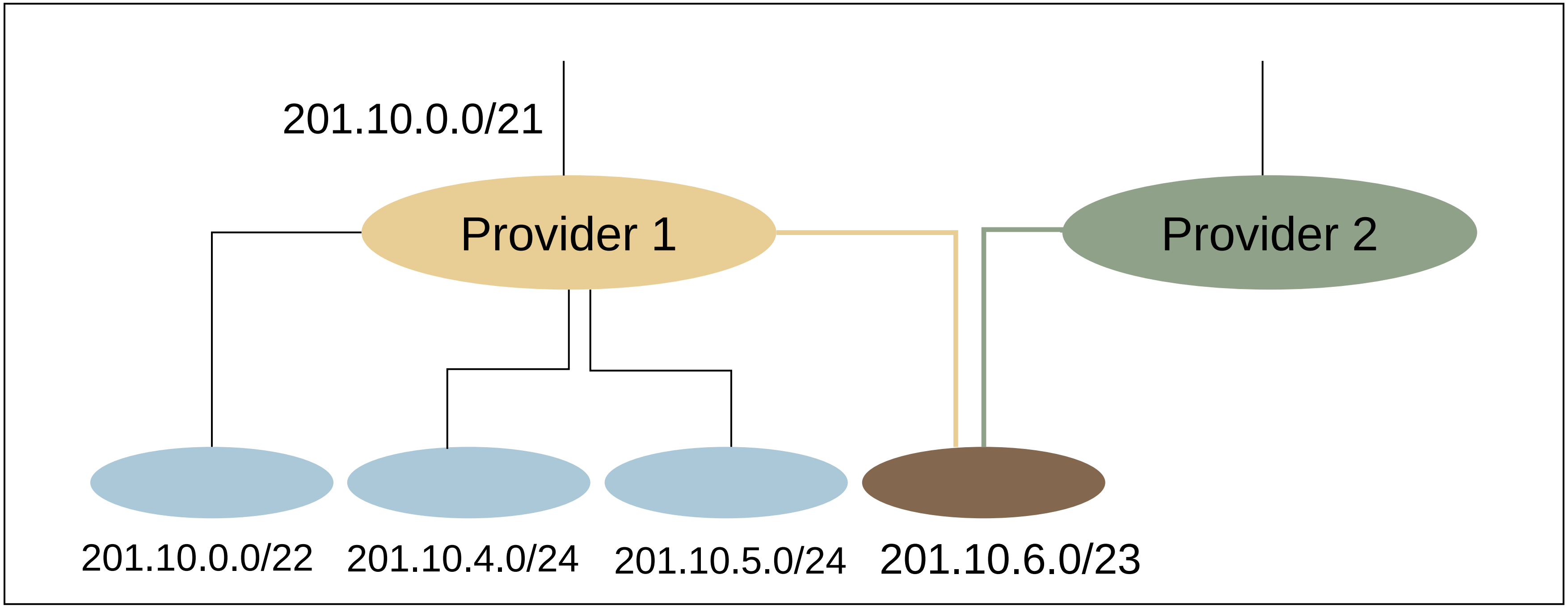
Scalability of
IP Address Allocation

Scalability: Address Aggregation



Routers in the rest of the Internet just need to know how to reach **201.10.0.0/21**. The provider can direct the IP packets to the appropriate ***customer***.

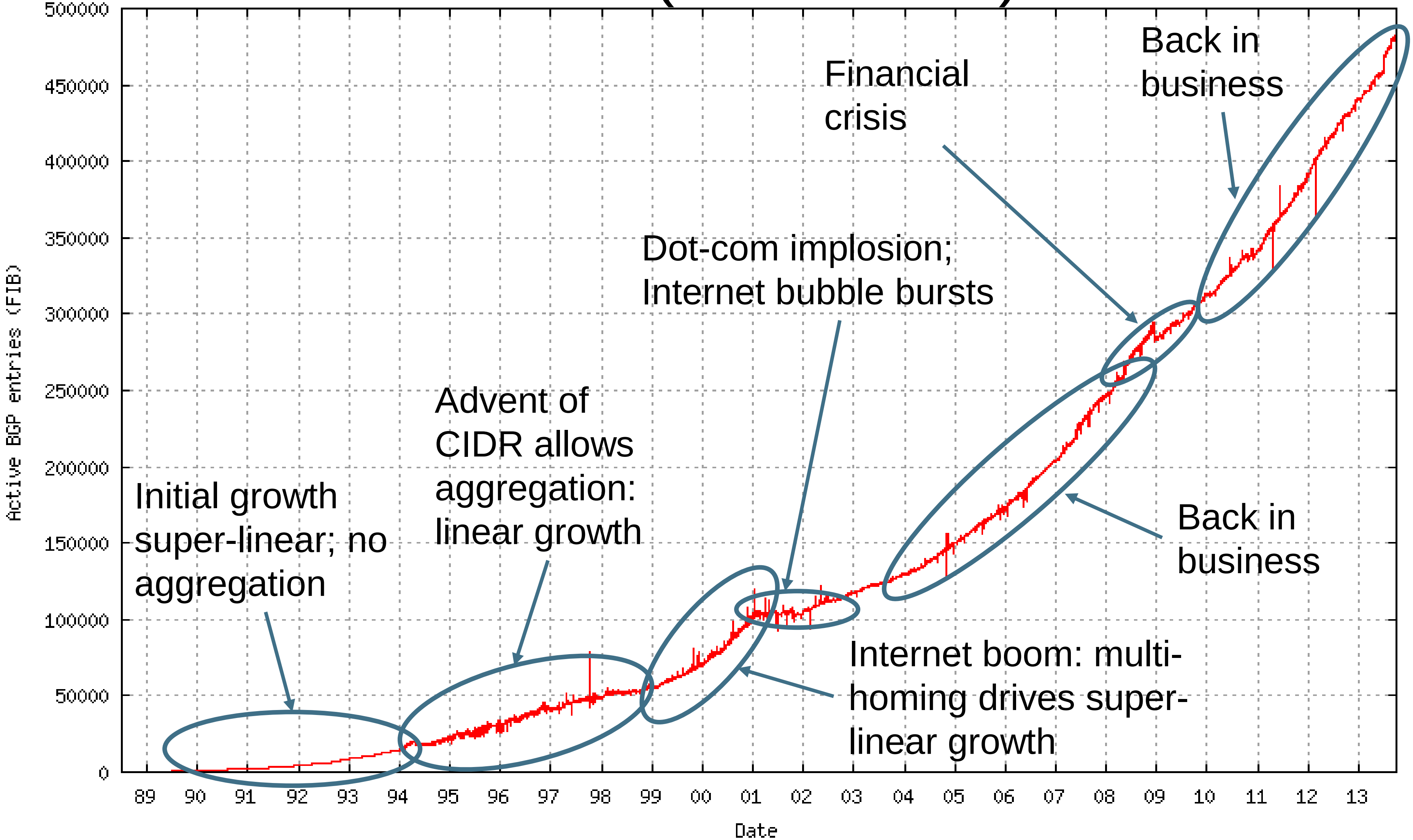
But, Aggregation Not Always Possible



Multi-homed customer with 201.10.6.0/23 has two providers; other parts of the Internet need to know how to reach these destinations through *both* providers

→ /23 route must be globally visible

Growth in Routed Prefixes (1989–2013)



Special Purpose Address Blocks

- Private addresses
  - By agreement, ***not routed*** in the public Internet
  - For networks not meant for general Internet connectivity
  - Blocks: **10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16**
- Link-local
  - By agreement, not forwarded by ***any*** router
  - Used for single-link communication only
  - Intent: auto-configuration (especially when *DHCP* fails)
  - Block: **169.254.0.0/16**

Special Purpose Address Blocks

- Loopback
  - Address blocks that refer to the local machine
  - Block: **127.0.0.0/8**
  - Usually only **127.0.0.1/32** is used
- Limited broadcast
  - Sent to every host attached to the local network
  - Block: **255.255.255.255/32**

Scalability Through Nonuniform Hierarchy

- Summary:
  - **Hierarchical** addressing
    - Critical for ***scalable*** system
    - Don’t require everyone to know everyone else
    - Reduces amount of updating when something changes
  - **Nonuniform** hierarchy
    - Useful for heterogeneous networks of different sizes
    - Initial class-based addressing was far too coarse
    - Classless Inter-Domain Routing (CIDR) gains much more flexibility

Scalability of IP Address Allocation

The End

IP Address Allocation

Obtaining a Block of Addresses

- Separation of control
  - Prefix: assigned *to* an institution
  - Addresses: assigned *by* the institution to their nodes

Obtaining a Block of Addresses

- Who assigns prefixes?
  - Internet Corporation for Assigned Names and Numbers (ICANN)
    - Allocates large address blocks to *regional Internet registries*
    - ***ICANN*** is **politically charged**
  - Regional Internet registries (RIRs)
    - E.g., ***ARIN*** (American Registry for Internet Numbers)
    - Allocates address blocks within their regions
    - Allocated to Internet service providers (ISPs) and large institutions (for money)
  - Internet service providers (ISPs)
    - Allocate address blocks to their customers (could be recursive)
      - Often without charge

Figuring Out Who Owns an Address

- Address registries
  - Public record of address allocations
  - Internet service providers (ISPs) should update when giving addresses to customers
  - However, records are notoriously out-of-date
- Ways to query
  - UNIX: “whois –h whois.arin.net 169.229.60.27”
  - Arin Whois
  - Geektools Whois
  - …

Are 32-Bit Addresses Enough?

- Not all that many unique addresses
  - 2^{32} = 4,294,967,296 (just over four billion)
  - Plus, some (many) reserved for special purposes
  - And, addresses are allocated in larger blocks
- And, many devices need IP addresses
  - Computers, PDAs, routers, tanks, toasters, …
- Long-term solution (**perhaps**): larger address space
  - IPv6 has 128-bit addresses ($2^{128} = 3.403 \times 10^{38}$)
- Short-term solutions: limping along with IPv4
  - Private addresses
  - Network address translation (NAT)
  - Dynamically-assigned addresses (Dynamic Host Configuration Protocol or DHCP)

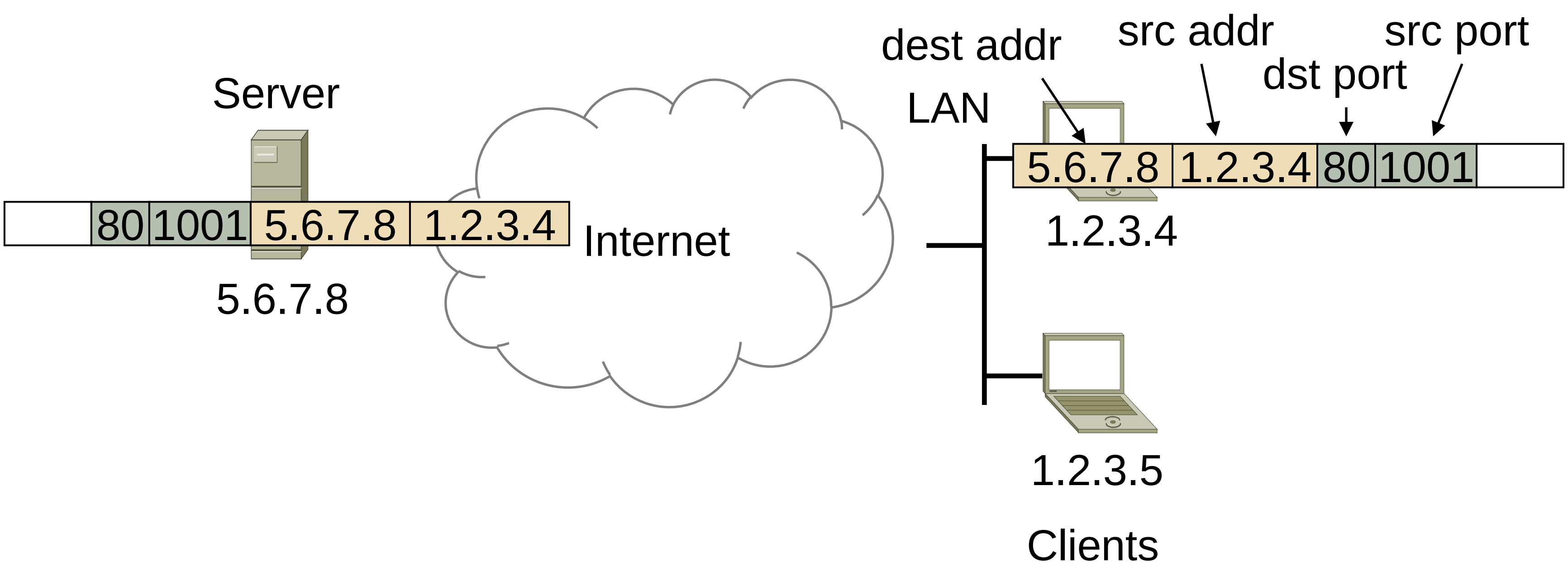
IP Address Allocation

The End

Network Address Translation (NAT)

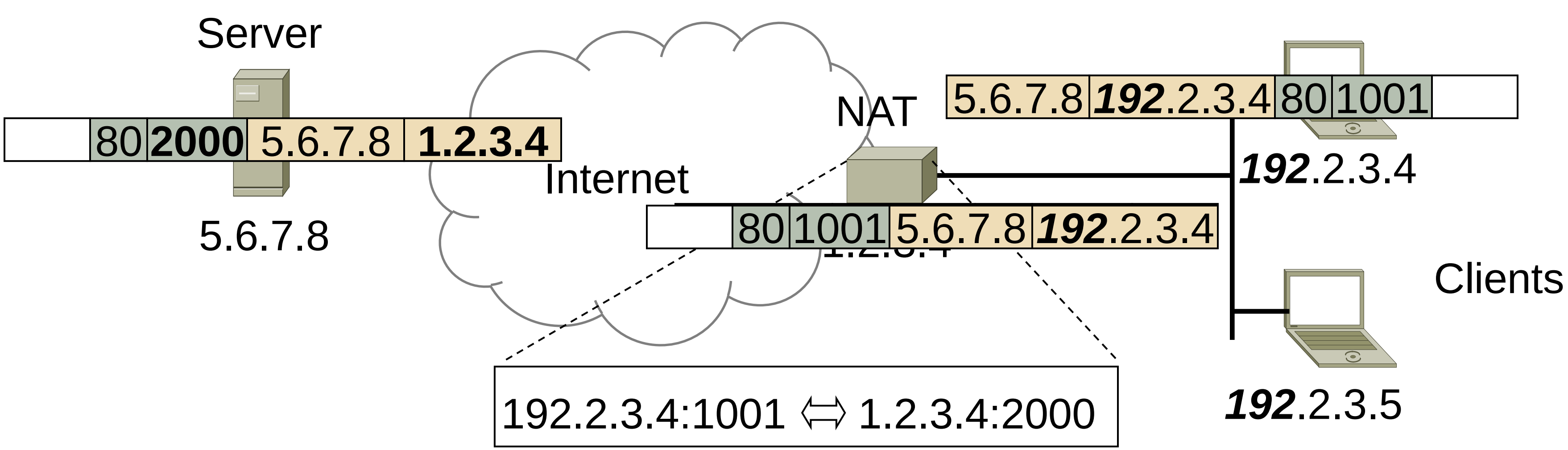
Network Address Translation (NAT)

- Before NAT…
  - Every machine connected to the Internet had a unique IP address



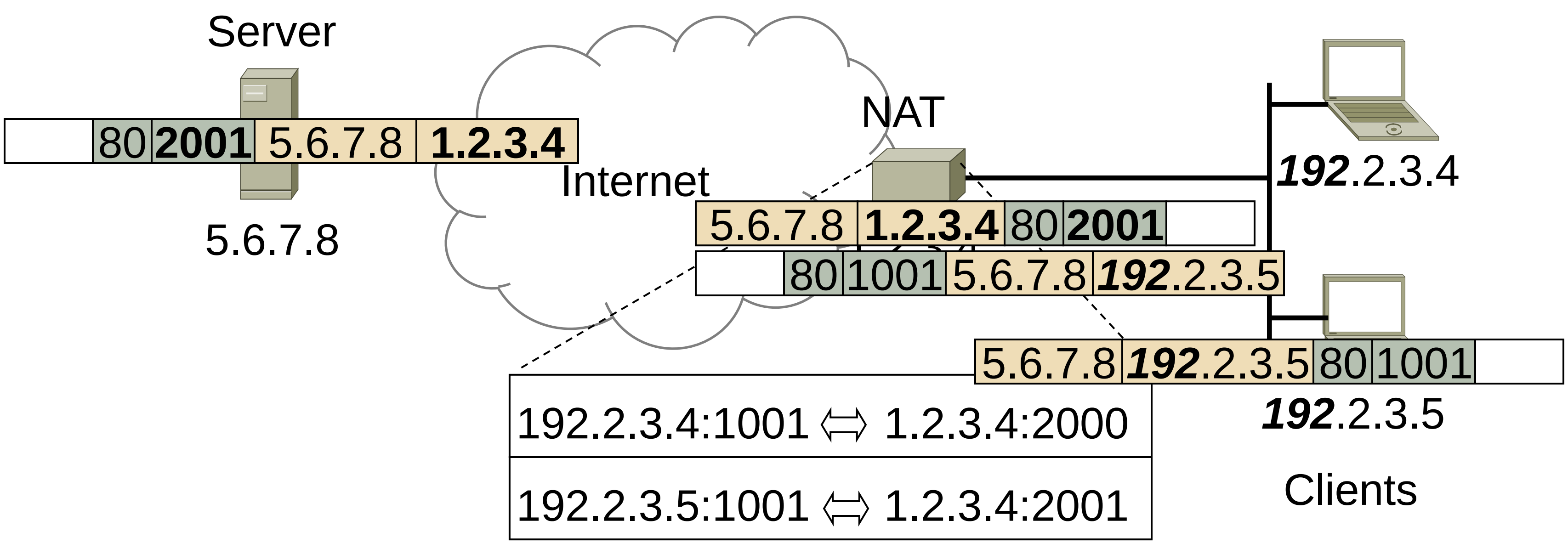
Network Address Translation (cont.)

- Independently assign addresses to machines behind the same NAT
  - Usually in address block **192.168.0.0/16**
- Use port numbers to multiplex/demultiplex internal addresses



Network Address Translation (cont.)

- Independently assign addresses to machines behind the same NAT
  - Usually in address block **192.168.0.0/16**
- Use port numbers to multiplex demultiplex internal addresses



Hard Policy Questions

- How much address space per geographic region?
  - Equal amount per country?
  - Proportional to the population?
  - What about addresses already allocated?
- Address space portability?
  - Keep your address block when you change providers?
  - Pro: avoid having to renumber your equipment
  - Con: reduces the effectiveness of address aggregation
- Keeping the address registries up to date?
  - What about mergers and acquisitions?
  - Delegation of address blocks to customers?
  - As a result, the registries are often out of date

Summary of IP Addressing

- 32-bit numbers identify **interfaces**
- Allocated in prefixes
- **Nonuniform hierarchy** for scalability and flexibility
  - Routing is based on **CIDR**
- A number of special purpose blocks reserved
- Address allocation:
  - ICANN → RIR → ISP → customer network → host
- Issues to be covered later
  - How hosts get their addresses (**DHCP**)
  - How to map from an IP address to a link address (**ARP**)
  - How to map from human readable host/domain names to IP addresses (**DNS**)

Network Address Translation (NAT)

The End