IP Addressing and Forwarding (with some review of IP)

EE122 Fall 2012

Scott Shenker

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, Vern Paxson and other colleagues at Princeton and UC Berkeley

Agenda for Today

- Review of IP:
  - Quick Overview of Fragmentation
  - Review of IPv4 vs IPv6
  - Quick Security Analysis
- IP Addressing and Forwarding
  - to be continued on Thursday

Fragmentation

Why do I care about fragmentation?

- I don't. Not one whit.

- But it is a good exercise in header engineering
  - They could have done this stupidly, but didn't

- And it gives you a chance to show you understand how the various header fields work….
  - This will be on midterm, so **wake up**.

Where Should Reassembly Occur?

Classic case of E2E principle

- Must be done at ends
  - Fragments take different paths
- Imposes burden on network
  - Complicated reassembly algorithm
  - Must hold onto state
- *Little benefit, large cost for network reassembly*

Fragmentation Fields

- **Identifier**: which fragments belong together
- **Flags**:
  - **R**eserved: ignore
  - **D**F: don't fragment
  - **M**F: more fragments coming
- **Offset**: portion of datagram this fragment contains
  - **in 8-byte units**
- What if fragments arrive out of order?
  - Isn't MF meaningless?
  - Doesn't the data get out of order?

Why This Works

- Fragment without MF set (last fragment)
  - Tells host which are the last bits bits in datagram
- All other fragments fill in holes in datagram
- Can tell when holes are filled, regardless of order

Example of Fragmentation

- Suppose we have a 4000 byte datagram sent from host 1.2.3.4 to host 3.4.5.6 …

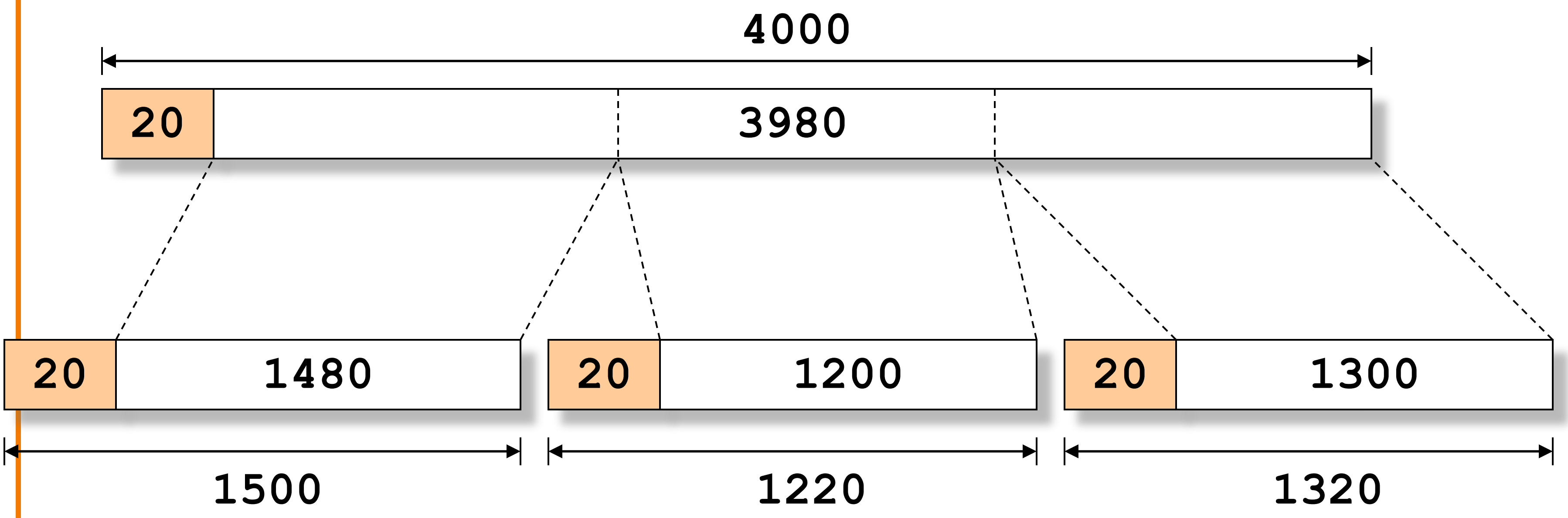
<table>
<tr><td>Version
4</td><td>Header Length
5</td><td>Type of Service
0</td><td colspan="2">Total Length: 4000</td></tr>
<tr><td colspan="3">Identification: 56273</td><td>R/D/M
0/0/0</td><td>Fragment Offset: 0</td></tr>
<tr><td colspan="2">TTL
127</td><td>Protocol
6</td><td colspan="2">Checksum: 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 of payload here)

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

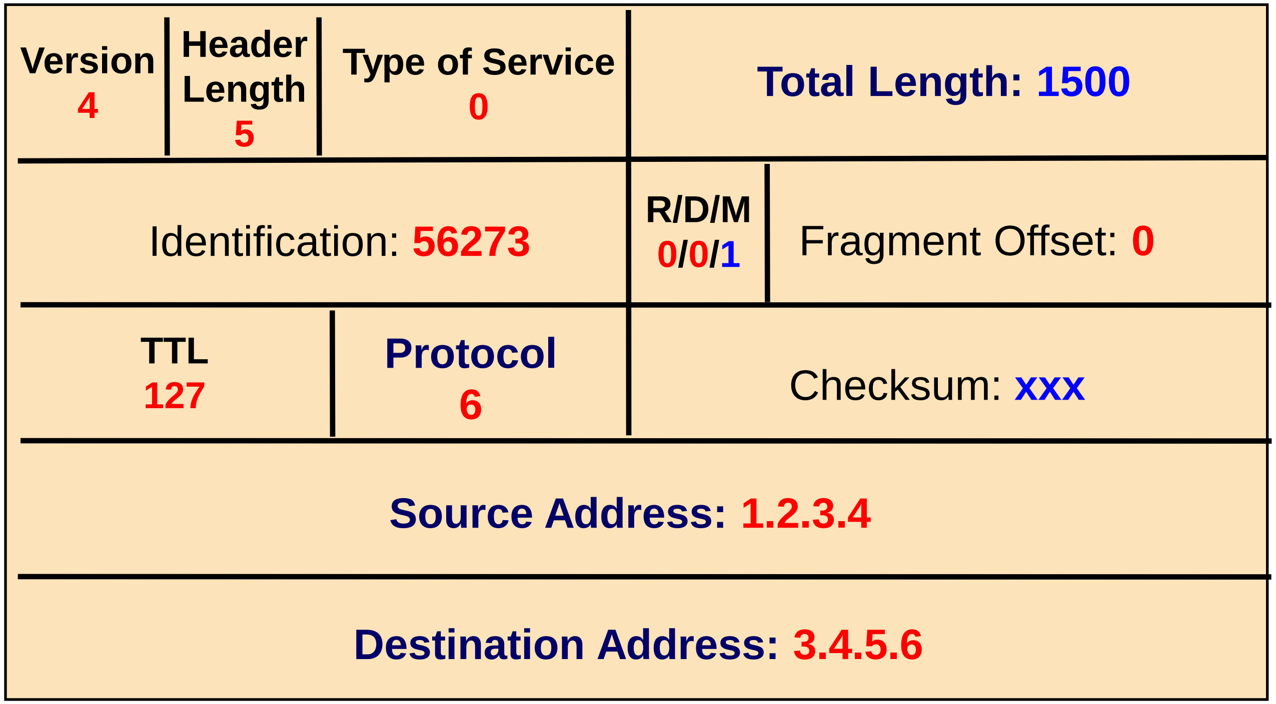
Example of Fragmentation (con’t)

- Datagram split into 3 pieces
- Example:



Example of Fragmentation, con't

- Datagram split into 3 pieces. Possible first piece:



Example of Fragmentation, con’t

- Possible second piece: Frag#1 covered 1480bytes

Version 4	Header Length 5	Type of Service 0	Total Length: 1220
Identification: 56273		R/D/M 0/0/1	Fragment Offset: 185 (185 * 8 = 1480)
TTL 127	Protocol 6	Checksum: yyy	
Source Address: 1.2.3.4			
Destination Address: 3.4.5.6			

Example of Fragmentation, con't

- Possible third piece: 1480+1200 = 2680

Version 4	Header Length 5	Type of Service 0	Total Length: 1320
Identification: 56273		R/D/M 0/0/0	Fragment Offset: 335 (335 * 8 = 2680)
TTL 127	Protocol 6	Checksum: zzz	
Source Address: 1.2.3.4			
Destination Address: 3.4.5.6			

Offsets vs Numbering Fragments?

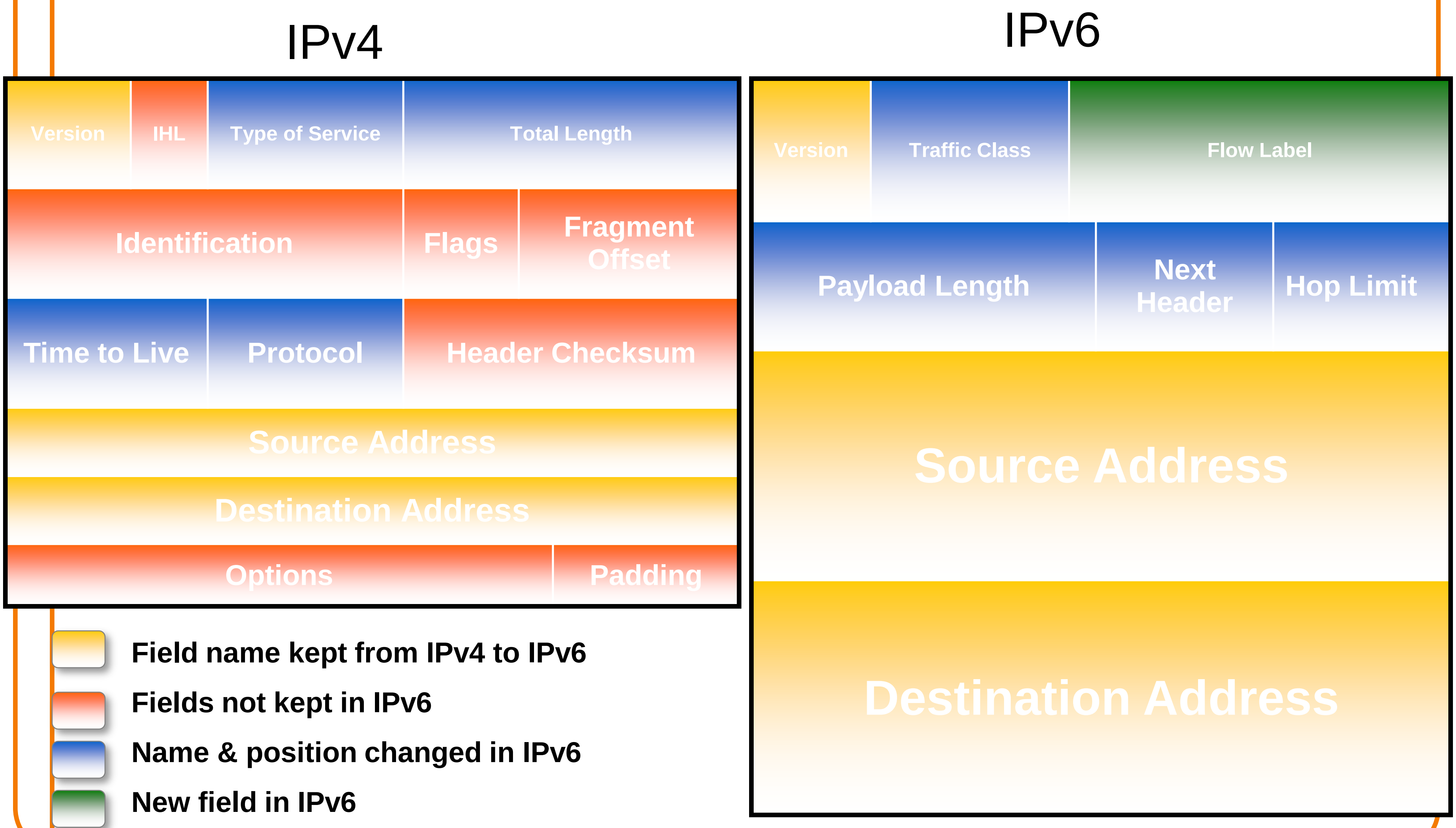
- Q: why use a byte-offset for fragments rather than a numbering each fragment?

- Ans #1: with a byte offset, the receiver can lay down the bytes in memory when they arrive

- Ans #2 *(more fundamental)*: allows further fragmentation of fragments

IPv6

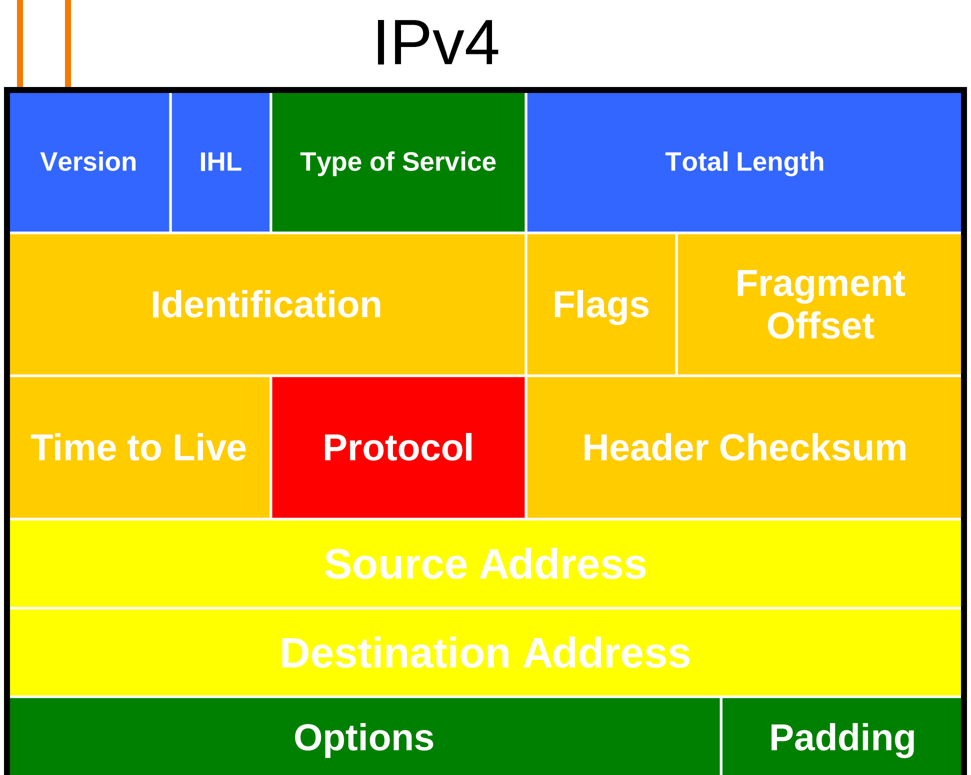
IPv4 and IPv6 Header Comparison

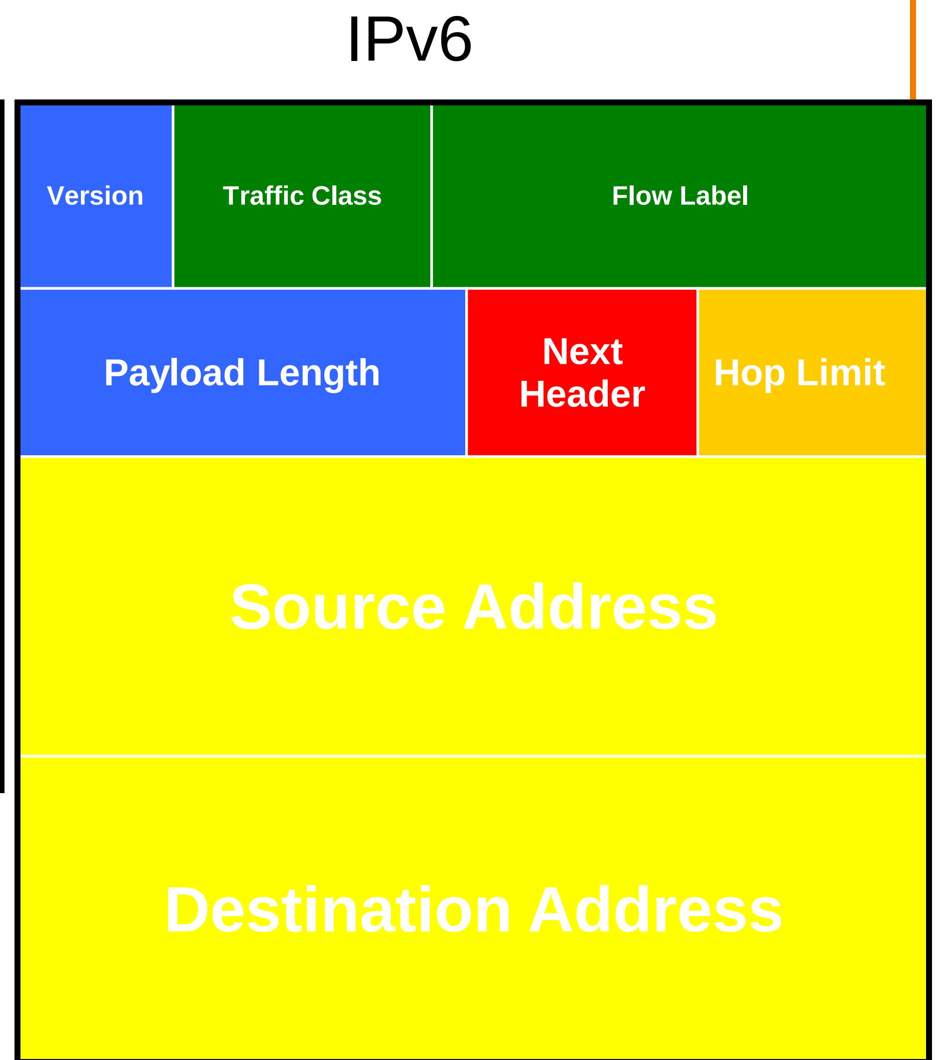


Philosophy of Changes

- Don’t deal with problems: leave to ends
  - Eliminated fragmentation
  - Eliminated checksum
- Simplify handling:
  - New options mechanism (uses next header approach)
  - Eliminated header length
- Provide general flow label for packet
  - Not tied to semantics
  - Provides great flexibility

Comparison of Design Philosophy





- To Destination and Back (expanded)
- Deal with Problems (greatly reduced)
- Read Correctly (reduced)
- Special Handling (similar)

Improving on IPv4 and IPv6?

- Why include unverifiable source address?
  - Would like accountability ***and*** anonymity (now neither)
  - Return address can be communicated at higher layer
- Why packet header used at edge same as core?
  - Edge: host tells network what service it wants
  - Core: packet tells switch how to handle it
    - One is local to host, one is global to network
- Some kind of payment/responsibility field?
  - Who is responsible for paying for packet delivery?
  - Source, destination, other?
- Other ideas?

Quick Security Analysis of IP Packet Header

More for mindset than content

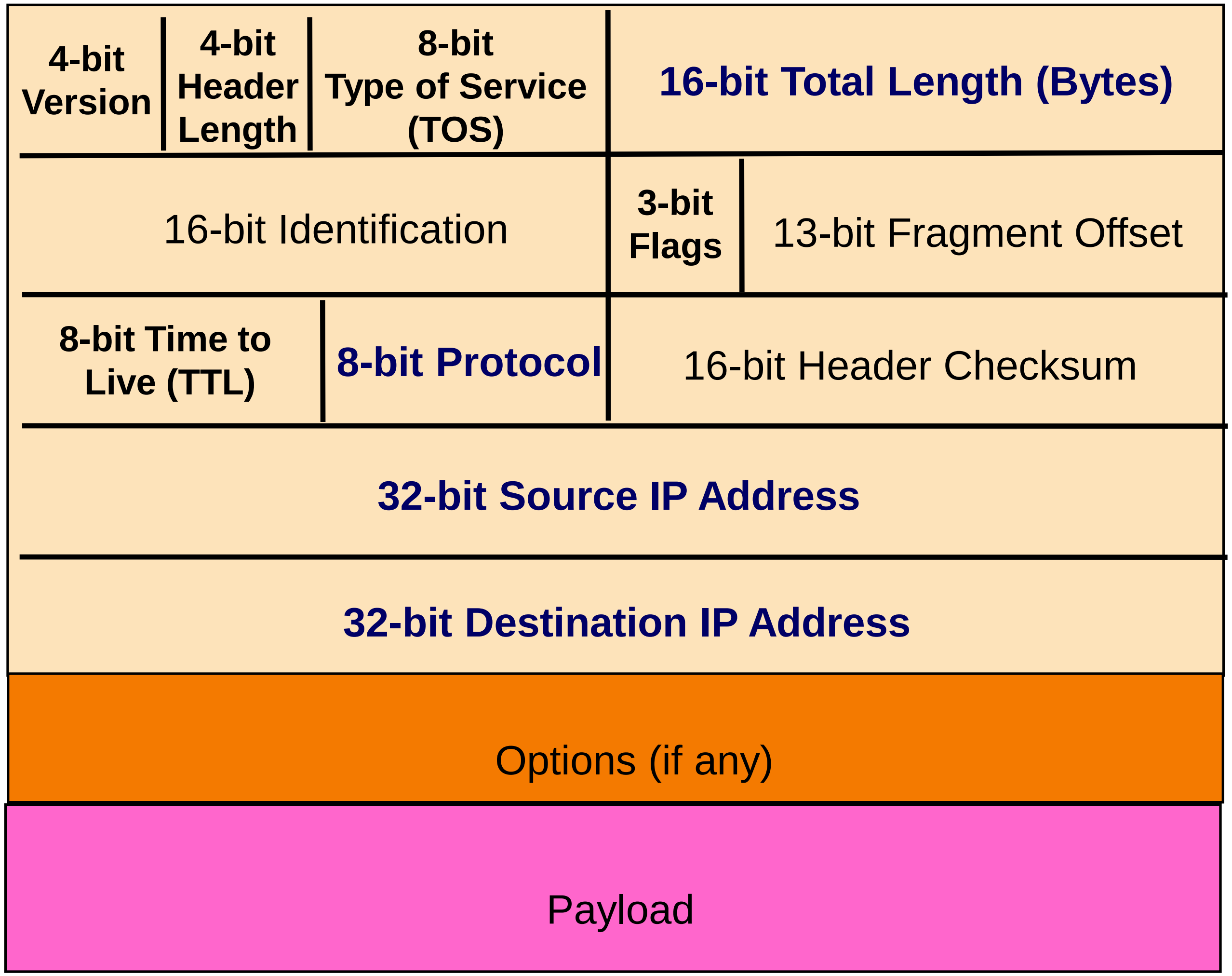
The workings of a paranoid mind…..

Focus on Sender Attacks

- Ignore (for now) attacks by others:
  - Traffic analysis
  - Snooping payload
  - Denial of service

- Focus mostly on vulnerabilities sender can exploit

IP Packet Structure



IP Address Integrity

- Source address should be the sending host
  - But, who’s checking?
  - You could send packets with any source you want
  - ***Why is checking hard?***

Implications of IP Address Integrity

- Why would someone use a bogus source address?
- Launch a **denial-of-service** attack
  - Send excessive packets to the destination
  - … to overload the node, or the links leading to the node
  - But: victim can identify/filter you by the source address
- Evade detection by "spoofing"
  - Put **someone else's** source address in the packets
    - **Or**: use many **different** ones so can't be filtered
- Or: as a way to bother the spoofed host
  - Spoofed host is wrongly blamed
  - Spoofed host may receive return traffic from the receiver

More Security Implications

- Version field (4 bits) …. ?
  - Issue: fledgling **IPv6** deployment means sometimes connectivity exceeds security enforcement
  - E.g., firewall rules only set up for **IPv4**

- Header length (4 bits) …. ?
  - Controls presence of IP **options**
    - E.g., **Source Route** lets sender control path taken through network - say, sidestep security monitoring
  - IP options often processed in router's slow path
    - Allows attacker to stress router for denial-of-service
  - Firewalls often configured to **drop** packets with options.

Security Implications of TOS? (8 bits)

- Attacker sets TOS priority for their traffic?
  - If regular traffic does not set TOS, then network prefers the attack traffic, greatly increasing damage

- What if network charges for TOS traffic …
  - … and attacker spoofs the victim's source address?

- Today, network TOS generally **does not work**
  - Due to very hard problems with billing
  - TOS has now been redefined for *Differentiated Service*
    - Discussed later in course

Security Implications of Fragmentation?

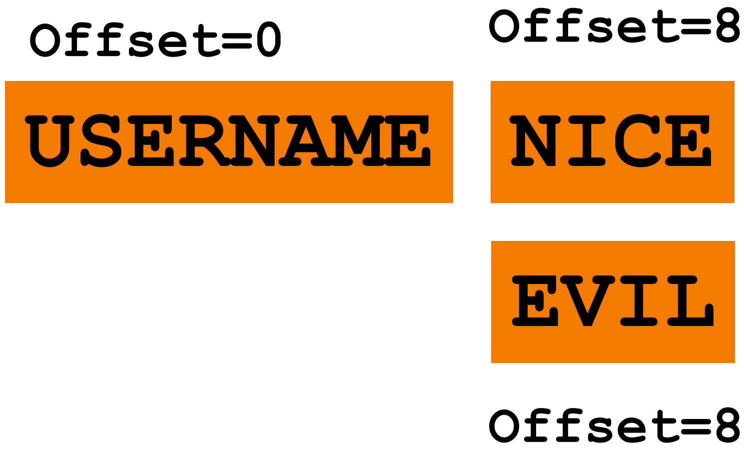
- Allows **evasion** of network monitoring/enforcement
- E.g., split an attack across multiple fragments
  - Packet inspection won't match a "signature"

`Offset=0`	`Offset=8`
`Nasty-at`	`tack-bytes`

- Can be addressed by monitor remembering previous fragments
  - But that costs **state**, which is another vector of attack

More Fragmentation Attacks

- What if 2 overlapping fragments are inconsistent?



- How does network monitor know whether receiver sees **USERNAME NICE** or **USERNAME EVIL**?

Even More Fragmentation Attacks

- What if fragments exceed IP datagram limit?

```
Offset=65528
```

`NineBytes`

  - Maximum size of 13-bit field: 0x1FFF = 8191
    Byte offset into final datagram = 8191*8 = 65528
    Length of final datagram = 65528 + 9 = **65537**

- Result: **kernel crash**
  - Denial-of-service using just a few packets
  - Fixed in modern OS's

Even Even More Fragmentation Attacks

- What happens if attacker doesn't send all of the fragments in a datagram?
- Receiver (or firewall) winds up holding the ones they receive for a long time
  - **State-holding** attack

Security Implications of TTL? (8 bits)

- Allows discovery of **topology** (a la *traceroute*)
- Can provide a hint that a packet is spoofed
  - It arrives at a router w/ a TTL different than packets from that address usually have
    - Because path from attacker to router has different # hops
  - Though this is *brittle* in the presence of routing changes
- Initial value is somewhat distinctive to sender's operating system. This plus other such initializations allow OS **fingerprinting** …
  - Which allow attacker to infer its likely vulnerabilities

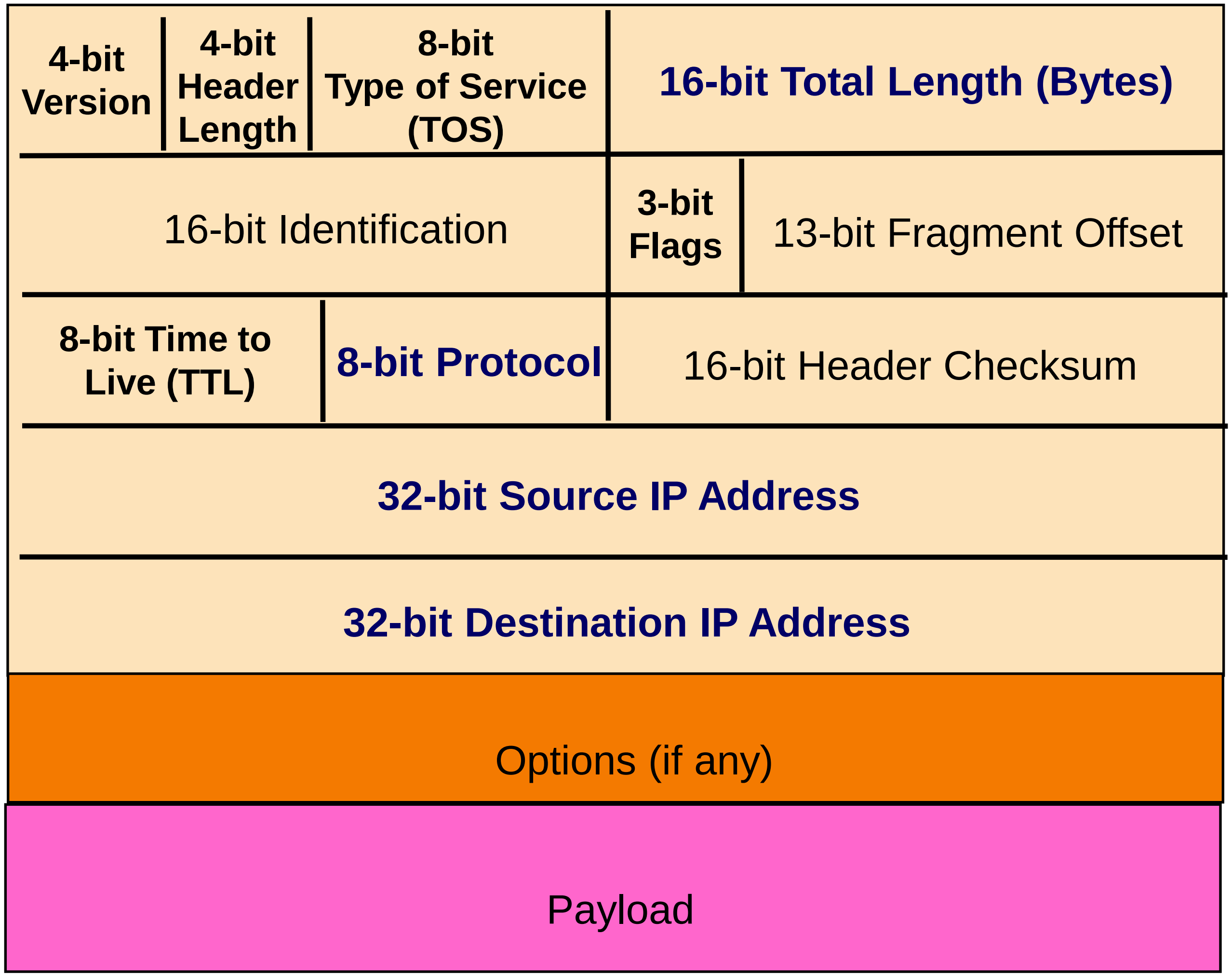
Security Implications of Remainder?

- No apparent problems with protocol field (8 bits)
  - It's just a demux'ing handle
  - If set incorrectly, next layer will find packet ill-formed
- Bad IP **checksum** field (16 bits) will cause packet to be discarded by the network
  - Not an effective attack…

IP Addressing

Basics of Addressing

Have covered everything but addresses!



Use of Addresses

1. Used by routers to forward packets to destination
2. Very poor identifier (forget about this use for now)

Focus on use in forwarding

Forwarding vs Routing

- Routing: "**control plane**"
  - Computing paths the packets will follow
  - Distributed protocol leads to state at each router
- Forwarding: "**data plane**"
  - Directing a data packet to an outgoing link
  - Individual router using routing state
- Two very different timescales….
  - Forwarding: single packet transmission times: μs
  - Routing: can be seconds

Designing an Addressing Scheme

- Must support very fast forwarding
  - Relatively simple lookup
  - Relatively small routing tables
- Routing state must be scalably computable
  - Cannot involve massive exchanges of state

Current IP Addressing

- Reflects series of necessary hacks
  - Necessary to survive, but not pretty…
- No one would design such a system from scratch
- Simple to design a much better scheme
  - Which you will do next lecture!

Layer 2 Addressing

- Typically uses MAC addresses
- Unique numbers burned into interface cards
  - Random string of bits
  - No location information
- Local area networks route on these “flat” addresses

Why can’t we use this approach for IP?

Layer 2 is Local, but Layer 3 is Global!

- Would have entry for every device in the world
  - Must keep track of their location individually
  - Update table whenever they moved!
- Leads to large routing tables ($\sim 10^8$)
- Leads to unscalable routing algorithms
  - Global messages whenever laptop moves

Addressing Goal: ***Scalable*** Routing

- State: Limited amount of routing state (i.e., table)
  - Much less than the number of hosts
- Churn: Limited rate of change in routing tables
  - Traffic, inconsistencies, complexity

Aggregation crucial for both

(use single entry to cover many addresses)

Aggregation only works if….

- Groups of addresses require same forwarding
- These groups are contiguous in address space
- These groups are relatively stable
- Few enough groups to make forwarding easy

Why Is Aggregation Nontrivial?

- Mobility: laptops, cellphones, etc.
- Multihoming: Many entities have two or more ISPs
- Institutional renumbering hard

5 Minute Break

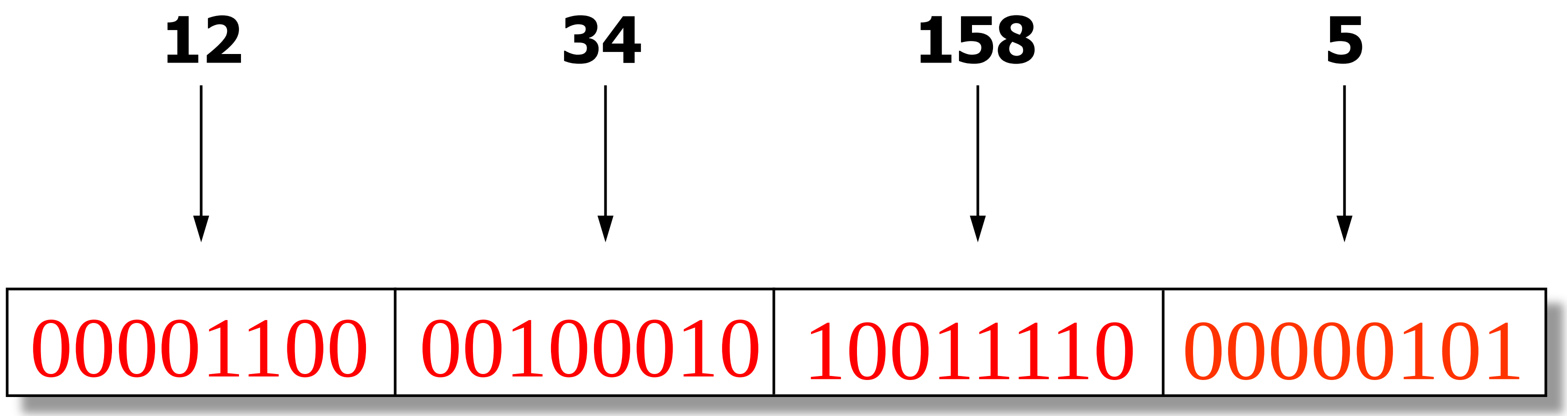
Basic Design

Design Questions

- ***What*** should an address be associated with?
  - *Telephone network is an ambiguous model*
  - Landlines: number refers to location (hard to move)
  - Cell phones: number refers to handset (easily movable)
- What **structure** should addresses have? What are the implications of that structure?
- ***Who*** determines who gets which addresses in the global Internet? What are the implications of how this is done?

IP Addresses (IPv4)

- Unique 32-bit number associated with an ***interface***
  - on a host, on a router, … connect to ports, links, etc.
  - Association can be long-term or short-term
- Use ***dotted-quad*** notation, e.g., **12.34.158.5**:



Examples

- What address is this? **80.19.240.51**

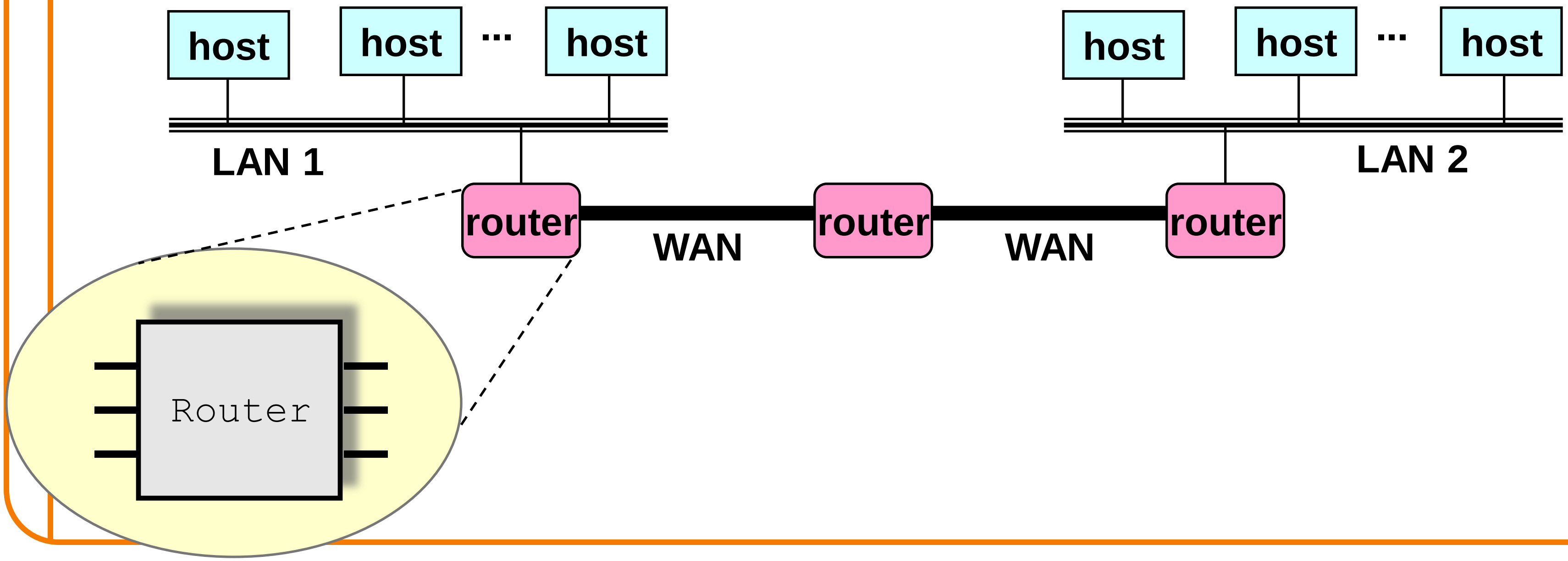
01010000	00010011	11110000	00110011

- How would you represent 68.115.183.7?

01000100	01110011	10110111	00000111

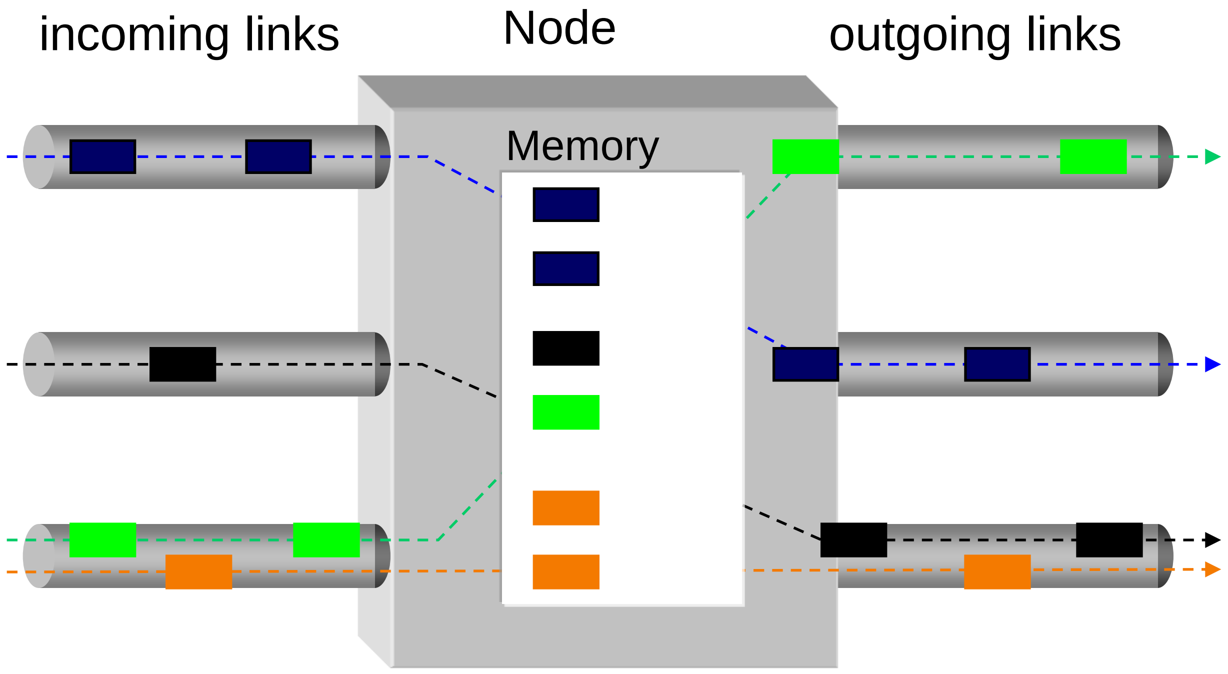
Routers in the Network

- Routers connect links and networks together
- Must forward packets towards destination



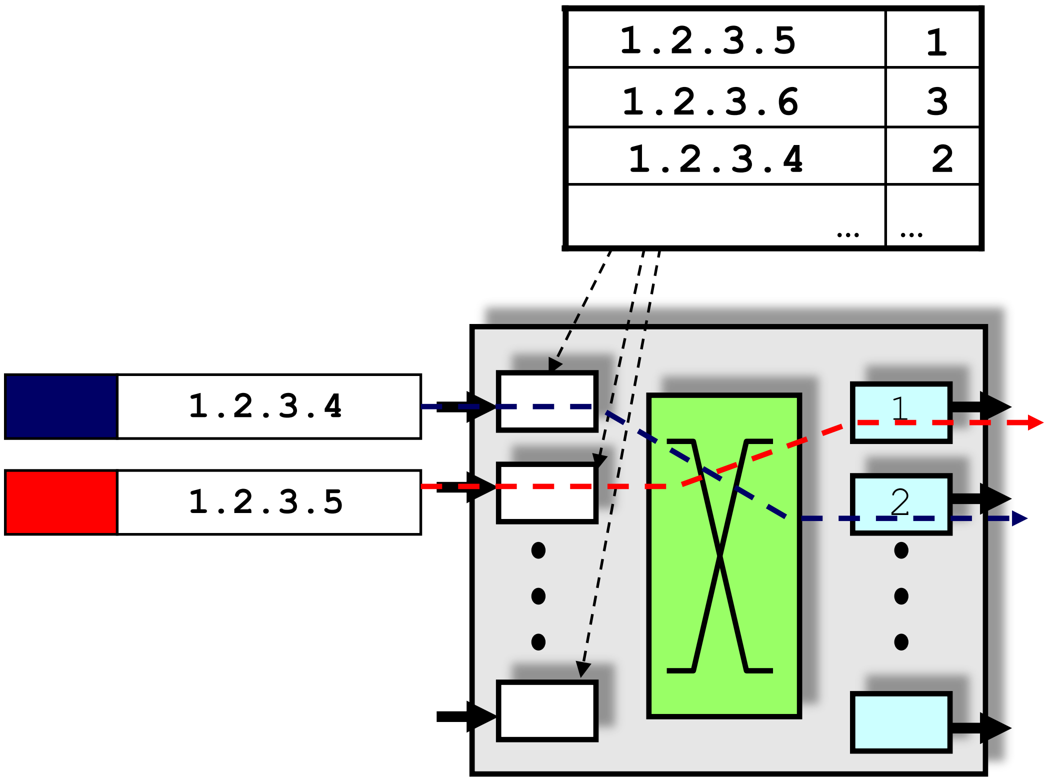
Routers Send Packets to Correct Port

Location of packet queues depends on switch design



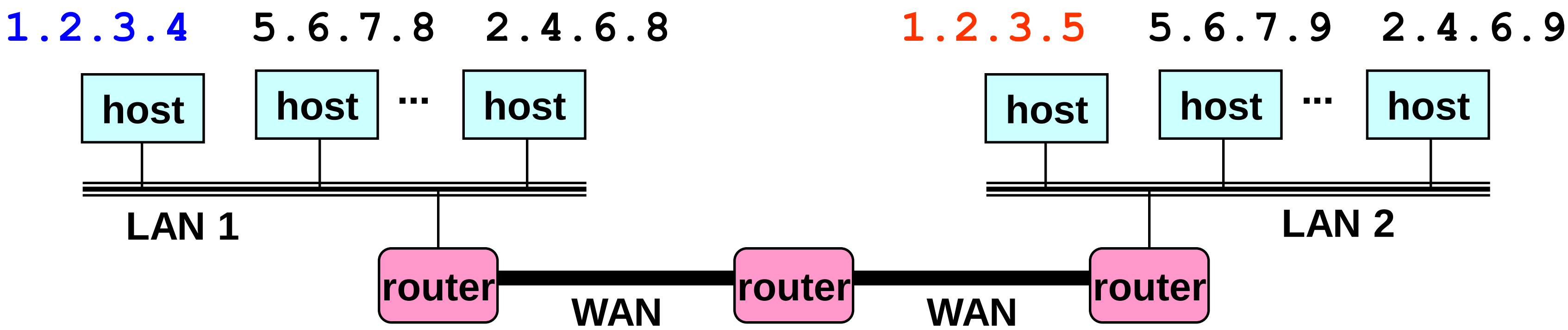
Forwarding Table Plays Crucial Role

- Table maps IP addresses into output interfaces
- Forwards packets based on destination address



Scalability Challenge

- Suppose hosts have random addresses
  - Then routers would need a separate entry for each host
  - Far too much state to hold in each router



1.2.3.4	←
1.2.3.5	→
⋮	

forwarding table

Two Universal Tricks in CS

- When you need more flexibility, you add…
  - *A layer of indirection*
- When you need more scalability, you impose…
  - *A hierarchical structure*

Hierarchical Addressing in U.S. Mail

- Addressing in the U.S. mail
  - Zip code: 94704
  - Street: Center Street
  - Building on street: 1947
  - Location in building: Suite 600
  - Name of occupant: Scott Shenker

- Forwarding the U.S. mail
  - Deliver letter to the post office in the zip code
  - Assign letter to mailman covering the street
  - Drop letter into mailbox for the building/room
  - Give letter to the appropriate person

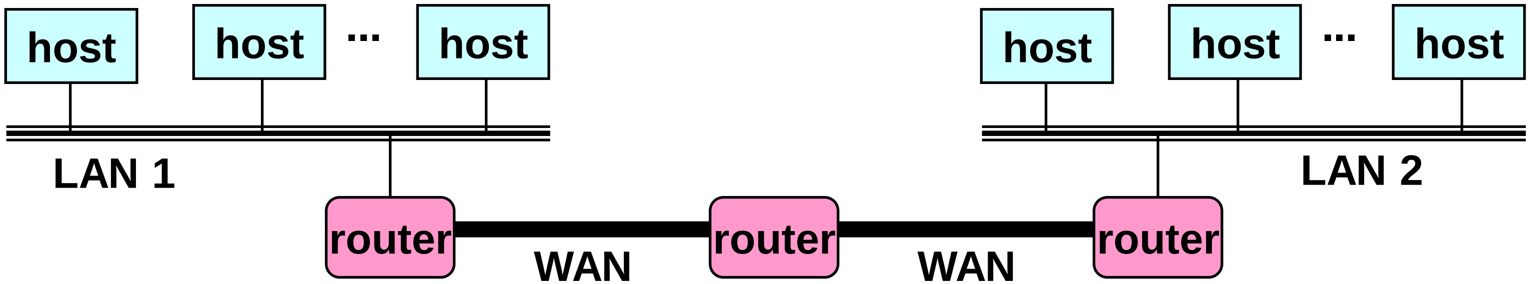
Who Knows What?

- Does anyone in the US Mail system know where every house is?

- Separate routing tables at each level of hierarchy
  - Each of manageable scale

Hierarchical Structure

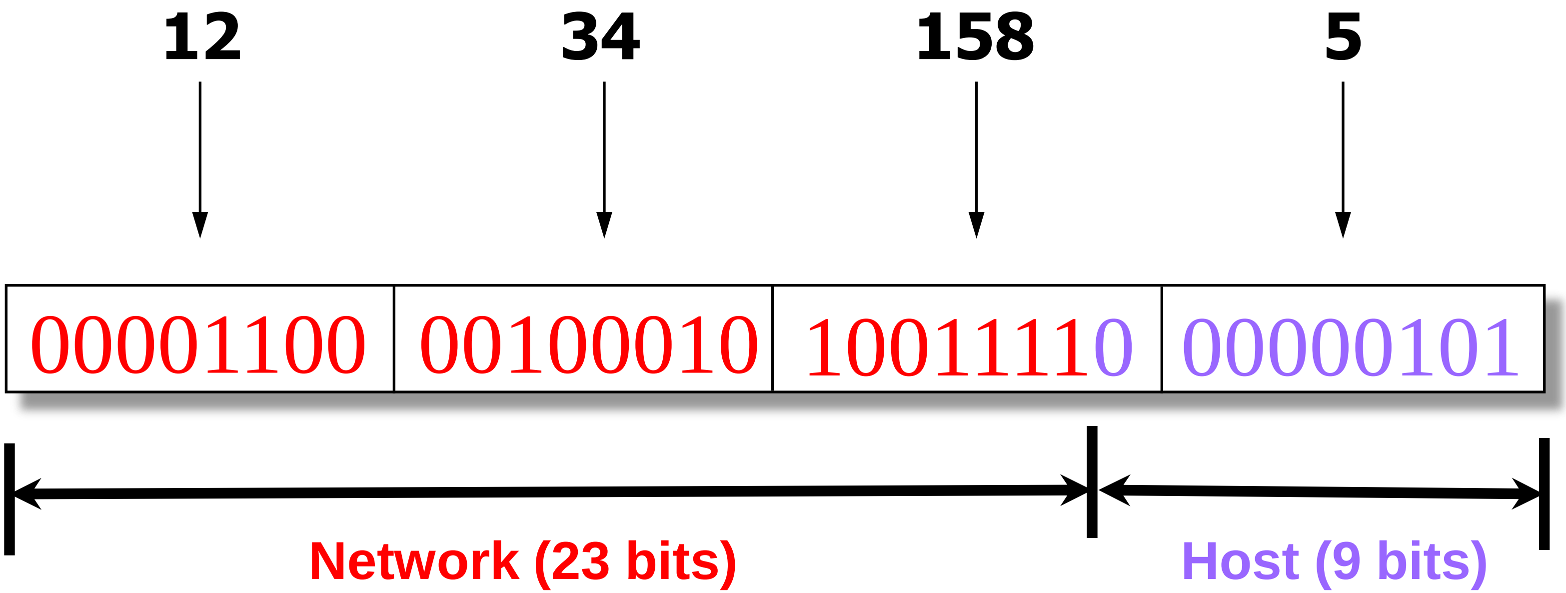
- The Internet is an “inter-network”
  - Used to connect *networks* together, not *hosts*
- Forms a natural two-level hierarchy:
  - WAN delivers to the right LAN *(i.e., deliver to zip code)*
  - LAN delivers to the right host *(i.e., deliver to house)*



LAN = Local Area Network
WAN = Wide Area Network

Hierarchical Addressing

- Prefix is ***network address***: suffix is ***host address***
- 12.34.158.0/23 is a 23-bit prefix with 2^9 addresses
  - **Terminology:** "Slash 23"



IP Address and a 23-bit Subnet Mask

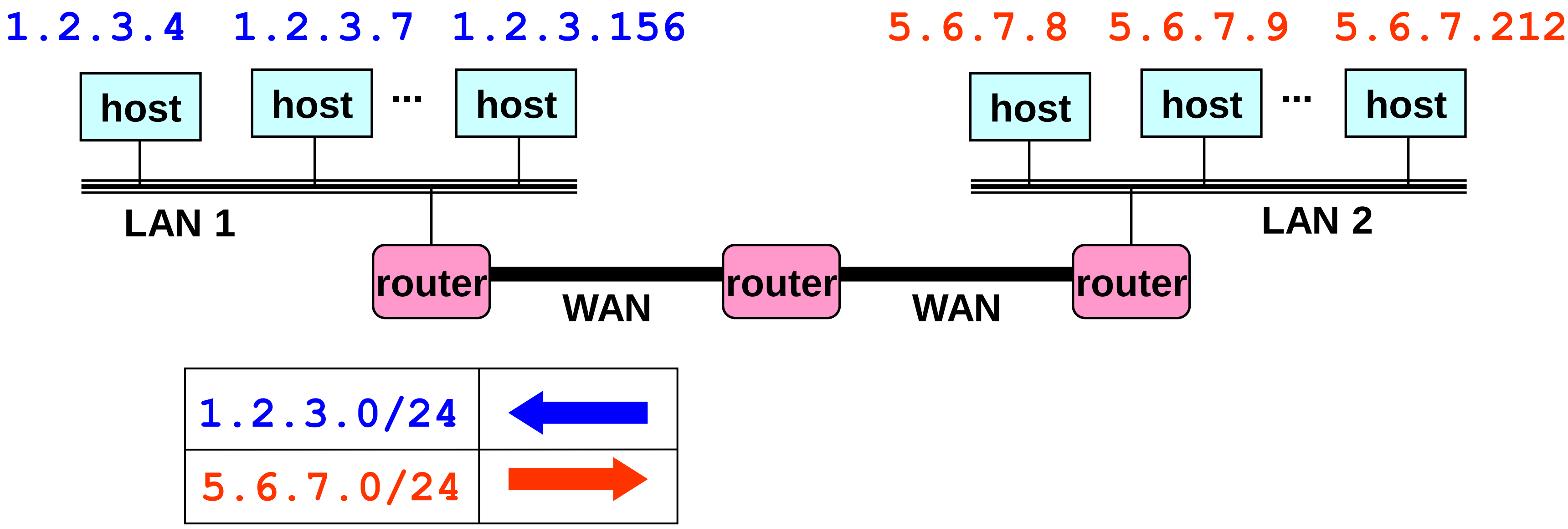
Address

12	34	158	5
00001100	00100010	10011110	00000101
11111111	11111111	11111110	00000000
255	255	254	0

Mask

Scalability Improved

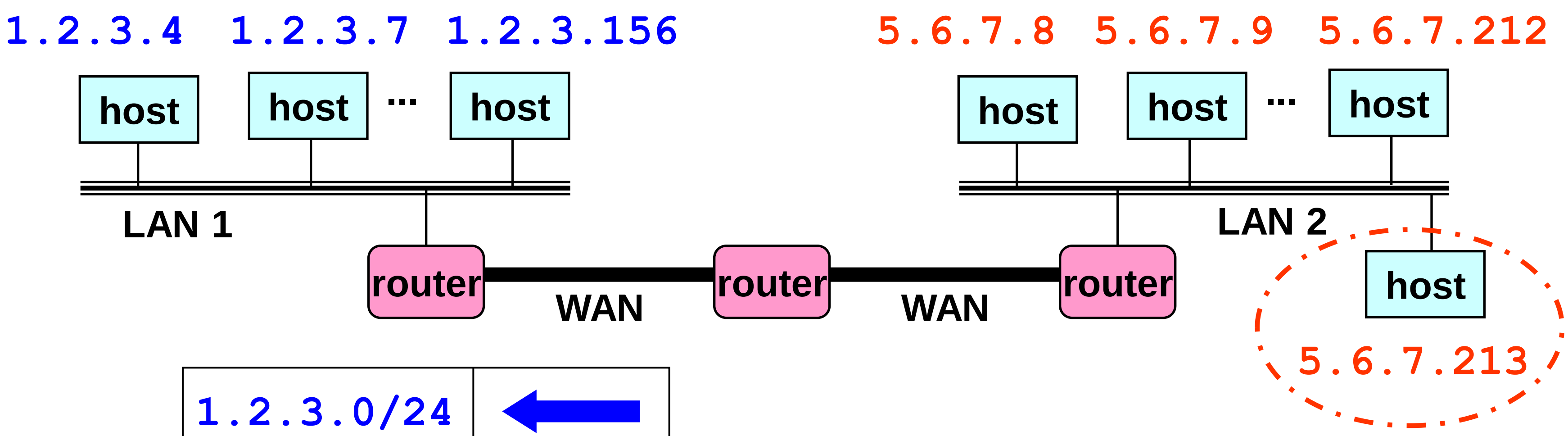
- Number nearby hosts with same prefix
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN



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



1.2.3.0/24	←
5.6.7.0/24	→

forwarding table

“Subnet” Terminology

- Think of LANs as special case of “subnets”
  - Subnet is region without routers containing addresses within the “subnet mask”
  - Could be a link, or LAN
- Textbook has an operational definition of subnet
  - Remove all interfaces from hosts, routers
  - The regions that remain connected are subnets
- Subnets are the lowest level of aggregation
  - No routers needed within a subnet

History of Internet Addressing

- Always dotted-quad notation
- Always network/host address split (subnets)
- But nature of that split has changed over time

Original Internet Addresses

- First eight bits: network address (/8)
- Last 24 bits: host address

Assumed 256 networks were more than enough!

Nice Features

- Transit routers looked at what portion of address?
  - ***Network***
- That portion of address space was flat
  - No need for hierarchy with 256 entries
- Rest of address only relevant on host’s network
- But did not provide for enough networks
  - Ubiquity of ethernet not foreseen

Next Design: Classful Addressing

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

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

- o 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******	********	********	********

- o Large blocks (e.g,. UCB has 128.32.0.0/16)

– Class C: [192..223] ⇒ assume /24 **(top bits = 110)**

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

- o Small blocks (e.g., ICIR has 192.150.187.0/24)
- o (My house used to have a /25)

Classful Addressing (cont'd)

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

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

o Multicast groups

– Class E: [240..255] **(top bits 11110)**

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

o Reserved for future use

- What problems can classful addressing lead to?
  – Only comes in 3 sizes
  – Routers can end up knowing about ***many*** class C's (/24s)
  – Wasted address space

Today's Addressing: CIDR

- CIDR = Classless Interdomain Routing
- Flexible division between network and host addresses
- ***Must specify both address and mask***
  - Clarifies where boundary between addresses lies
  - Classful addressing communicate this with first few bits
  - CIDR requires explicit mask

CIDR Addressing

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

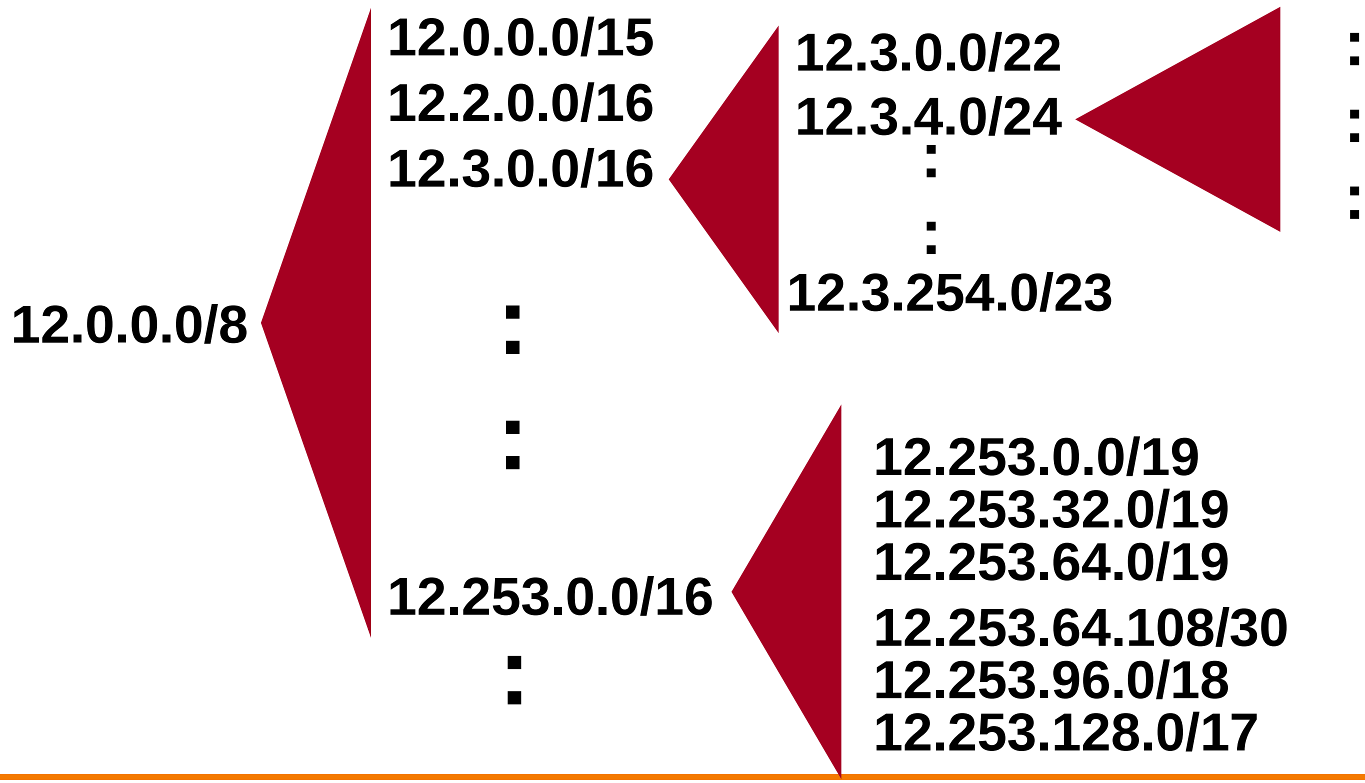
Address	00001100	00000100	00000000	00000000
Mask	11111111	11111110	00000000	00000000

|← Network Prefix →|← for hosts →|

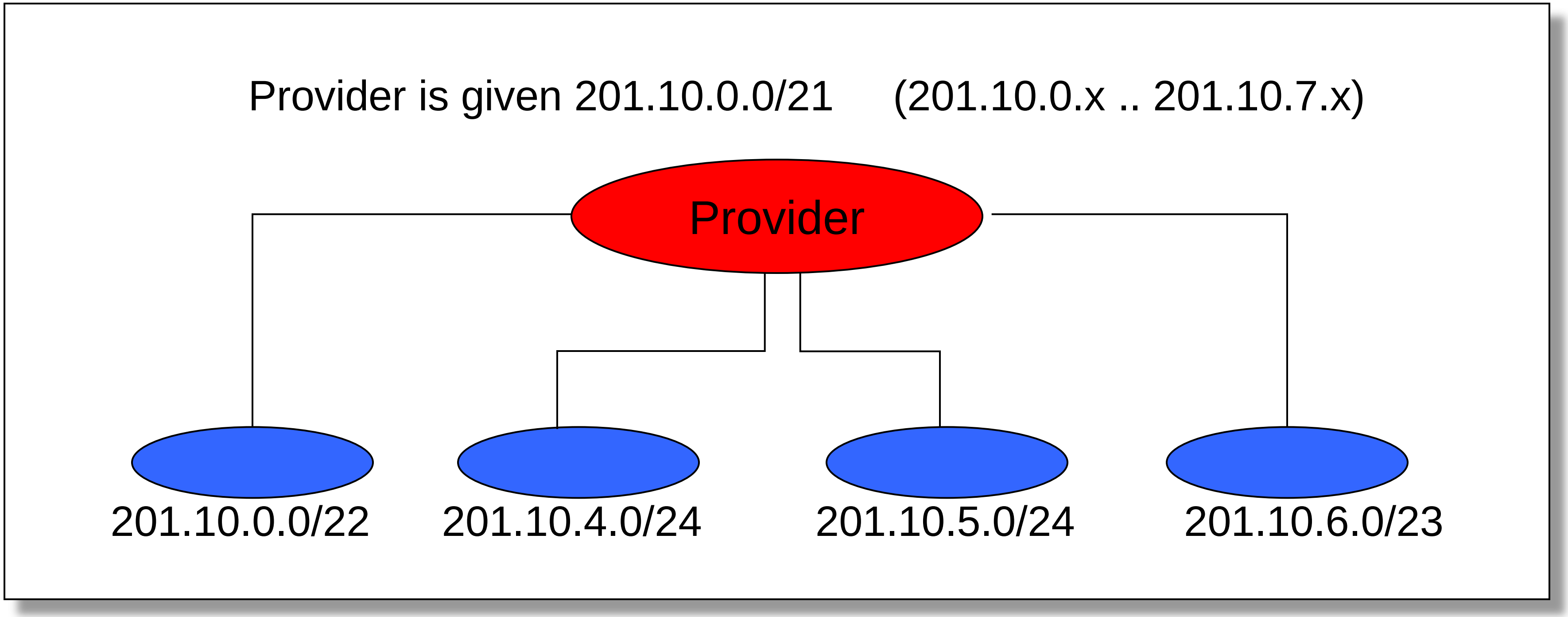
Written as 12.4.0.0/15 or 12.4/15

CIDR: Hierarchal Address Allocation

- Prefixes are key to Internet scalability
  - Addresses allocated in contiguous chunks (prefixes)
  - Routing protocols and packet forwarding based on prefixes
  - Recursively break down chunks as get closer to host

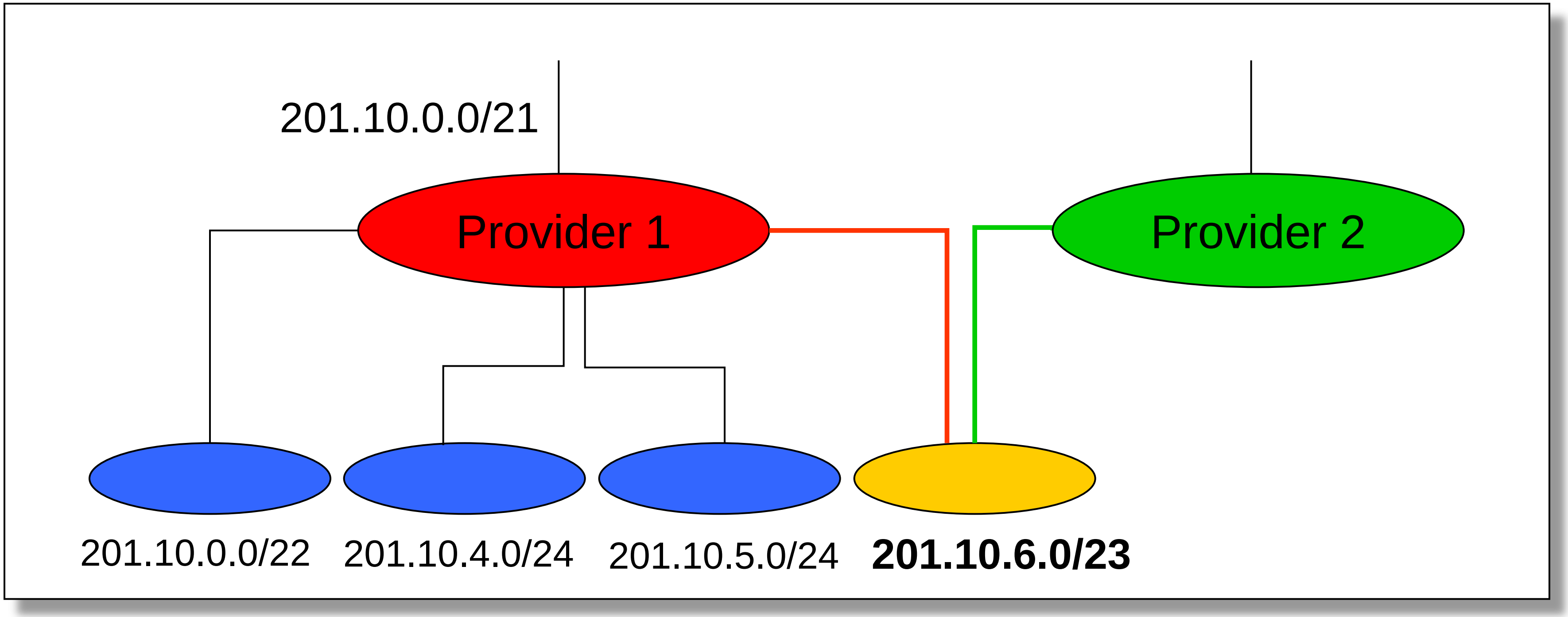


Scalability: Address Aggregation



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.

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.**

$\Rightarrow$ /23 route must be globally visible

Summary

- Fragmentation is a pain, but you have to know it
- IP header can used for various attacks
- Addressing is easy if you don't need to aggregate
  - But we do, and therein lies all the fun
- Next time: