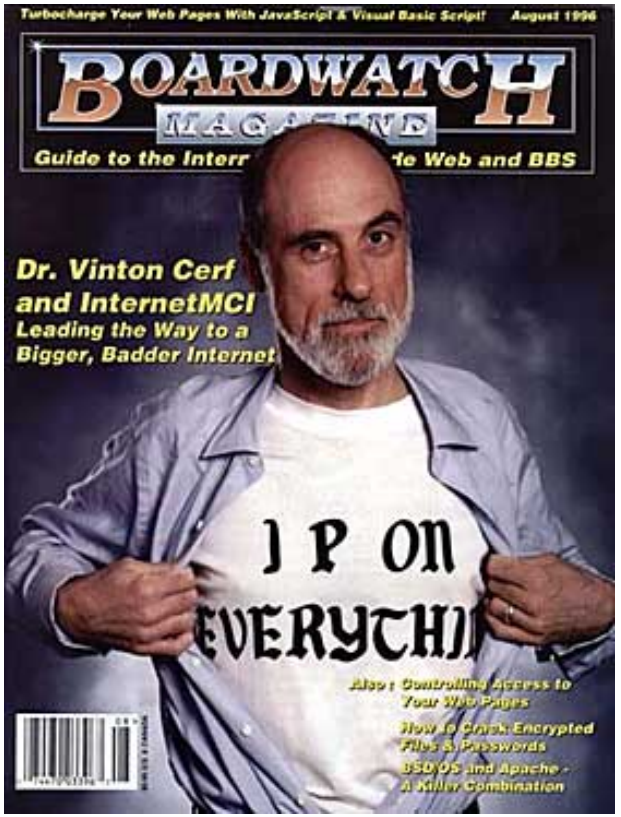
Network layer: Functionality, IP Addressing & Forwarding

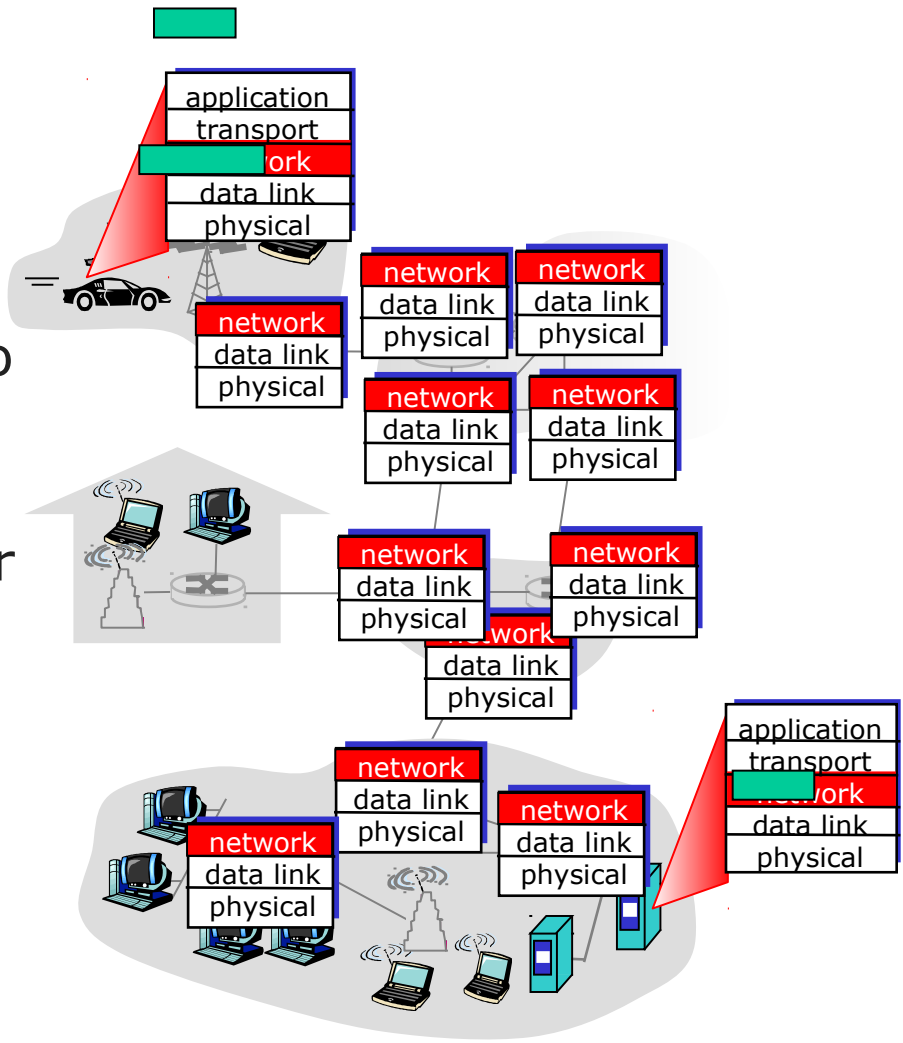
Vivek Shah

Based on slides compiled by
Marcos Vaz Salles



Network Layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



Source: Kurose & Ross

Two Key Network-Layer Functions

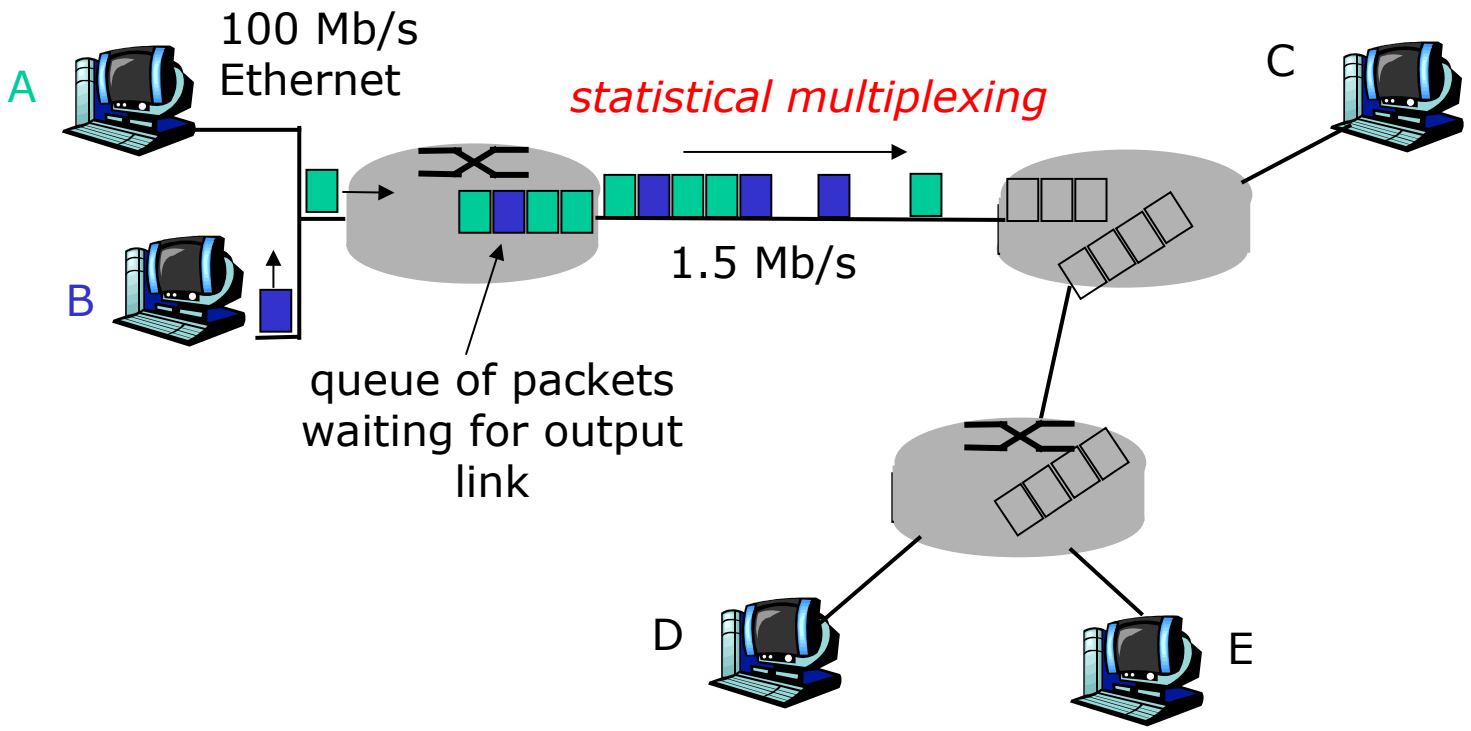
- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.
  - *routing algorithms*

analogy:

- routing: process of planning trip from source to dest
- forwarding: process of getting through single interchange

Source: Kurose & Ross

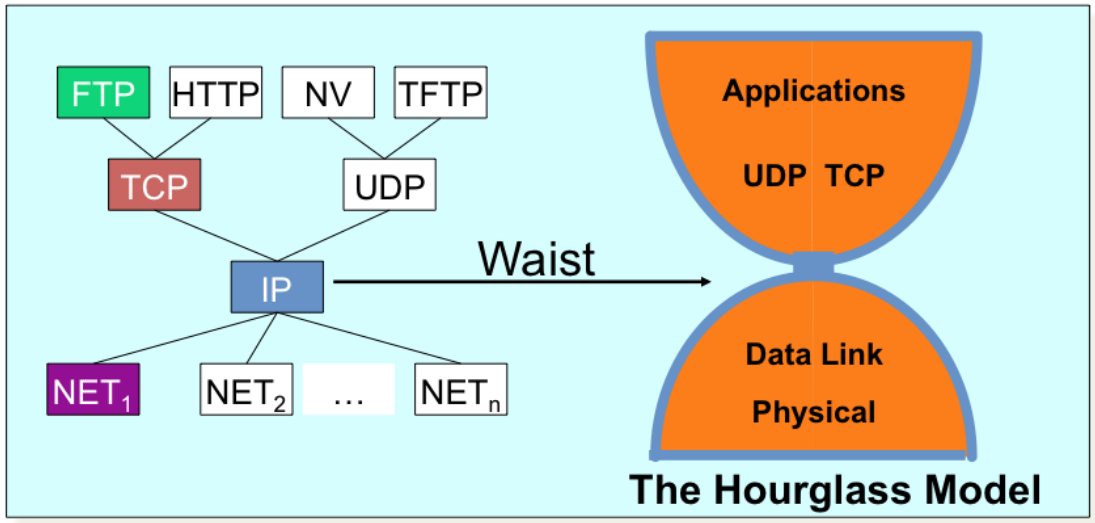
Recap: Packet Switching, Statistical Multiplexing



- sequence of A & B packets has no fixed timing pattern
  - bandwidth shared on demand: statistical multiplexing.
- TDM: each host gets same slot in revolving TDM frame.

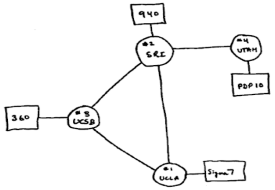
Source: Kurose & Ross

Recap: The Internet Protocol Suite



- Did IP provide any of the following services:
  - Bandwidth guarantees?
  - No loss?
  - Ordered delivery of packets?
  - Timing guarantees on delivery?
  - Congestion feedback?
- Why?

The waist facilitates interoperability



- What service does IP actually provide?

Source: Freedman (partial)

Network Layer Service Models

Service Model	Guarantees ?				Congestion feedback
	Bandwidth	Loss	Order	Timing	
best effort	none	no	no	no	no (inferred via loss)
CBR	constant rate	yes	yes	yes	no congestion
VBR	guaranteed rate	yes	yes	yes	no congestion
ABR	guaranteed minimum	no	yes	no	yes
UBR	none	no	yes	no	no

Source: Kurose & Ross

IP Service: Best-Effort is Enough

Source: Freedman

- **No error detection or correction**
  - Higher-level protocol can provide error checking
- **Successive packets may not follow the same path**
  - Not a problem as long as packets reach the destination
- **Packets can be delivered out-of-order**
  - Receiver can put packets back in order (if necessary)
- **Packets may be lost or arbitrarily delayed**
  - Sender can send the packets again (if desired)
- **No network congestion control (beyond "drop")**
  - Sender can slow down in response to loss or delay

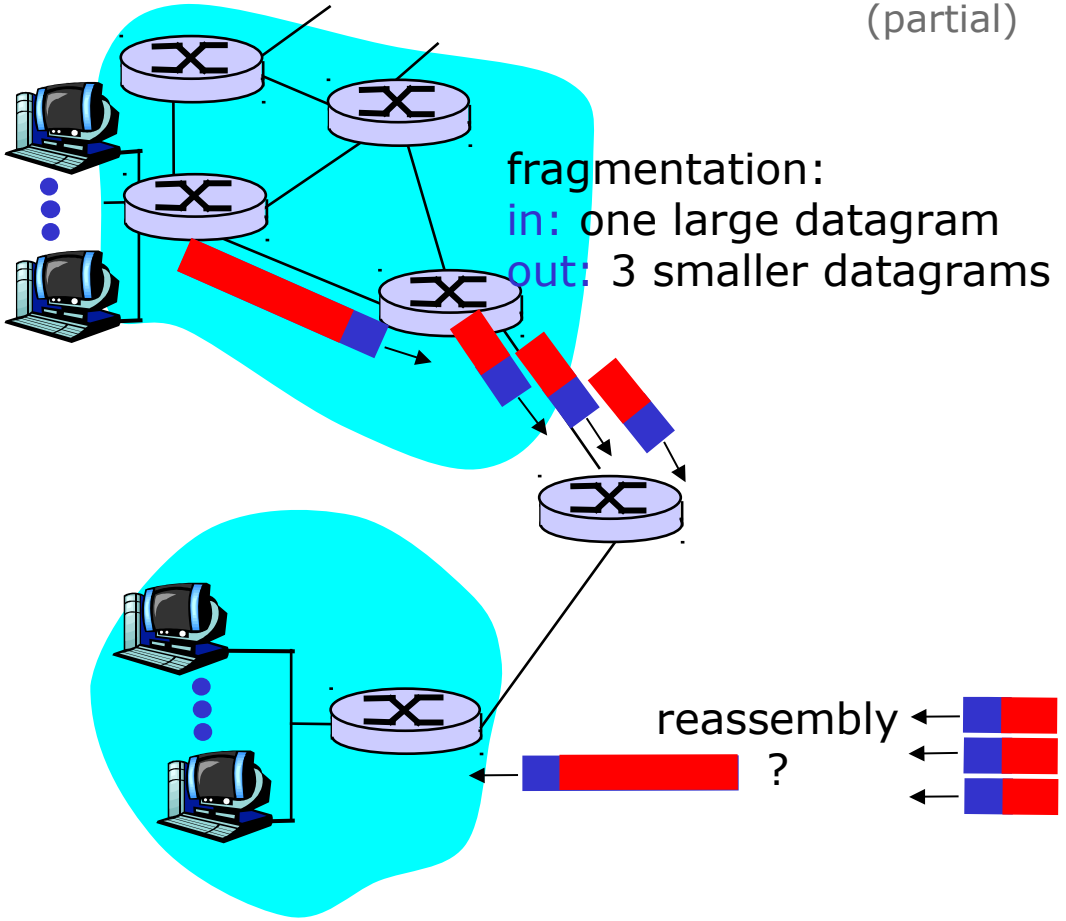
END-TO-END ARGUMENTS IN SYSTEM DESIGN

J.H. Saltzer, D.P. Reed and D.D. Clark*

IP Fragmentation and Reassembly

Source: Kurose & Ross (partial)

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net



Where should reassembly occur? In the network or at the final destination? Why?

History: Why IP Packets?

- IP proposed in the early 1970s
  - Defense Advanced Research Project Agency (DARPA)
- Goal: connect existing networks
  - Multiplexed utilization of existing networks
  - E.g., connect packet radio networks to the ARPAnet
- Motivating applications
  - Remote login to server machines
  - Inherently bursty traffic with long silent periods
- Prior ARPAnet experience with packet switching
  - Previously showed store-and-forward packet switching

Source: Freedman

Other Main Driving Goals (In Order)

- Communication should continue despite failures
  - Survive equipment failure or physical attack
  - Traffic between two hosts continue on another path
- Support multiple types of communication services
  - Differing requirements for speed, latency, & reliability
  - Bidirectional reliable delivery vs. message service
- Accommodate a variety of networks
  - Both military and commercial facilities
  - Minimize assumptions about the underlying network

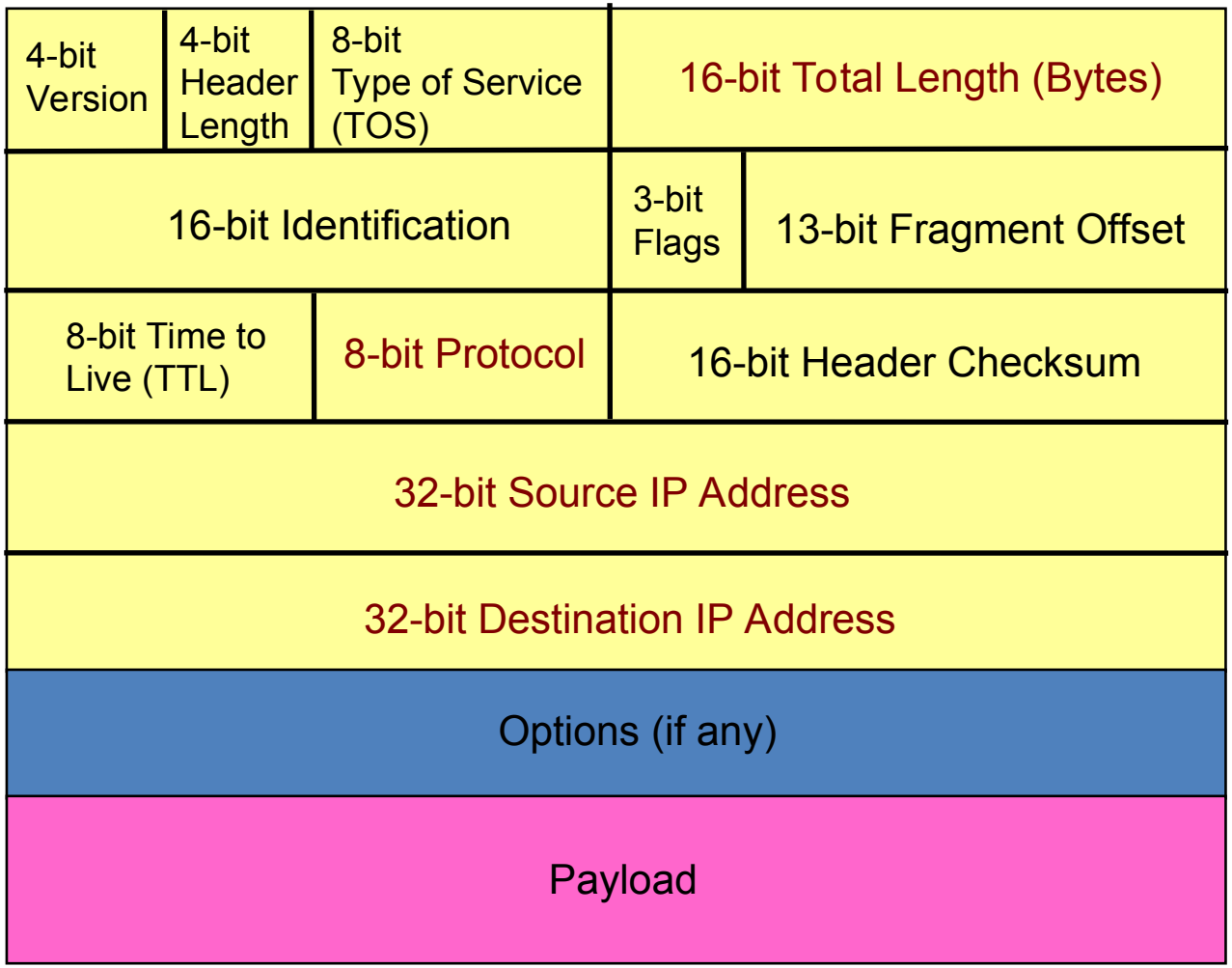
Source: Freedman

Other Driving Goals, Somewhat Met

- Permit distributed management of resources
  - Nodes managed by different institutions
  - … though this is still rather challenging
- Cost-effectiveness
  - Statistical multiplexing through packet switching
  - … though packet headers and re-transmissions wasteful
- Ease of attaching new hosts
  - Standard implementations of end-host protocols
  - … though still need a fair amount of end-host software
- Accountability for use of resources
  - Monitoring functions in the nodes
  - … though this is still fairly limited and immature

Source: Freedman

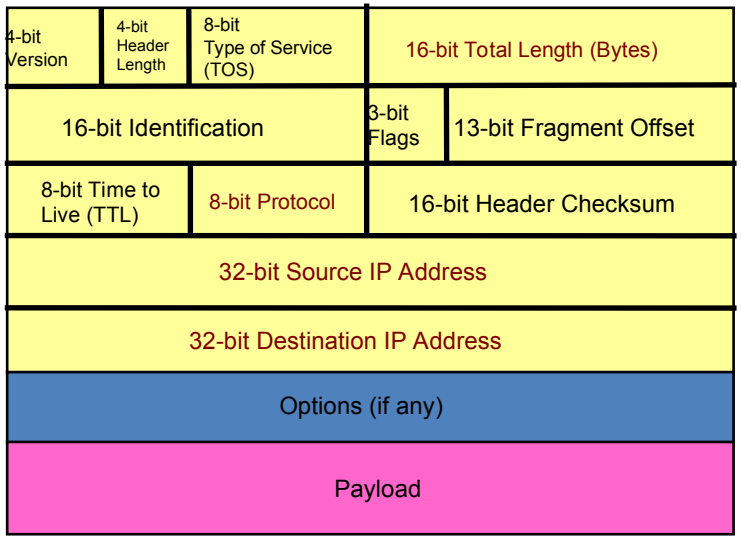
IP Packet Structure (IPv4)



Source: Freedman

IP Header: Version, Length, ToS

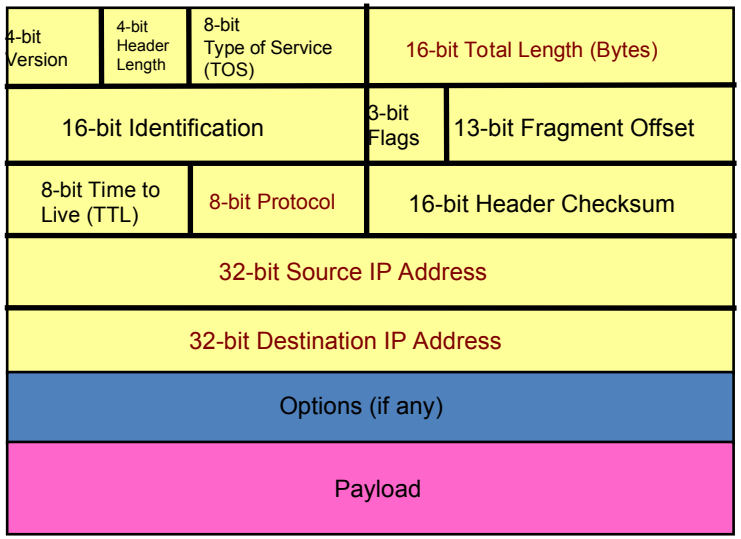
- **IP Version number (4 bits)**
  - Necessary to know what other fields to expect: how to parse?
  - "4" (for IPv4), "6" (for IPv6)
- **Header length (4 bits)**
  - # of 32-bit words in header
  - Typically "5" for 20-byte IPv4 header, more if "IP options"
- **Type-of-Service (8 bits)**
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high b/w for bulk transfer



Source: Freedman (partial)

IP Header: Length, Fragments, TTL

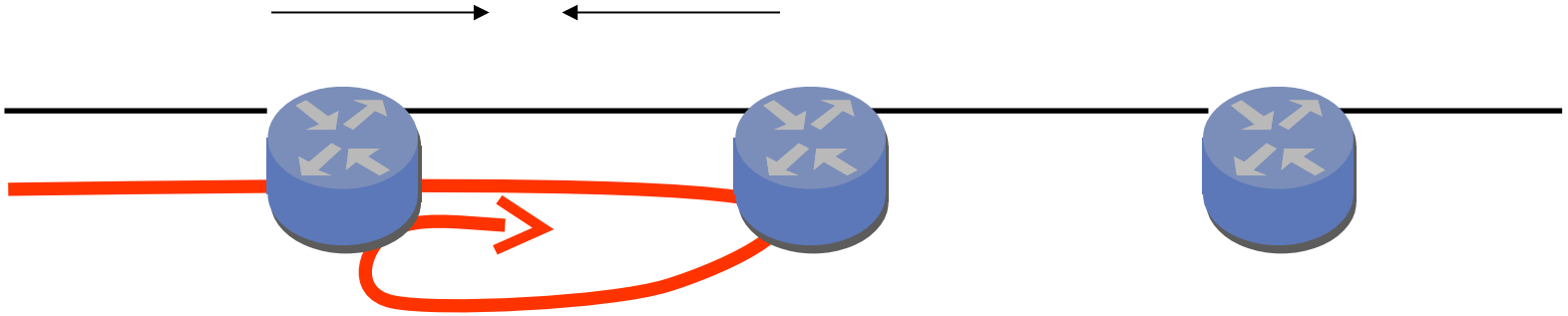
- **Total length (16 bits)**
  - # of bytes in the packet
  - Max size is 63,535 bytes (2^{16} -1)
  - Links may have harder limits: Ethernet "Max Transmission Unit" (MTU) commonly 1500 bytes
- **Fragmentation information (32 bits)**
  - Packet identifier, flags, and fragment offset
  - Split large IP packet into fragments if link cannot handle size
- **Time-To-Live (8 bits)**
  - Helps identify packets stuck in forwarding loops
  - … and eventually discard from network



Source: Freedman (partial)

IP Header: More on Time-to-Live (TTL)

- Potential robustness problem
  - Forwarding loops can cause packets to cycle forever
  - Confusing if the packet arrives much later

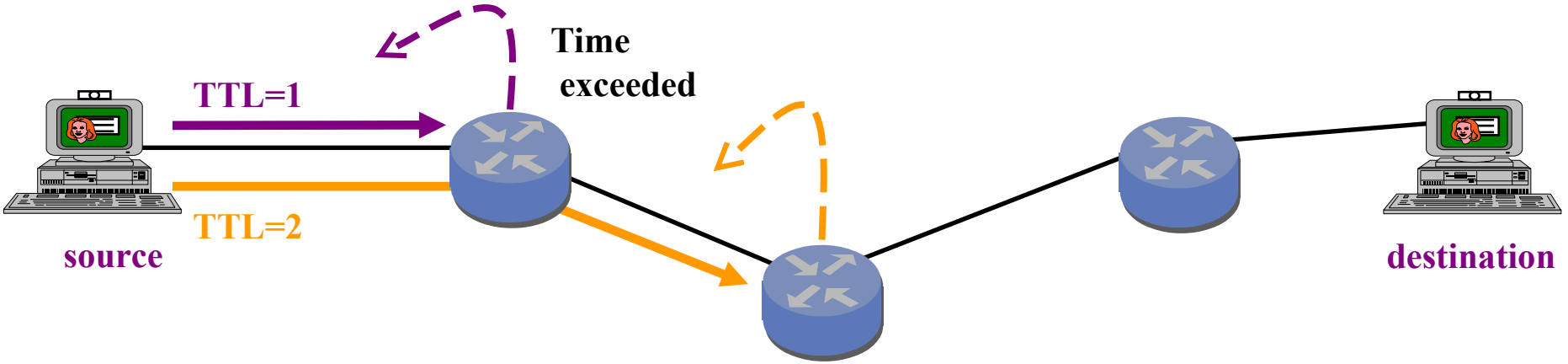


- Time-to-live field in packet header
  - TTL field decremented by each router on path
  - Packet is discarded when TTL field reaches 0…
  - …and "time exceeded" message (ICMP) sent to source

Source: Freedman

IP Header: Use of TTL in Traceroute

- Time-To-Live field in IP packet header
  - Source sends a packet with a TTL of *n*
  - Each router along the path decrements the TTL
  - "TTL exceeded" sent when TTL reaches *0*
- Traceroute tool exploits this TTL behavior

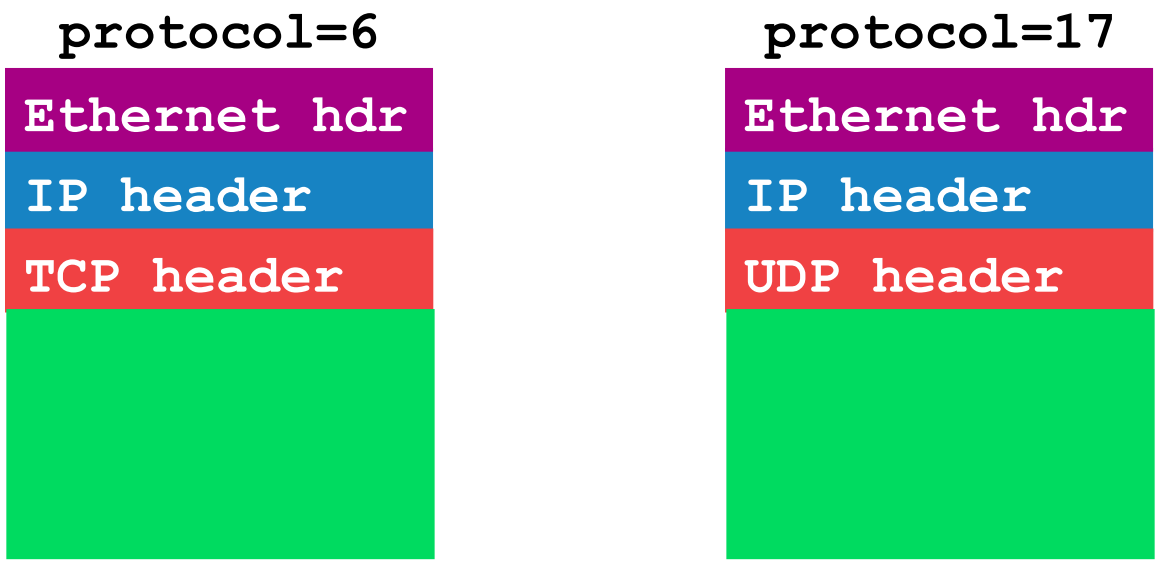


Send packets with TTL=1, 2, …
and record source of "time exceeded" message

Source: Freedman (partial)

IP Header Fields: Transport Protocol

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



Source: Freedman

IP Header: To and From Addresses

- Two IP addresses
  - Source and destination (32 bits each)
- **Destination address**
  - Unique identifier for receiving host
  - Allows each node to make forwarding decisions
- **Source address**
  - Unique identifier for sending host
  - Enables recipient to send a reply back to source

Source: Freedman

What about IPv6?

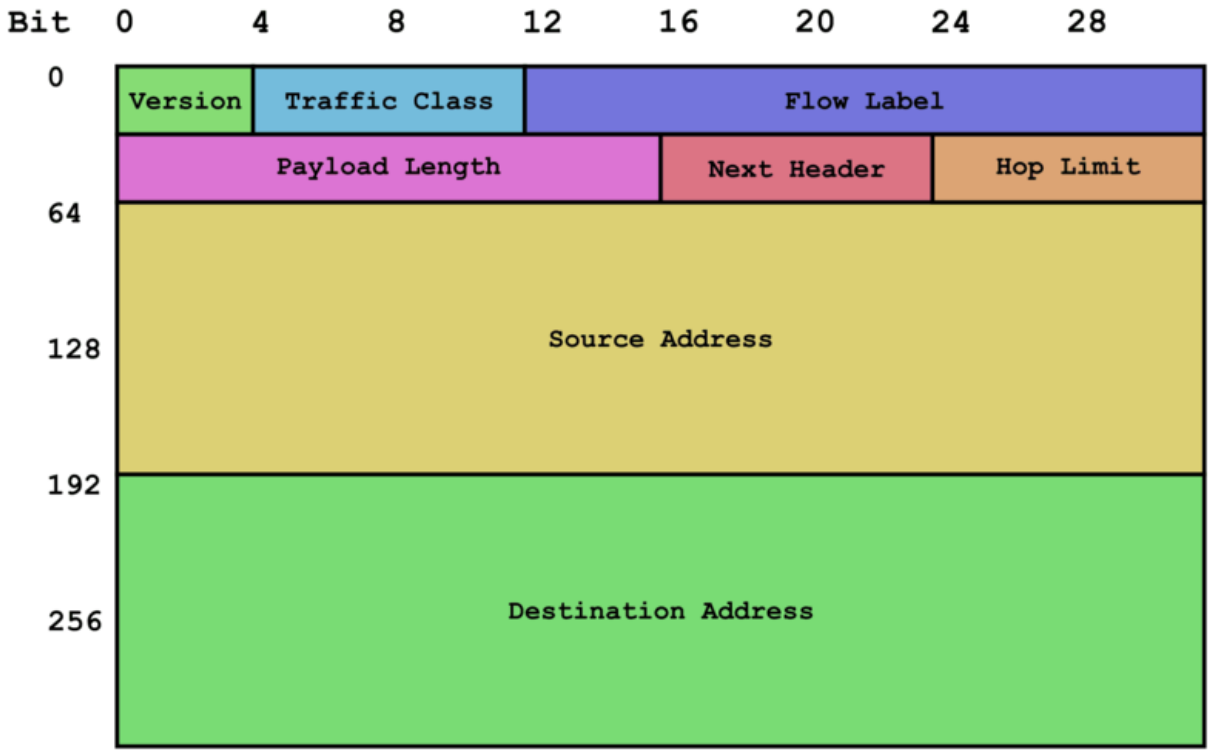


Image Source: Wikimedia Commons

- Similar format, but:
  - 128-bit addresses
  - No fragmentation, no checksum, options optional ☺

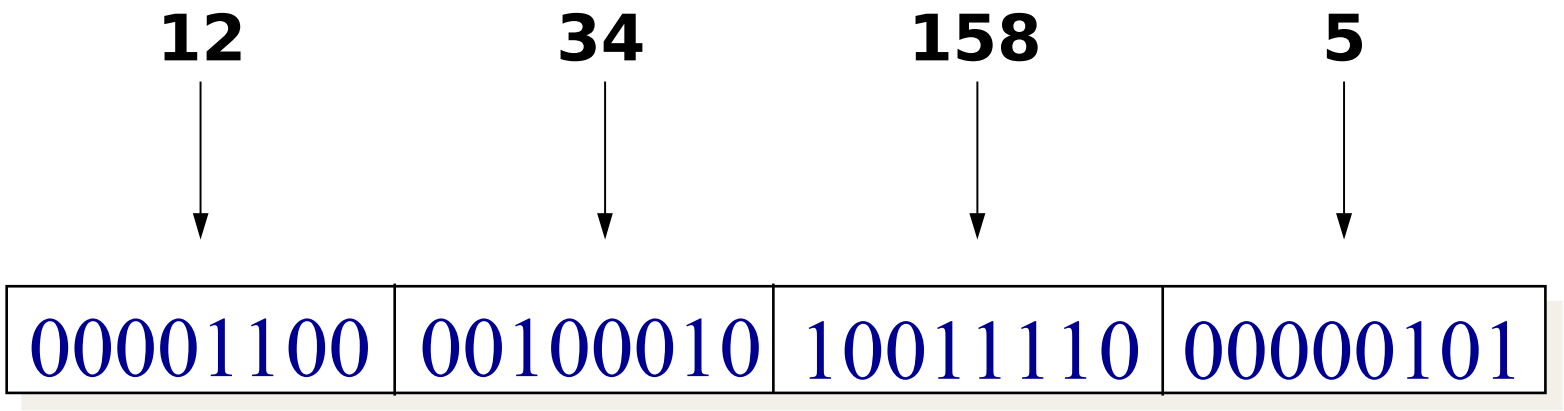
Source Address: What if Source Lies?

- Source address should be the sending host
  - But, who's checking? You can "spoof" any address!
- Why would someone want to do this?
  - Launch a denial-of-service attack
    - Send excessive packets to destination
    - … to overload node, or links leading to it
  - Evade detection by "spoofing"
    - But, victim could identify you by source addr, so lie!
  - Also, an attack against the spoofed host
    - Spoofed host is wrongly blamed
    - Spoofed host may receive return traffic from receiver

Source: Freedman

IP Address (IPv4)

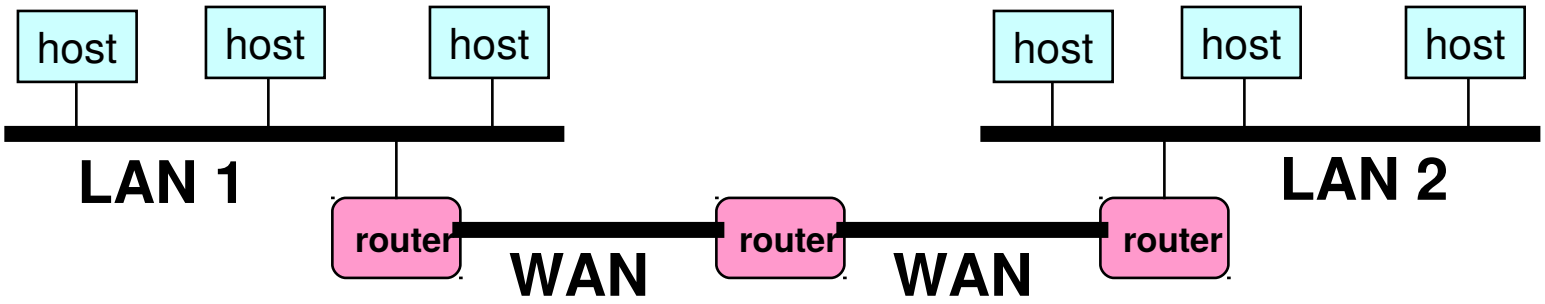
- A unique 32-bit number
- Identifies an interface (on a host, on a router, …)
- Represented in dotted-quad notation



Source: Freedman

Grouping Related Hosts

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



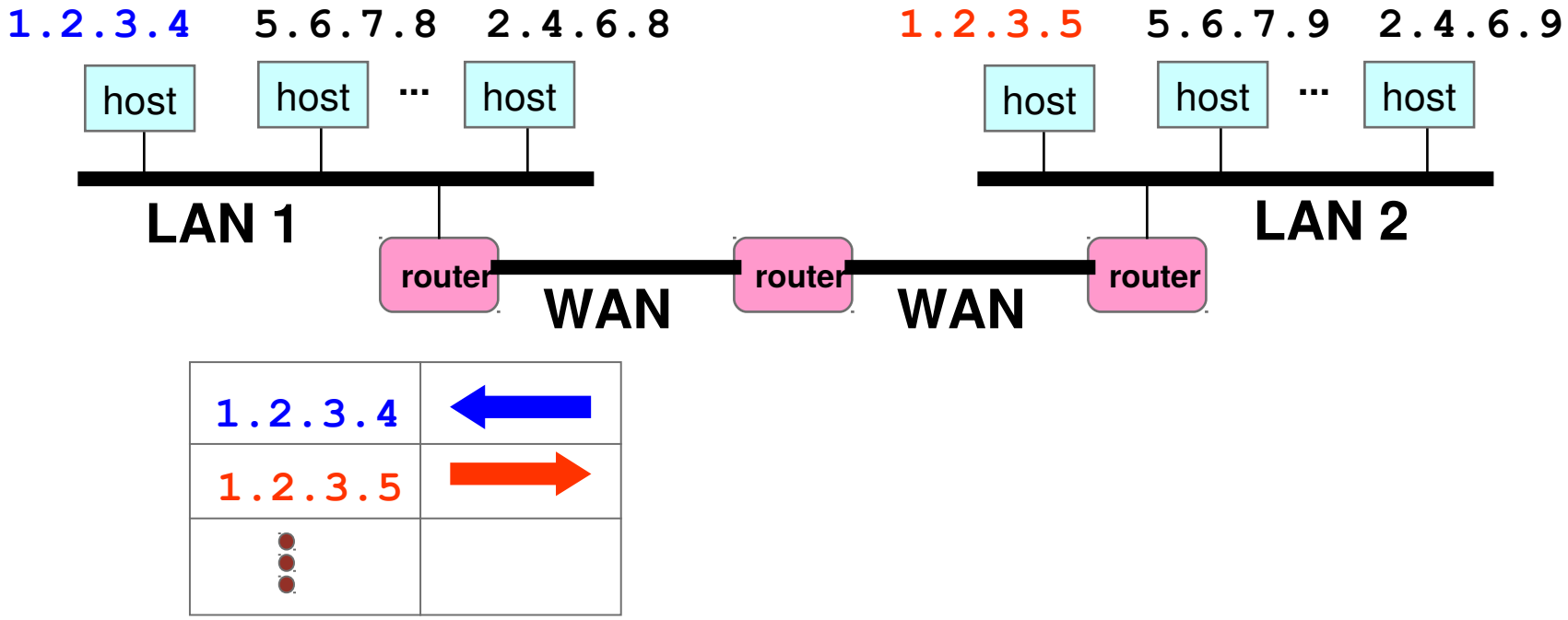
LAN = Local Area Network

WAN = Wide Area Network

Source: Freedman

Scalability Challenge

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



forwarding table a.k.a. FIB (forwarding information base)

Source: Freedman

Scalability Challenge

- Suppose hosts had arbitrary addresses
  - Then every router would need a lot of information
  - …to know how to direct packets toward *every* host
- Back of envelop calculations
  - 32-bit IP address: 4.29 billion (2^{32}) possibilities
  - How much storage?
    - Minimum: 4B address + 2B forwarding info per line
    - Total: 24.58 GB just for forwarding table
  - What happens if a network link gets cut?

Source: Freedman

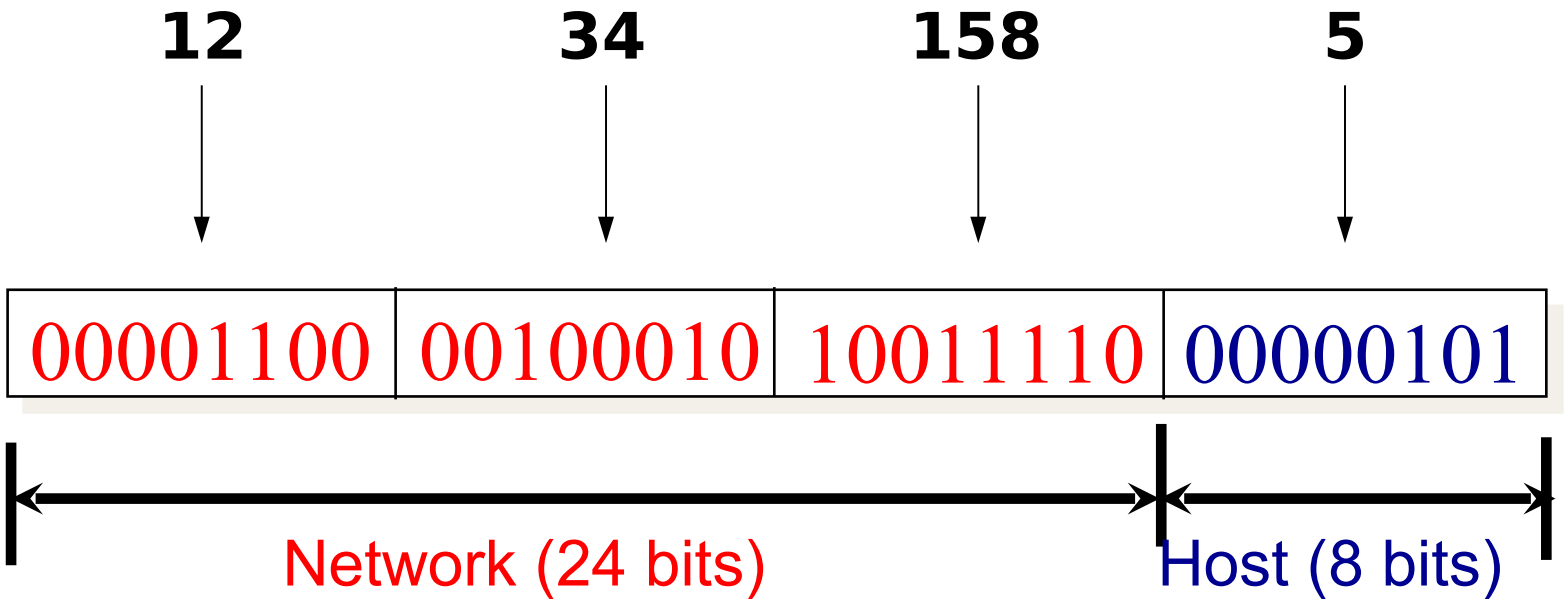
Standard CS Trick

- Have a scalability problem?
- Introduce hierarchy…

Source: Freedman

Hierarchical Addressing: IP Prefixes

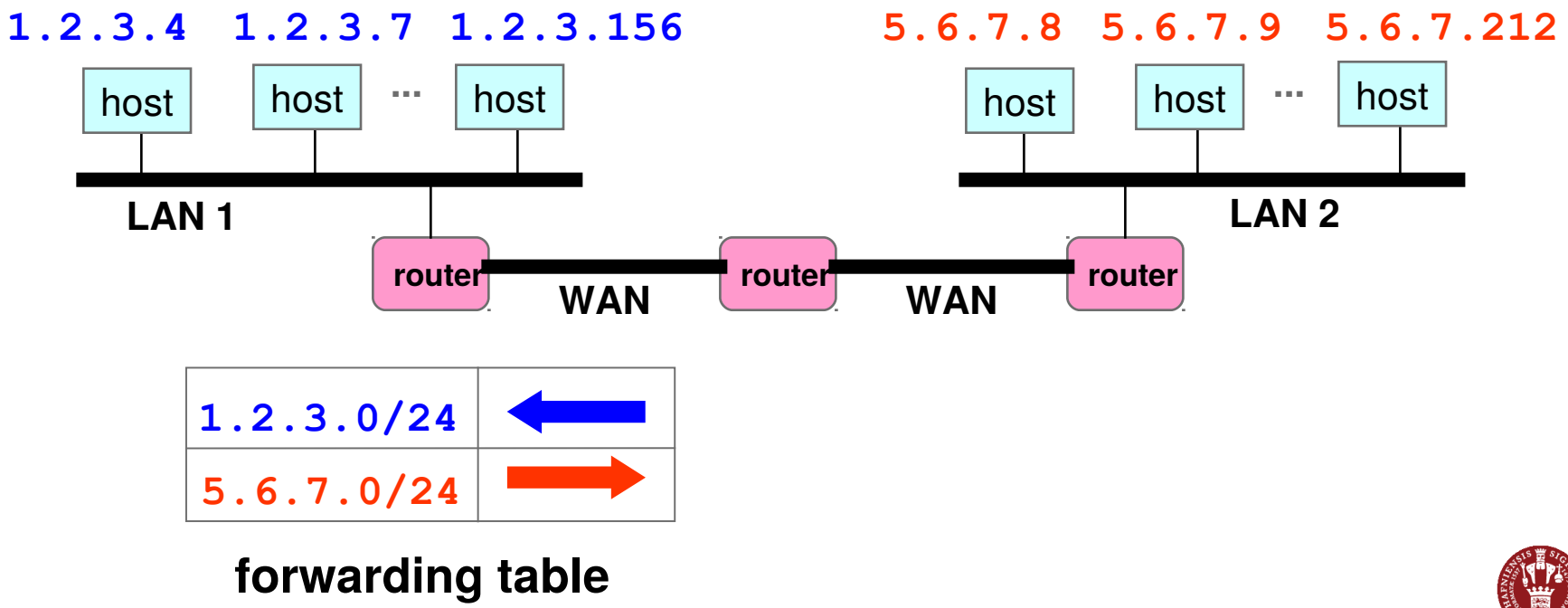
- IP addresses can be divided into two portions
  - Network (left) and host (right)
- 12.34.158.0/24 is a 24-bit **prefix**
  - Which covers 2^8 addresses (e.g., up to 255 hosts)



Source: Freedman

Scalability Improved

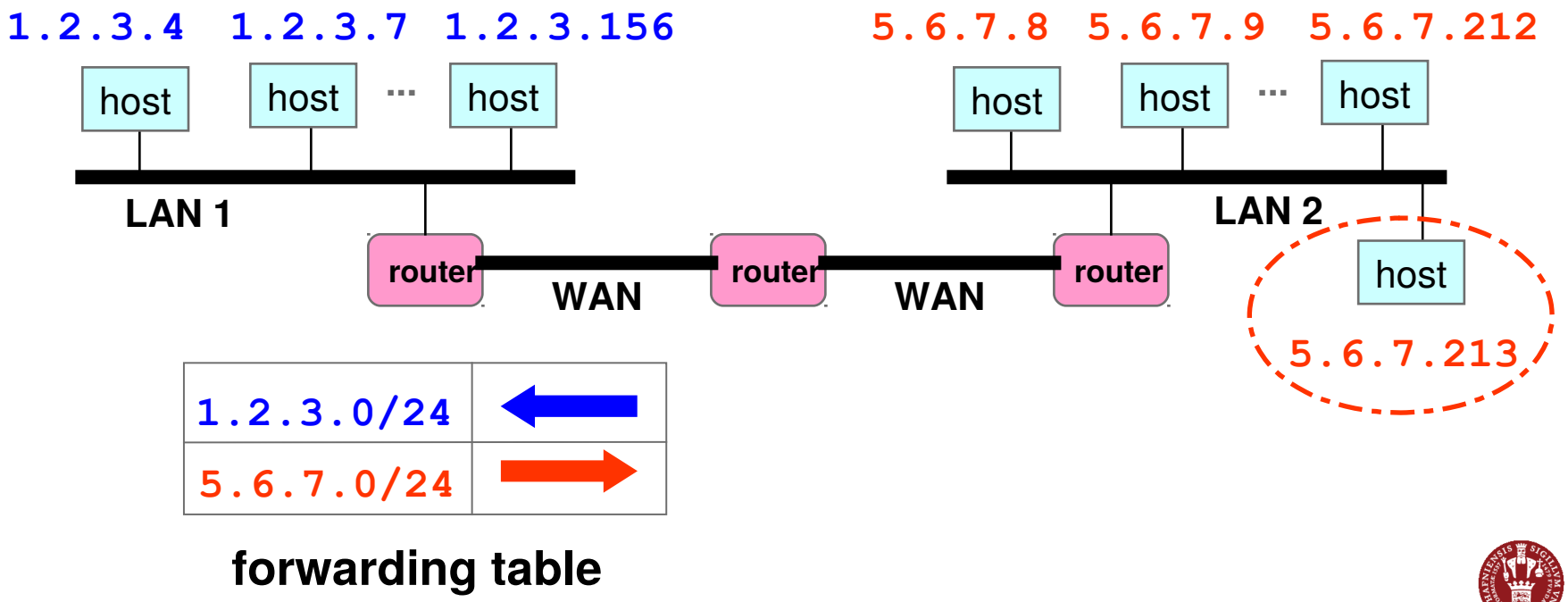
- Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN



Source: Freedman

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-table entry

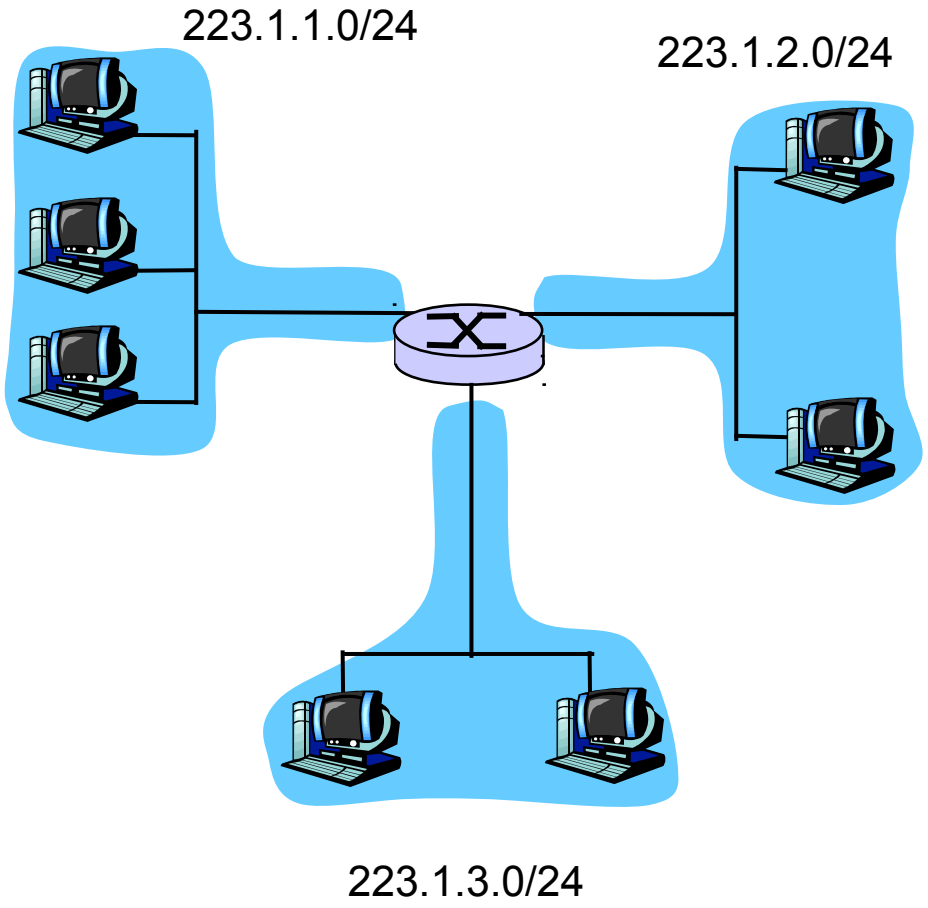


Source: Freedman

Subnets

Recipe

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



Subnet mask: /24

Source: Kurose & Ross

Address Allocation: Classful Addresssing

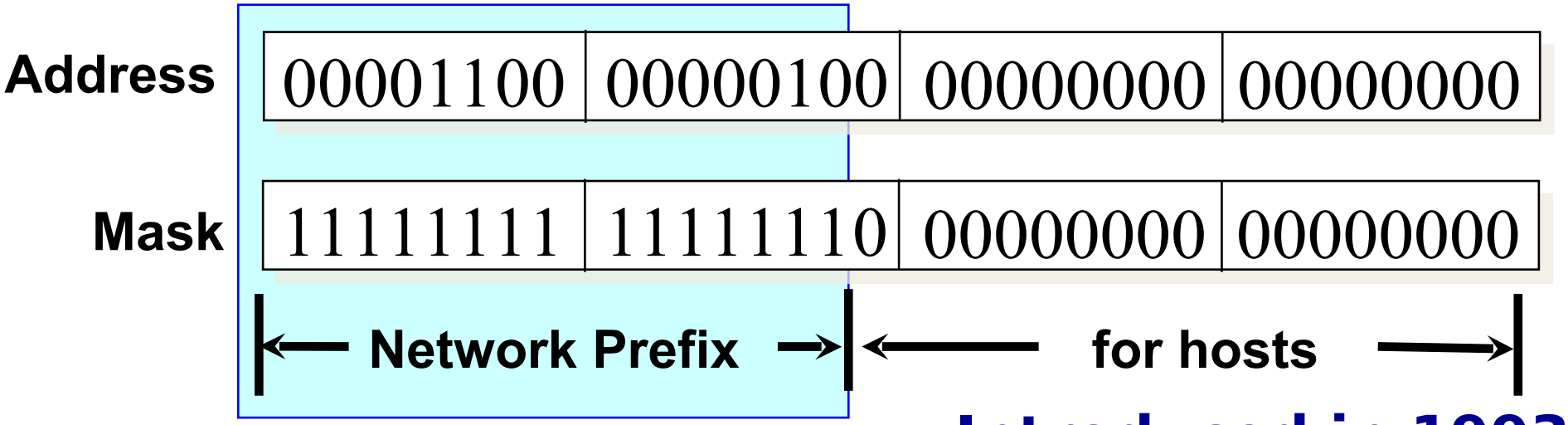
- In olden days, only fixed allocation sizes
  - Class A: 0* : Very large /8 blocks (MIT has 18.0.0.0/8)
  - Class B: 10* : Large /16 blocks (Princeton has 128.112.0.0/16)
  - Class C: 110* : Small /24 blocks
  - Class D: 1110* : Multicast groups
  - Class E: 11110* : Reserved for future use
- Why folks use dotted-quad notation!
- Position of "first 0" made it easy to determine class of address in hardware (hence, how to parse)

Source: Freedman (partial)

Classless Inter-Domain Routing (CIDR)

Source: Freedman

- IP prefix = IP address (AND) subnet mask
- IP Address : 12.4.0.0, Mask: 255.254.0.0



Written as 12.4.0.0/15

Introduced in 1993
RFC 1518-1519

```
$ ifconfig
en1: flags=8863<UP,BROADCAST,…,MULTICAST> mtu 1500
     inet 192.168.1.1 netmask 0xffffff00 broadcast 192.168.1.255
     ether 21:23:0e:f3:51:3a
```

IP Calculations

- Consider the following IP address of a host:

  220.224.30.82/24

- Determine:
  - The network address (i.e., IP prefix)
  - The broadcast address of the network
  - The min/max addresses for *hosts* in the network
    - And by consequence, the number of possible hosts

What about IPv6?

- Similar CIDR methodology, but 128-bit instead of 32-bit addresses are used
  - From little over 4 billion to "little" over 340 undecillion addresses → ~3.4 x 10^{38} addresses
- **Address notation**
  - 8 groups of 4 hexadecimal digits (16-bits in a group)

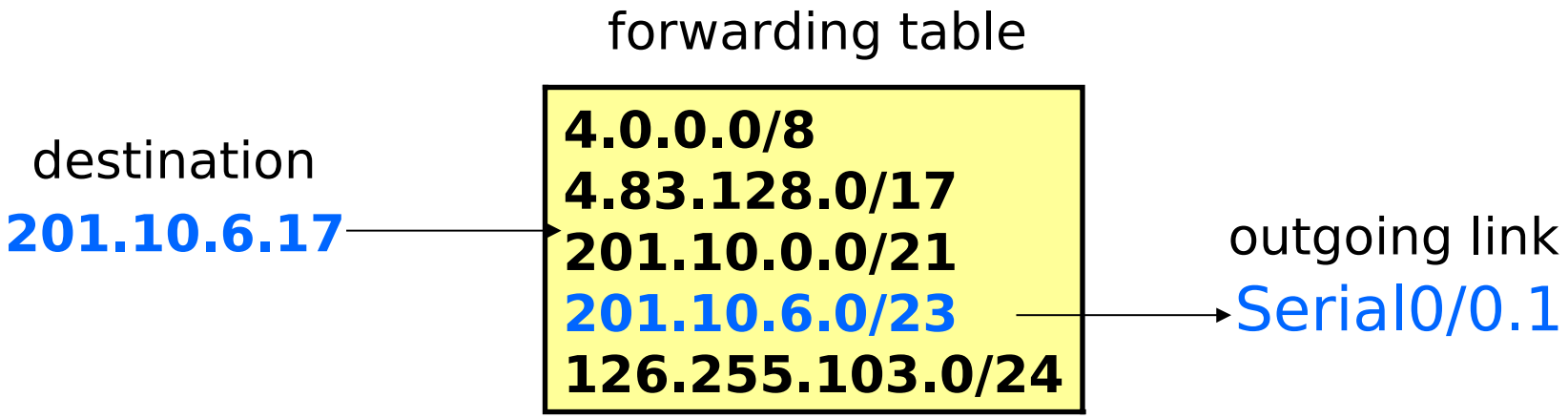
2001:0db8:85a3:0000:0000:8a2e:0370:7334

  - Leading zeros omitted and leftmost string of zero groups compressed to ::

2001:db8:85a3::8a2e:370:7334

Forwarding Revisited

- How to resolve multiple matches?
  - Router identifies most specific prefix: *longest prefix match (LPM)*
  - Cute algorithmic problem to achieve fast lookups



Source: Freedman (partial)

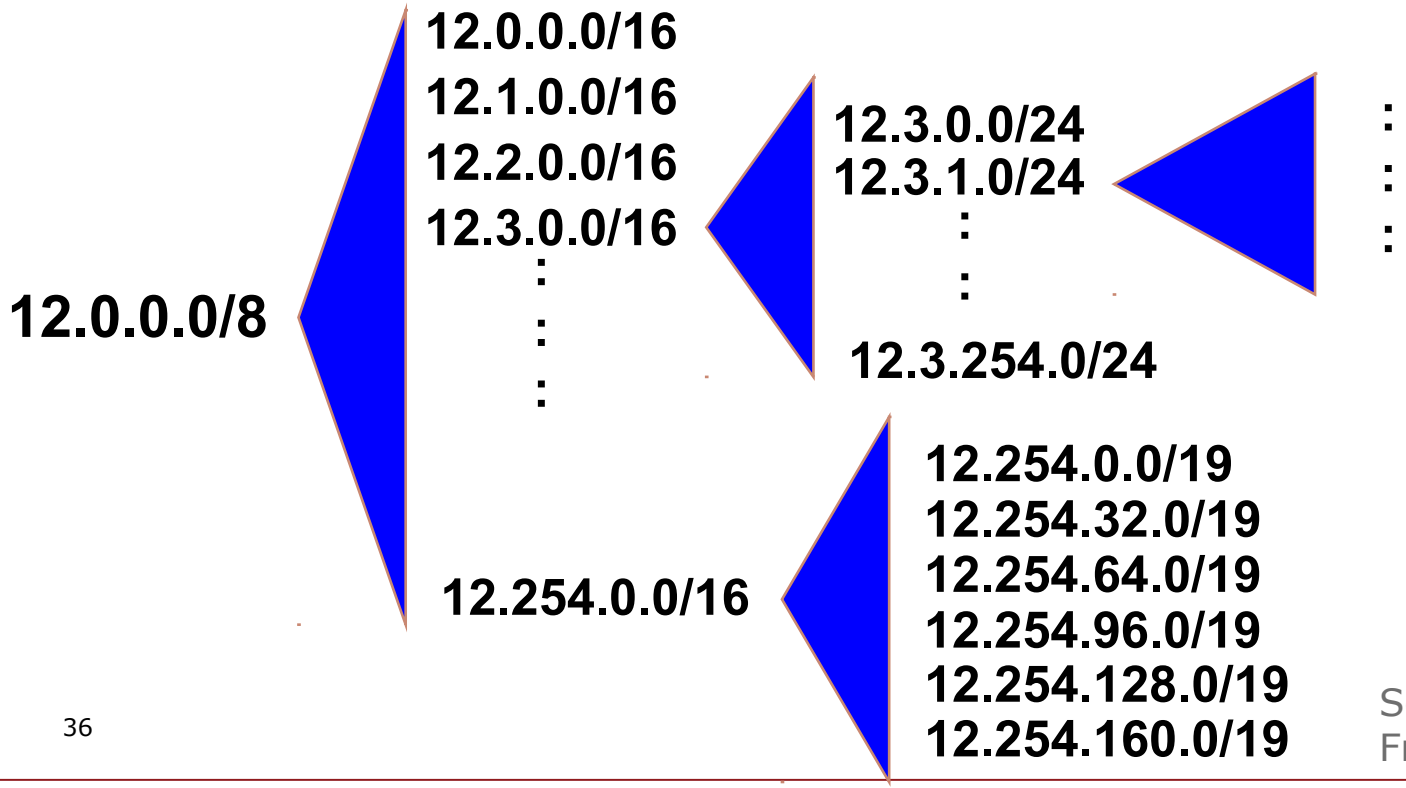
Simplest Algorithm is Too Slow

- Scan the forwarding table one entry at a time
  - Keep track of entry with longest-prefix (by netmask)
- Overhead is linear in size of forwarding table
  - Today, that means 350,000 entries!
  - How much time do you have to process?
    - Consider 10Gbps routers and 64B packets
    - 10^{10} / 8 / 64: 19,531,250 packets per second
    - 51 nanoseconds per packet
- Need greater efficiency to keep up with *line rate*
  - Better algorithms
  - Hardware implementations

Source: Freedman (partial)

CIDR: Hierarchal Address Allocation

- Prefixes are key to Internet scalability
  - Address allocated in contiguous chunks (prefixes)
  - Routing protocols and packet forwarding based on prefixes
  - Today, routing tables contain ~350,000 prefixes (vs. 4B)



Source: Freedman

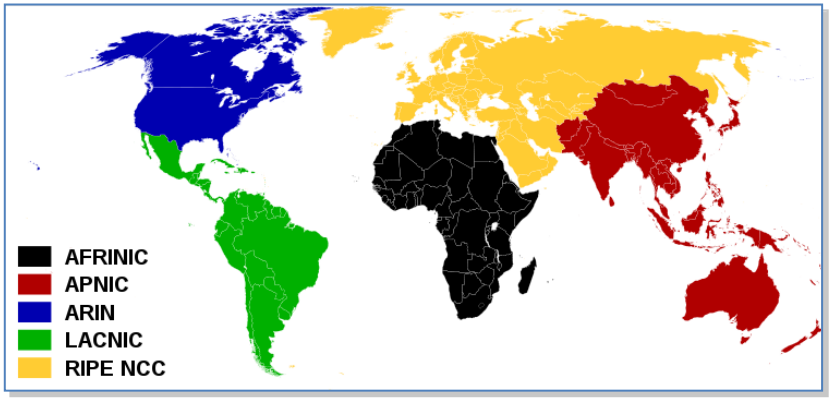
Obtaining a Block of Addresses

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

Internet Corp. for Assigned Names and Numbers (IANA)

Regional Internet Registries (RIRs)

Internet Service Providers (ISPs)



Source: Freedman

Are 32-bit Addresses Enough?

- Not all that many unique addresses
  - 2^{32} = 4,294,967,296 (just over four billion)
  - Some are reserved for special purposes
  - Addresses are allocated non-uniformly
    - A fraternity/dorm at MIT has as many IP addrs as Princeton!
- More devices need addr's: smartphones, toasters, …
- Long-term solution: a larger address space
  - IPv6 has 128-bit addresses ($2^{128} = 3.403 \times 10^{38}$)
- Short-term solutions: limping along with IPv4
  - Private addresses (RFC 1918):
    - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - Network address translation (NAT)
  - Dynamically-assigned addresses (DHCP)

Source: Freedman

Summary

- IP
  - THE Internet Protocol
    - Good enough is better than perfect
  - Fragmentation / reassembly, IP headers, TTL, spoofing
- IP addresses, prefixes, forwarding
  - IP addressing: practice IP prefix, num. hosts, min/max host, broadcast address calculations
  - Reading: Router switching fabrics in book