CSCI-1680
Network Layer:
IP Forwarding realities

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti

Administrivia

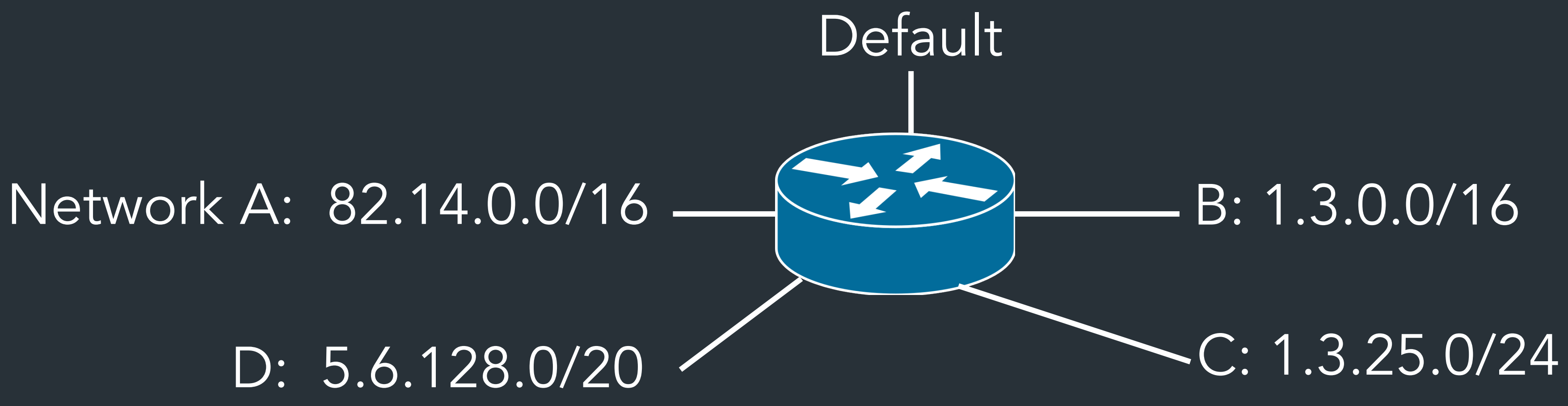
- Sign up for IP milestone meetings, preferably with your mentor TA, on or before Friday (Oct 6)
  - You don't need to show an implementation, but you are expected to talk about your design
  - Look for calendar link in email
- IP gearup II: Thursday 5-7pm in CIT368
  - Implementation and debugging tips
- HW1: Due Thursday (HW2 out either Thursday or next Tues)

Today

"Wrinkles" in IP forwarding

- Longest Prefix Match
- IP<->Link layer (ARP, DHCP)
- Network Address Translation (NAT)
- IPv6

After this: Routing



Prefix	IF/Next hop
82.14.0.0/16	(A)
1.3.0.0/16	(B)
1.3.4.0/24	(C)
5.6.128.0/20	(D)
0.0.0.0/0	(Default)

(X) is placeholder—could be an IP or an interface name

Warmup: based on the table, where would the router send packets destined for the following addresses:

1. 5.6.128.100
2. 1.3.1.1
3. 8.8.8.8
4. 1.3.4.8

What happens when prefixes overlap?

An IP can match on more than one row
=> need to pick the most specific (longest) prefix

Prefix	IF/Next hop
1.3.0.0/16	(B)
1.3.4.0/24	(C)
1.3.4.5/32	
0.0.0.0/0	(Default)

1.3.0.0/16	00000001 00000011 xxxxxxxx xxxxxxxx
1.3.4.0/24	00000001 00000011 00000100 xxxxxxxx

What happens when prefixes overlap?

An IP can match on more than one row
=> <u>need to pick the most specific (longest) prefix</u>

Prefix	IF/Next hop
1.3.0.0/16	(B)
1.3.4.0/24	(C)
1.3.4.5/32	
0.0.0.0/0	(Default)

1.3.0.0/16 00000001 00000011 xxxxxxxx xxxxxxxx

1.3.4.0/24 00000001 00000011 00000100 xxxxxxxx

More specific => best match!

What happens when prefixes overlap?

Prefix	IF/Next hop
1.3.0.0/16	(B)
1.3.4.0/24	(C)
1.3.4.5/32	
0.0.0.0/0	(Default)

An IP can match on more than one row
=> <u>need to pick the most specific (longest) prefix</u>

1.3.0.0/16 00000001 00000011 xxxxxxxx xxxxxxxx

1.3.4.0/24 00000001 00000011 00000100 xxxxxxxx

More specific => best match!

Other examples you'll see…

0.0.0.0/0 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx

1.2.3.5/32 00000001 00000011 00000100 00000101

What happens when prefixes overlap?

Prefix	IF/Next hop
1.3.0.0/16	(B)
1.3.4.0/24	(C)
1.3.4.5/32	
0.0.0.0/0	(Default)

An IP can match on more than one row
=> need to pick the most specific (longest) prefix

1.3.0.0/16 00000001 00000011 xxxxxxxx xxxxxxxx

1.3.4.0/24 00000001 00000011 00000100 xxxxxxxx

More specific => best match!

Other examples you'll see…

0.0.0.0/0 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx => Least specific!
(Used for default "catchall" routes)

1.2.3.5/32 00000001 00000011 00000100 00000101 => Most specific!
(Refers to a single host, often a local IP)

What happens when prefixes overlap?

Prefix	IF/Next hop
1.3.0.0/16	(B)
1.3.4.0/24	(C)
1.3.4.5/32	
0.0.0.0/0	(Default)

An IP can match on more than one row
=> need to pick the most specific (longest) prefix

1.3.0.0/16 00000001 00000011 xxxxxxxx xxxxxxxx

1.3.4.0/24 00000001 00000011 00000100 xxxxxxxx

More specific => best match!

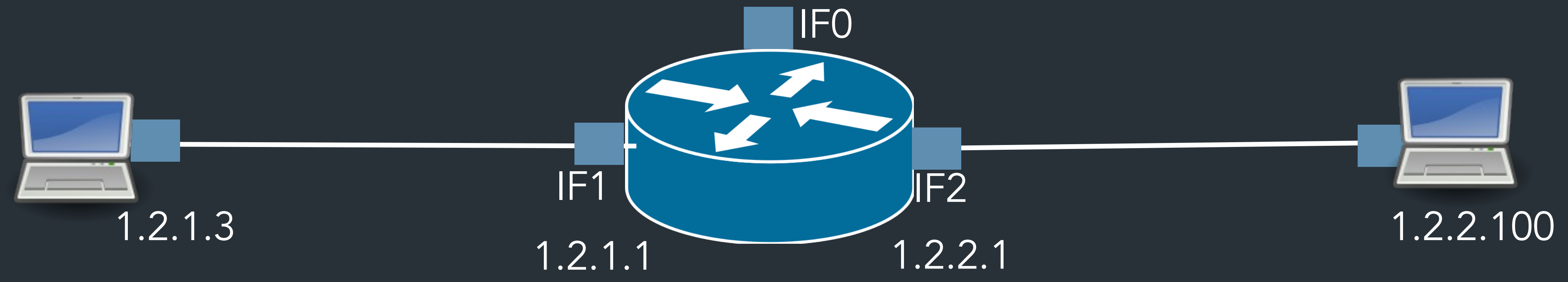
Other examples you'll see…

0.0.0.0/0 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx => Least specific! (Used for default "catchall" routes)

1.2.3.5/32 00000001 00000011 00000100 00000101 => Most specific! (Refers to a single host, often a local IP)

=>Longest prefix matching: can keep forwarding tables small by summarizing routes where possible, otherwise using specific prefixes

What happens at the link layer?



What does it mean to send to IF1?

The story so far:
=>Can "easily" communicate with nodes on the same network, but what about other networks?
=> Routers know about multiple networks, forward packets between them

Prefix	IF/Next hop
1.2.1.0/24	IF1
1.2.2.0/24	IF2
8.0.0.0/30	IF0
Default	8.0.0.2

"Local delivery": what does it mean to send to IF1?

So far: "easy" to communicate with nodes on the same network. But how?

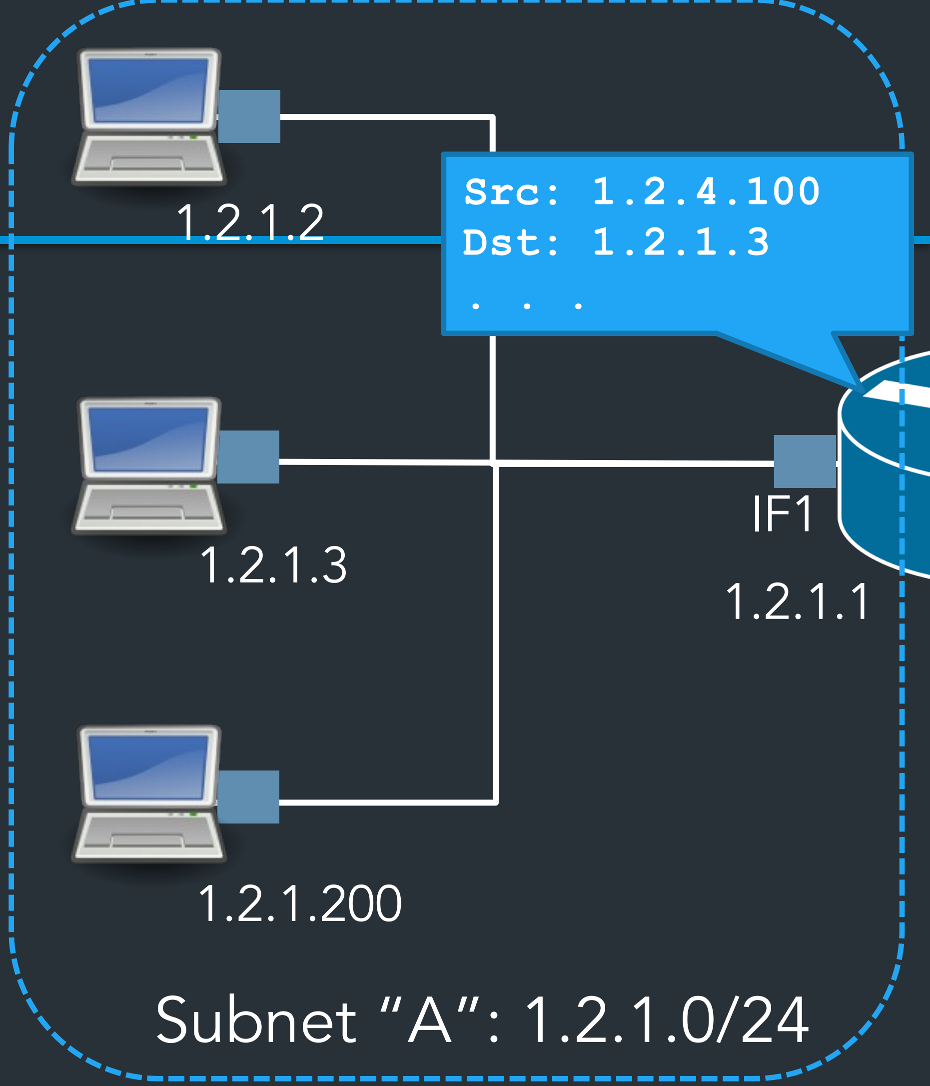
To send a packet on a local network, we need:

- Dest. IP (Network layer)
- Dest. MAC address (Link layer)

	Src	Dest
Link		???
IP	10.2.4.100	1.2.1.3

Assume: link layer can figure out the rest once we fill in this info

=> How do we get the MAC address?



Prefix	IF/Next hop
1.2.1.0/24	IF1
…	…

“Glue” between L2 and L3

Need a way to connect get link layer info (mac address) from network-layer info (IP address)

“What MAC address has IP 1.2.3.4?”

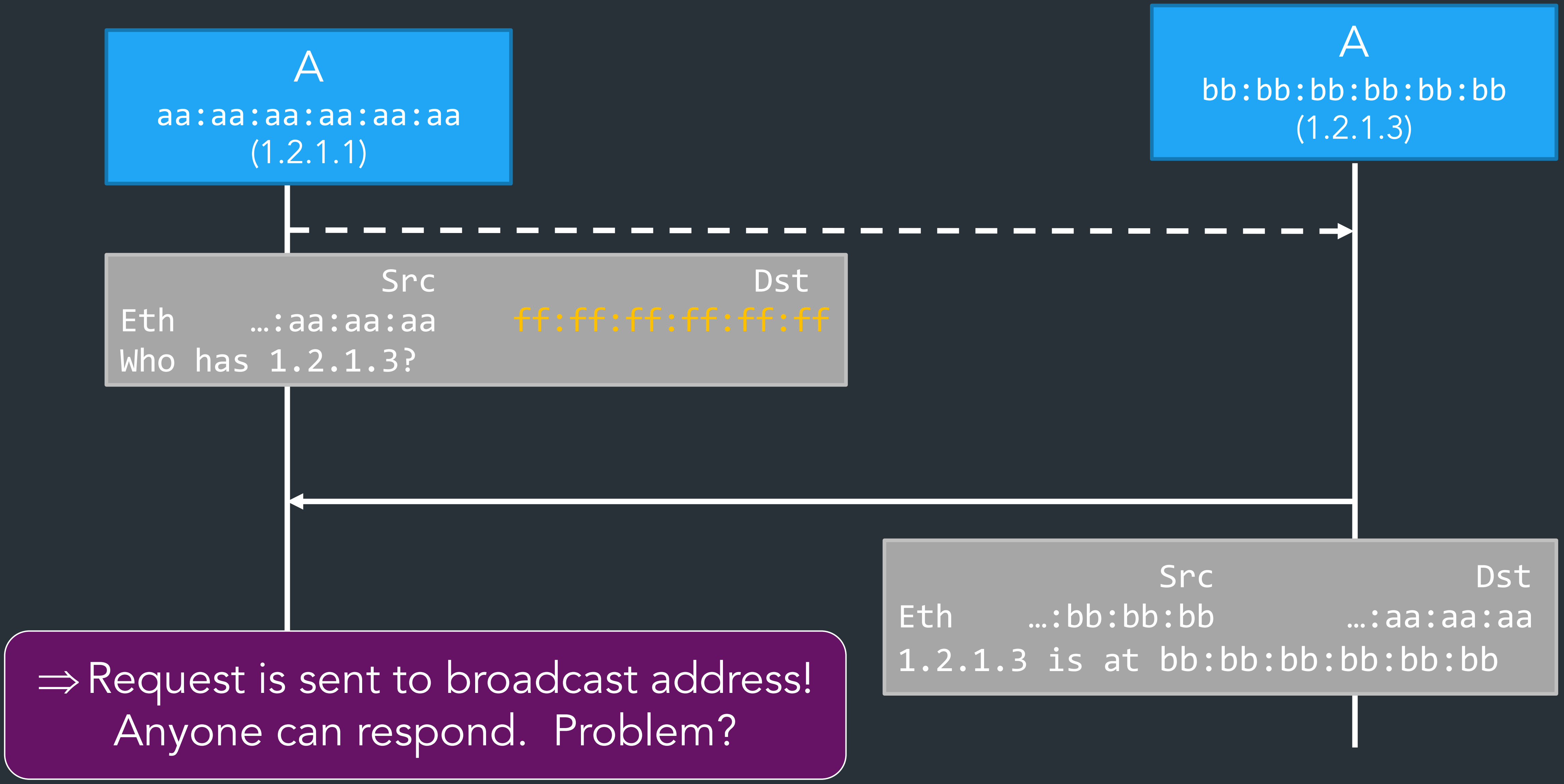
Ask the network!
=> Address Resolution Protocol (ARP)

ARP: Address resolution protocol

Given an IP address, ask network for the MAC address

- Maps IP addresses to mac addresses
  - Request: "`Who has 1.2.3.4?`"
  - Response: "`aa:bb:cc:dd:ee:ff is at 1.2.3.4`"

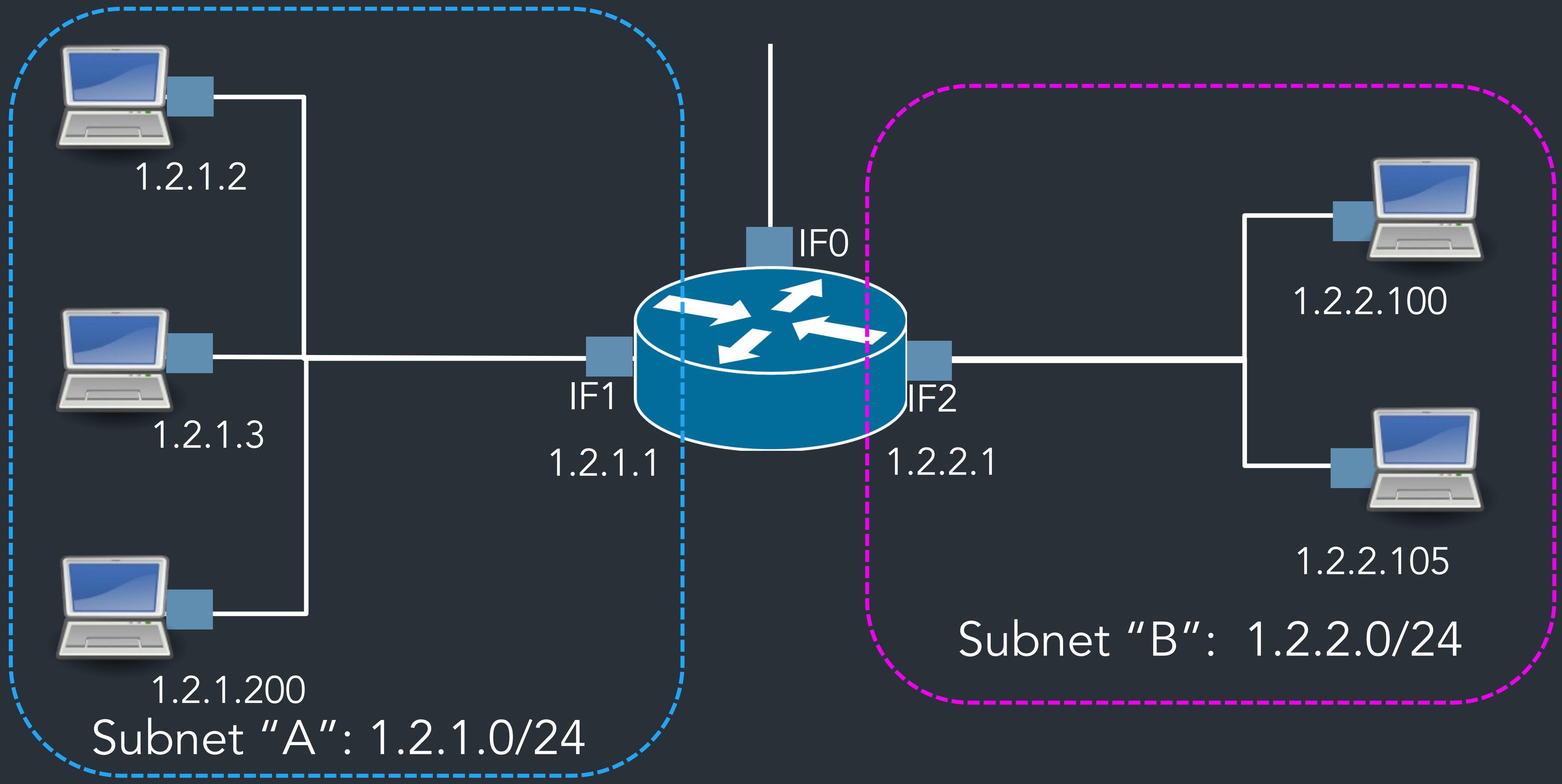
- ARP table: hosts cache IP->mac mappings
- Requests send to broadcast address: `ff:ff:ff:ff:ff:ff`
  - Anyone can respond: problem?



A
aa:aa:aa:aa:aa:aa
(1.2.1.1)
A
bb:bb:bb:bb:bb:bb
(1.2.1.3)
Src Dst
Eth …:aa:aa:aa ff:ff:ff:ff:ff:ff
Who has 1.2.1.3?
Src Dst
Eth …:bb:bb:bb …:aa:aa:aa
1.2.1.3 is at bb:bb:bb:bb:bb:bb
⇒ Request is sent to broadcast address!
Anyone can respond. Problem?

Example

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.17.44.1              ether   00:12:80:01:34:55   C                     eth0
172.17.44.25             ether   10:dd:b1:89:d5:f3   C                     eth0
172.17.44.6              ether   b8:27:eb:55:c3:45   C                     eth0
172.17.44.5              ether   00:1b:21:22:e0:22   C                     eth0
```


1.2.1.2
1.2.1.3
1.2.1.200
Subnet "A": 1.2.1.0/24
IF0
IF1
1.2.1.1
IF2
1.2.2.1
1.2.2.100
1.2.2.105
Subnet "B": 1.2.2.0/24

How do you get an IP address?

Getting an IP

Two ways to configure an IP:

- Static configuration: manually specify IP address, mask, gateway, …

- Automatic: ask the network for an IP when you connect!

Getting an IP

Two ways to configure an IP for a host:

- Static configuration: manually specify IP address, mask, gateway, …

  => More common with network devices that don't change often

- Automatic: ask the network for an IP when you connect!

  => Most common for end hosts

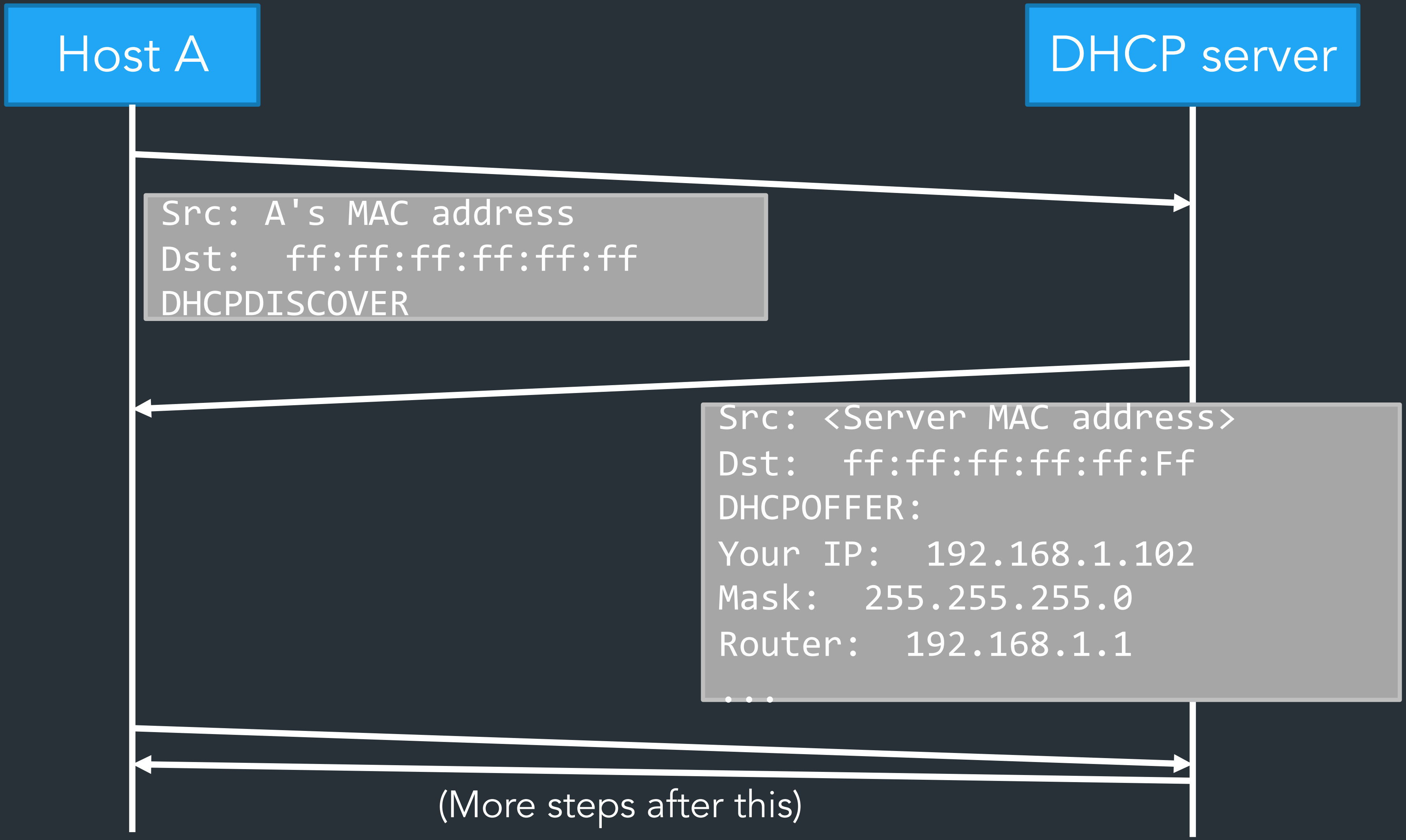
  => Dynamic Host Configuration Protocol (DHCP)

DHCP: The idea

- Networks are free to assign addresses within block to hosts
- Solution: Dynamic Host Configuration Protocol
  - Client: DHCP Discover to 255.255.255.255 (broadcast)
  - Server(s): DHCP Offer to 255.255.255.255 (why broadcast?)
  - Client: choose offer, DHCP Request (broadcast, why?)
  - Server: DHCP ACK (again broadcast)
- Result: address, gateway, netmask, DNS server

DHCP: The idea

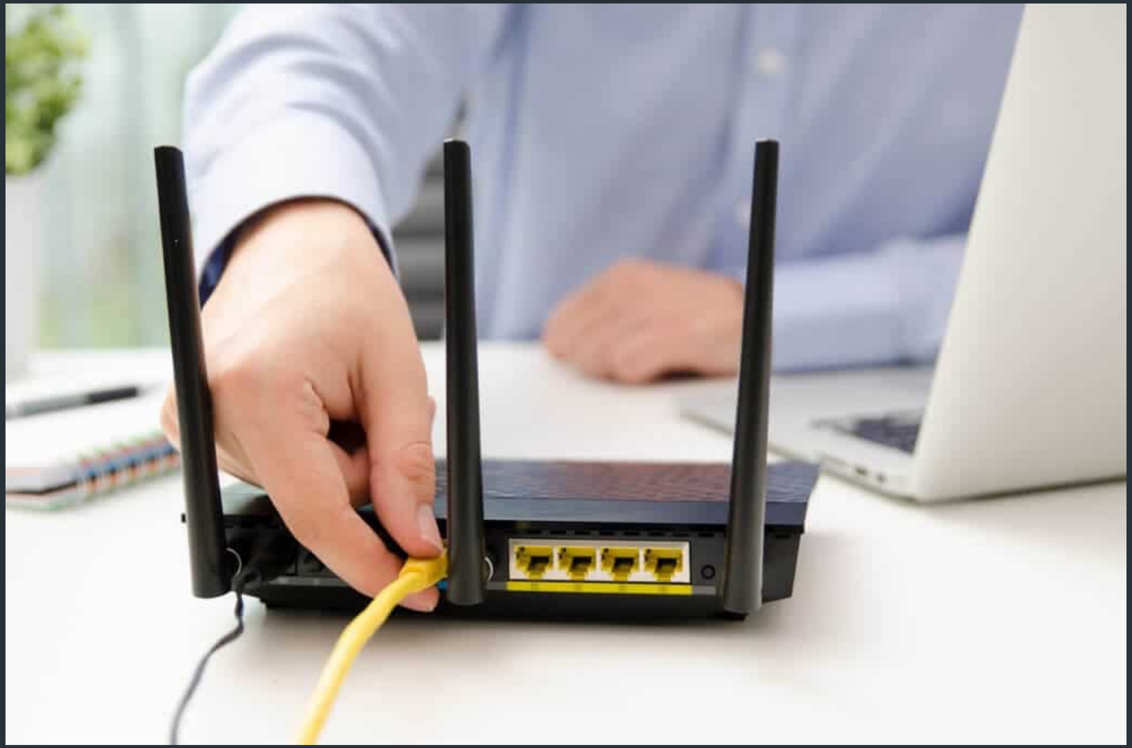
- Every network has a "pool" of IPs it can assign to hosts
  - Some subset of its prefix (eg. 192.168.1.0/24)
- When a host connects, it asks a DHCP server for an address from the pool
- DHCP server(s) act like allocators: give "leases" to IPs, provide other config info



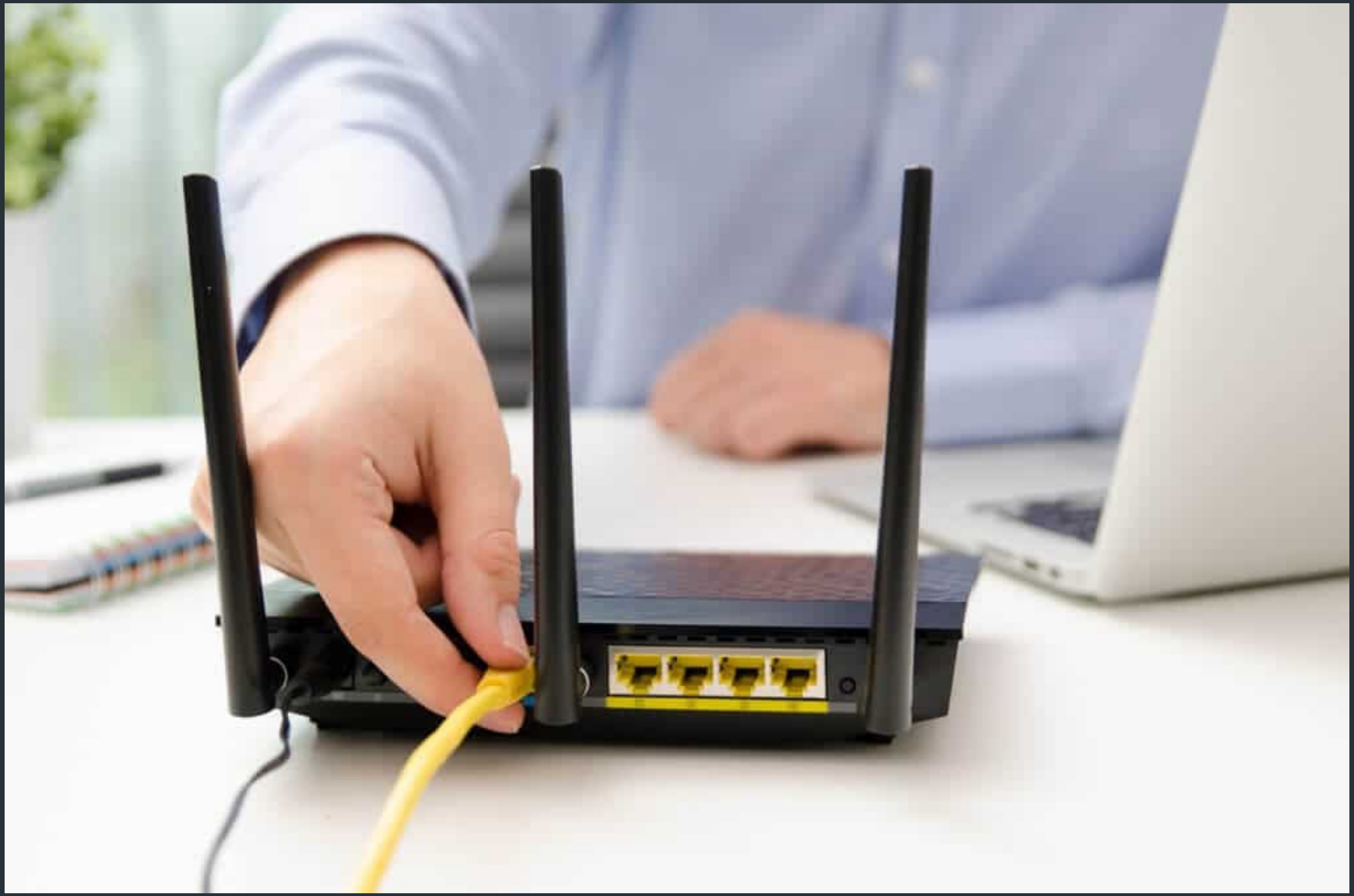
=> Again, host needs to use broadcast address. Why?
=> Problem?

A home router

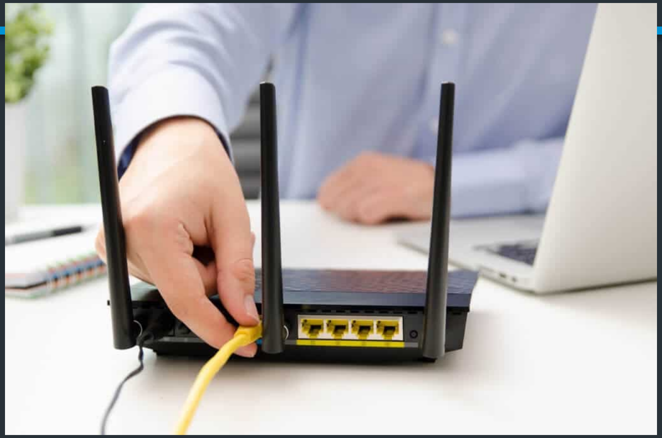
What's in this thing?



Story time



About those home routers…



You get just one IP from your ISP…

=> Need to share IP among many devices on the same network!

Common to create a "private" IP range used within local network

=> Routers need to do extra work to share public IP among private IPs

=> Network Address Translation (NAT)

(A form of connection multiplexing)

Private IPs (RFC1918)

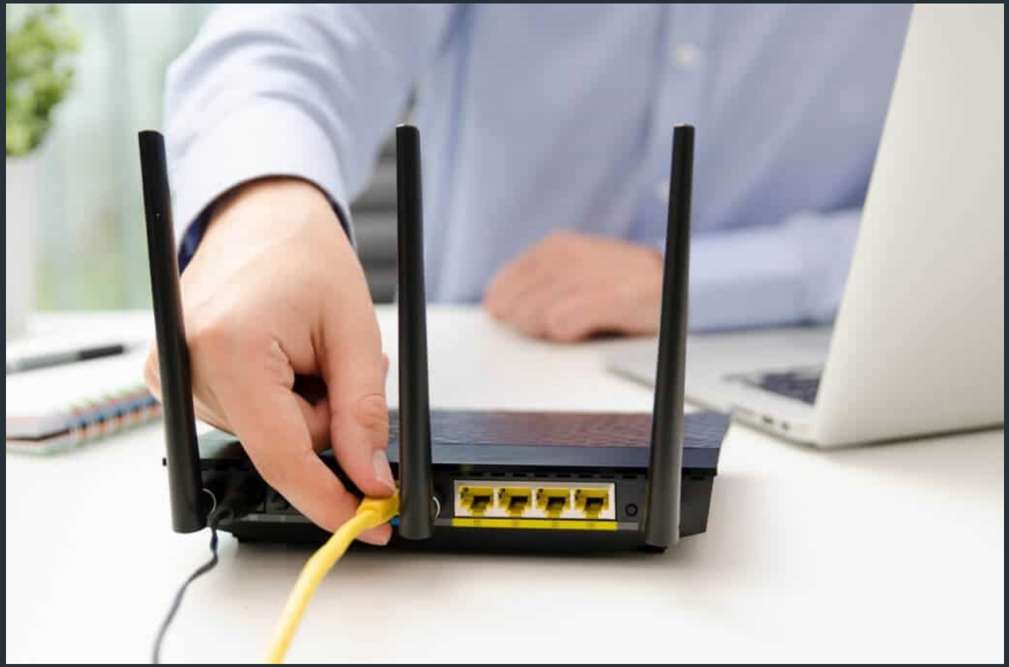
Some IP ranges are reserved:

Prefix	Use
127.0.0.0/8	"Loopback" address—always for current host
10.0.0.0/8	Reserved for private internal networks (RFC1918)
192.168.0.0/16	
172.16.0.0/12	

- Many networks will use these blocks internally
- These IPs should never be routed over the Internet!
  - What would happen if they were?

Network Address Translation

- What happens when hosts need to share an IP address?
- How to map private IP space to public IPs?



Network Address Translation (NAT)

- Despite CIDR, it's still difficult to allocate addresses (2^{32} is *only* 4 billion)
- NAT "hides" entire network behind one address
- Hosts are given *private* addresses
- Routers map outgoing packets to a free address/port
- Router reverse maps incoming packets
- Problems?

NAT Example

Problems with NAT

- Breaks end-to-end connectivity!
- Technically a violation of layering
- Need to do extra work at end hosts to establish end-to-end connection
  - VoIP (Voice/Video conferencing)
  - Games

NAT Traversal

Various methods, depending on the type of NAT

Examples:

- ICE: Interactive Connectivity Establishment (RFC8445)
- STUN: Session Traversal Utilities for NAT (RFC5389)

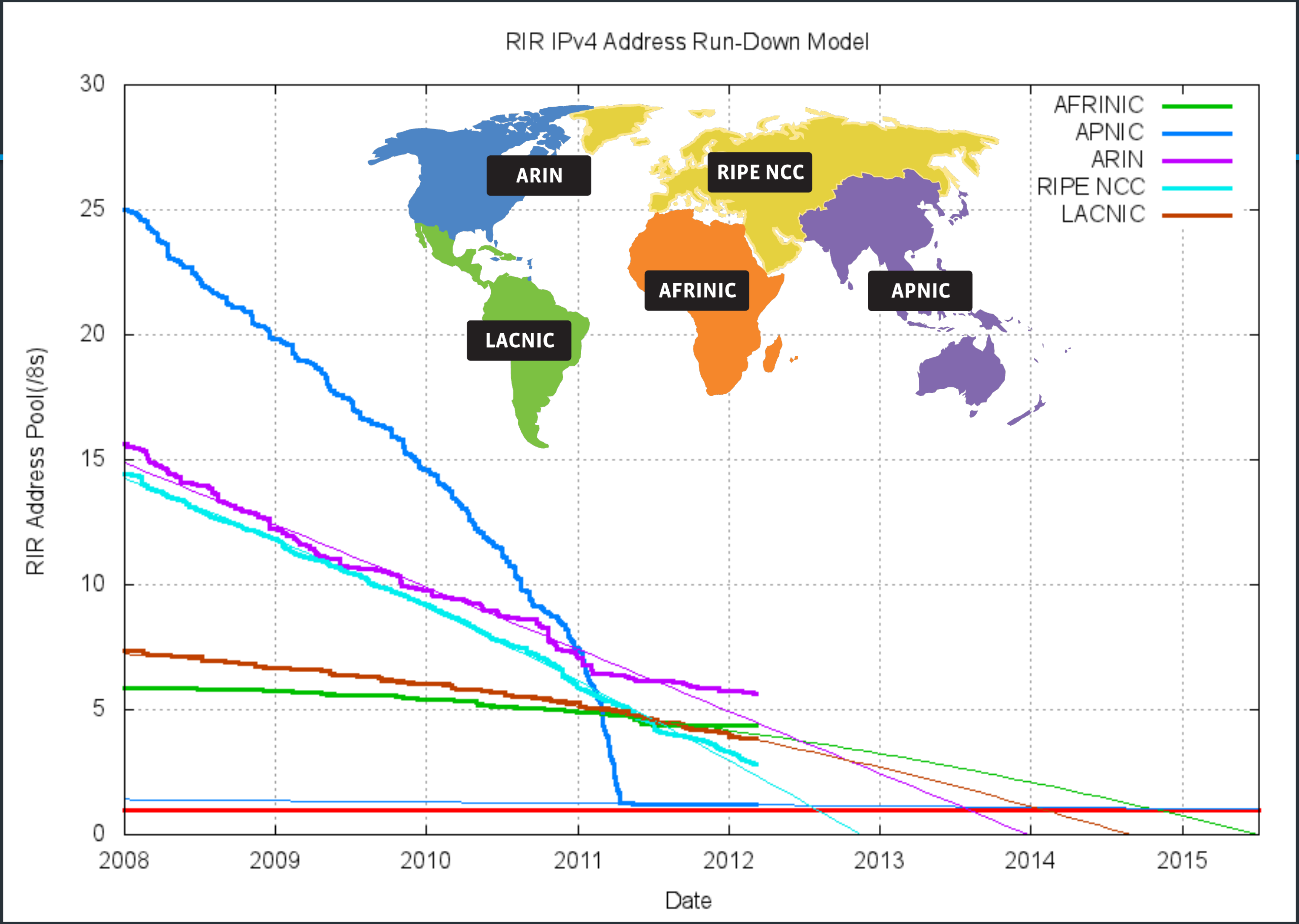
One idea: connect to external server via UDP, it tells you the address/port

IP challenge: Address space exhaustion

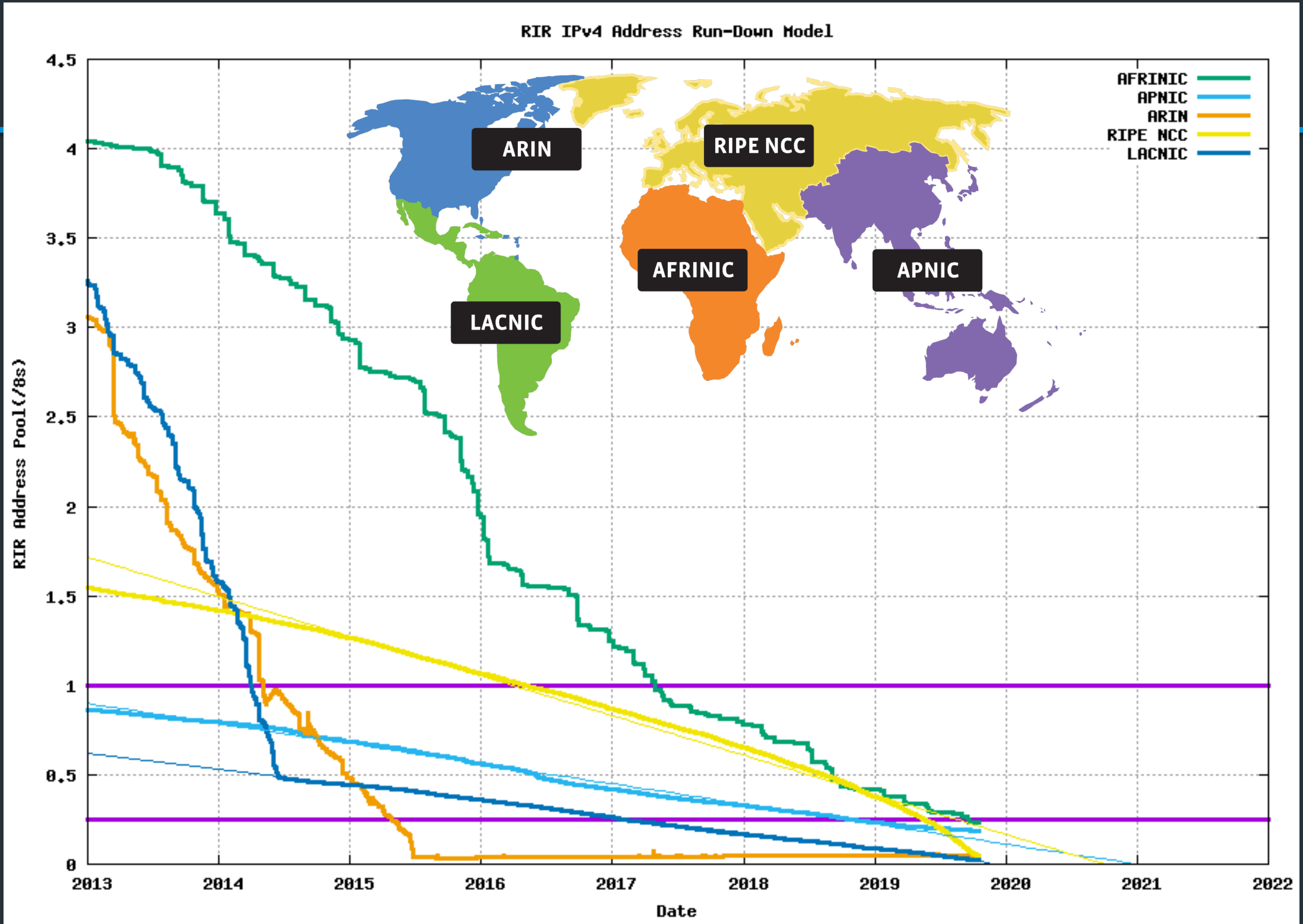
- IP version 4: ~4 billion IP addresses
  - World population: ~8 billion
  - Est. number of devices on Internet (2021): >10-30 billion

- Since 1990s: various tricks
  - Smarter allocations by registrars
  - Address sharing: Network Address Translation (NAT)
  - DHCP
  - Reclaiming unused space

- Long term solution: IP version 6

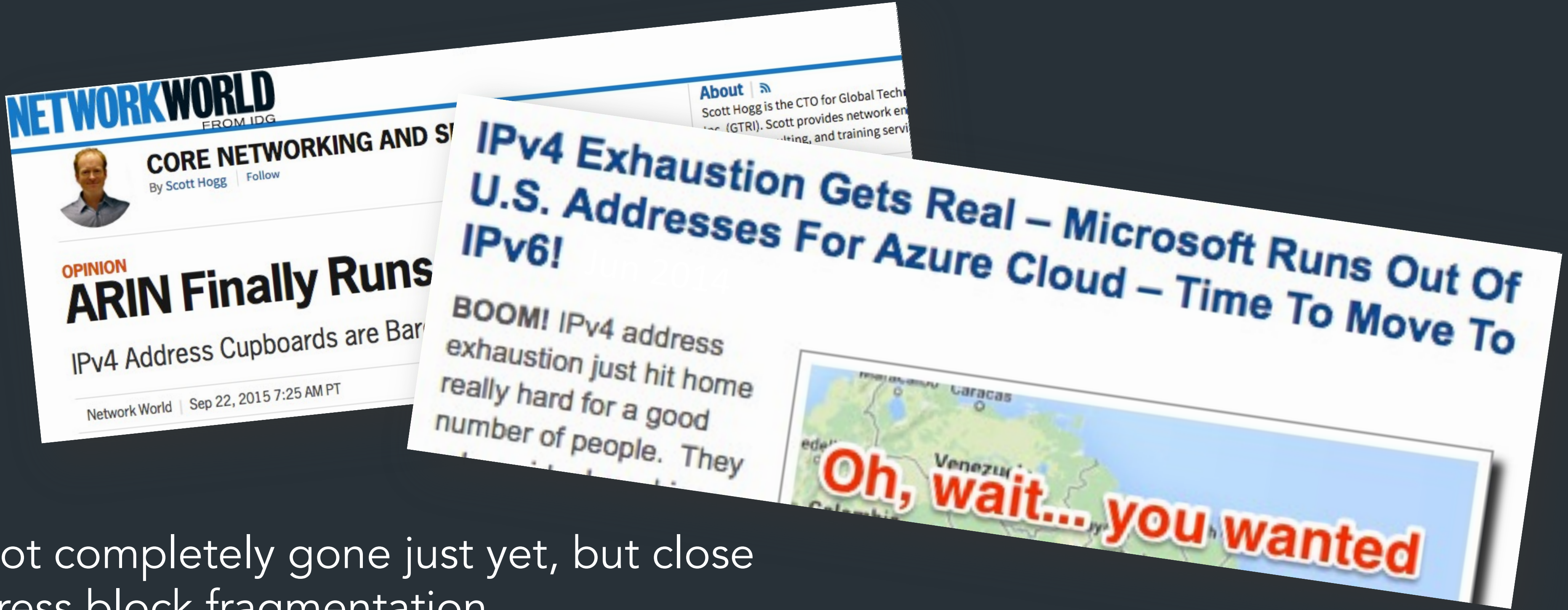


Source: potaroo.net/tools/ipv4



Source: potaroo.net/tools/ipv4

So what happened when we ran out of IPv4 addresses?


NETWORKWORLD FROM IDG
CORE NETWORKING AND S
By Scott Hogg | Follow
About
Scott Hogg is the CTO for Global Tech
(GTRI). Scott provides network en
and training servi
OPINION
ARIN Finally Runs
IPv4 Address Cupboards are Bar
Network World | Sep 22, 2015 7:25 AM PT

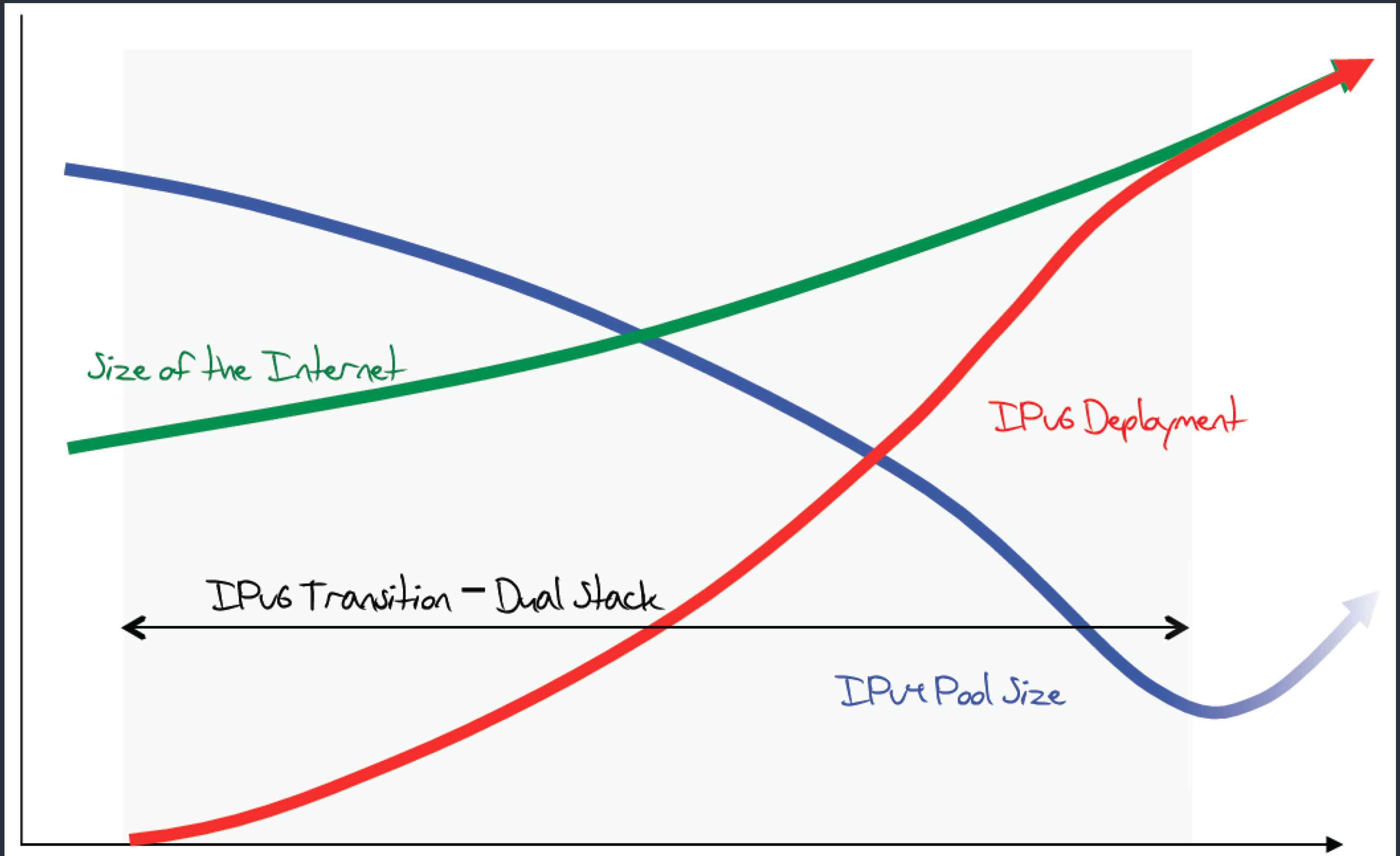
IPv4 Exhaustion Gets Real – Microsoft Runs Out Of U.S. Addresses For Azure Cloud – Time To Move To IPv6!
BOOM! IPv4 address exhaustion just hit home really hard for a good number of people. They
Caracas
Venezu
Oh, wait... you wanted

- It's not completely gone just yet, but close
- Address block fragmentation
  - Secondary market for IPv4
  - E.g., in 2011 Microsoft bought >600K US IPv4 addresses for $7.5M
- NATs galore
  - Home NATs, carrier-grade NATs

IPv6

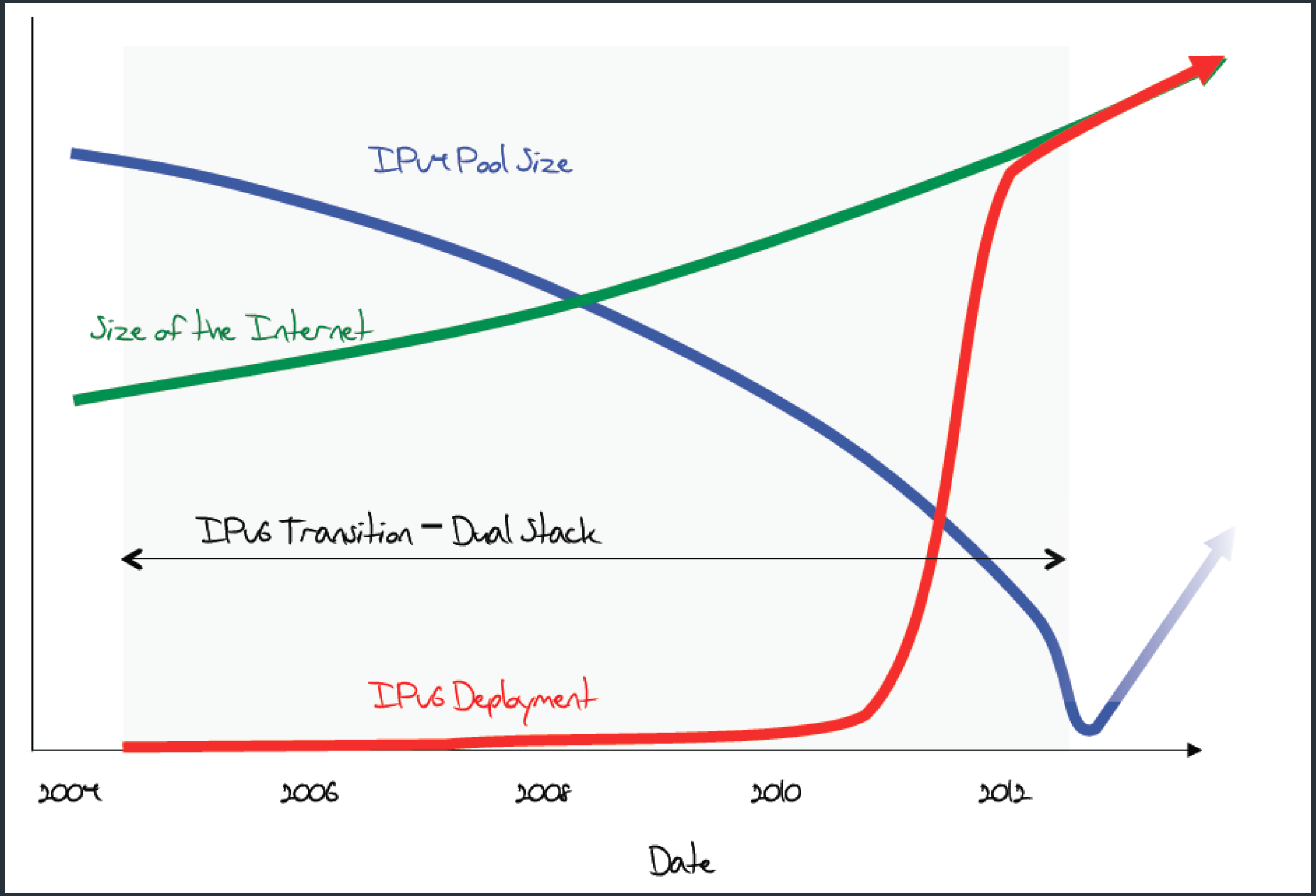
- Main motivation: IPv4 address exhaustion
- Initial idea: larger address space
- Need new packet format:
  - REALLY expensive to upgrade all infrastructure!
  - While at it, why don't we fix a bunch of things in IPv4?
- Work started in 1994, basic protocol published in 1998

The original expected plan

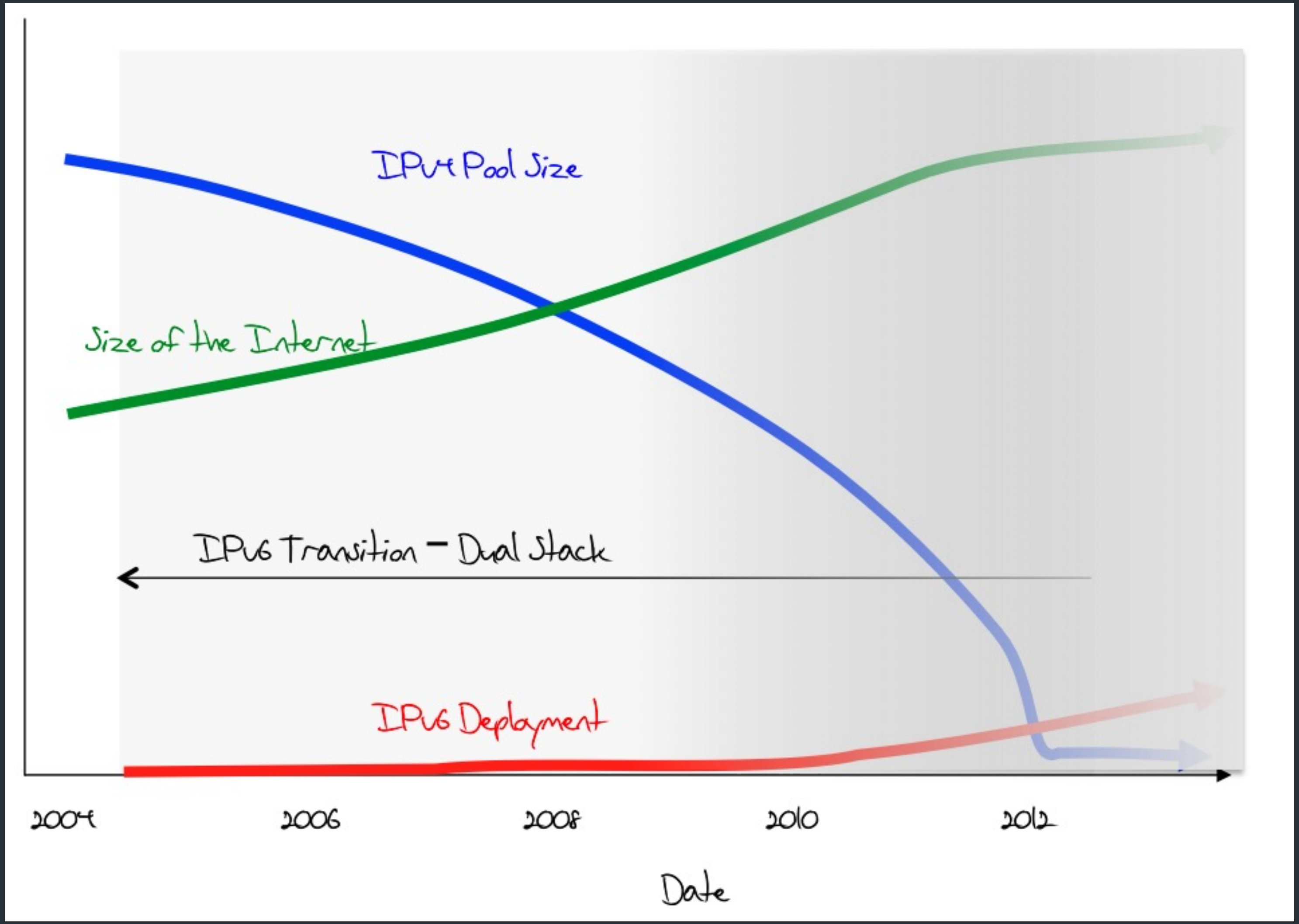


From: http://www.potaroo.net/ispcol/2012-08/EndPt2.html

The plan in 2011



What was happening (late 2012)



June 6th, 2012

Transition is not painless

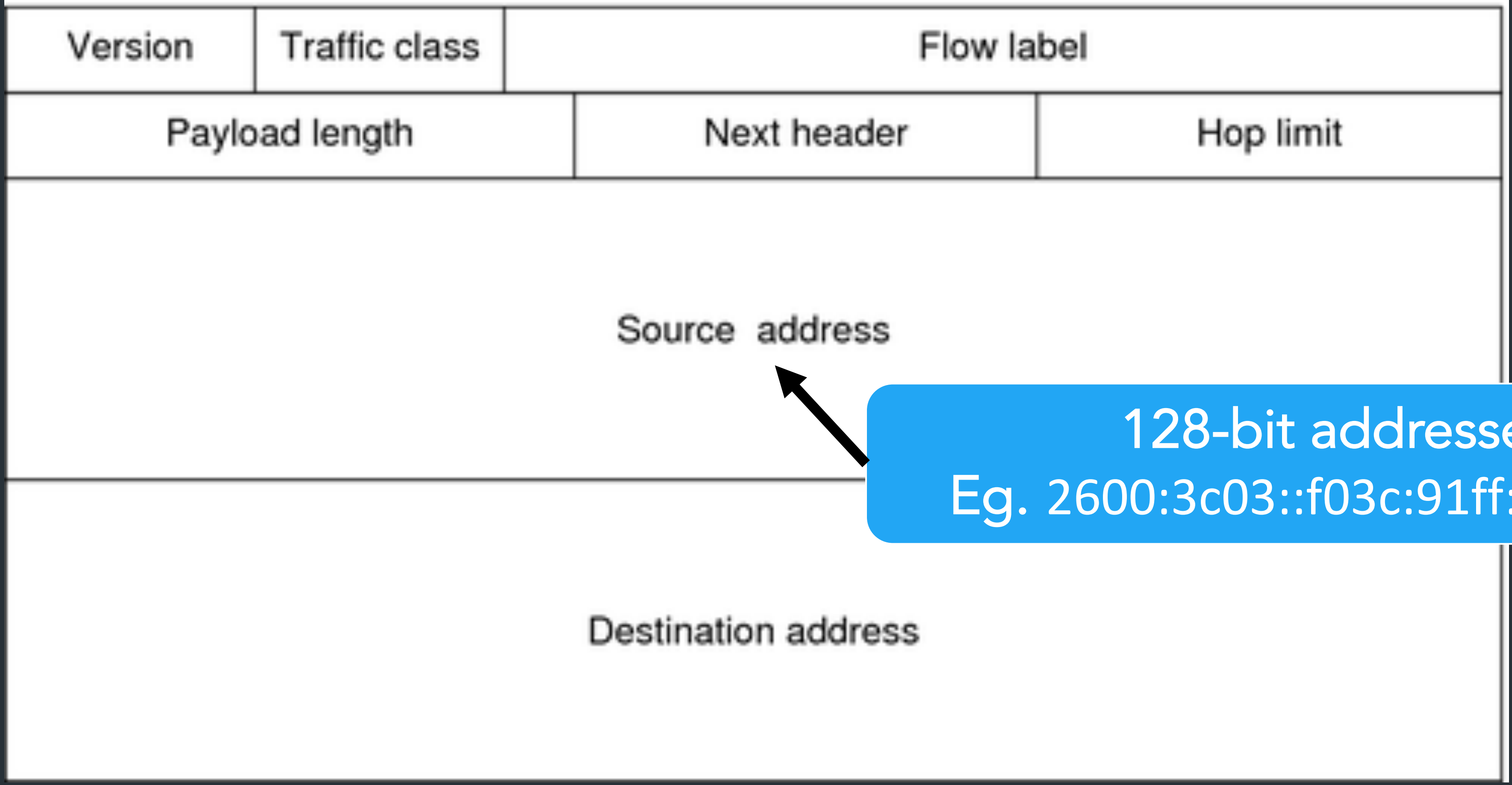
From http://www.internetsociety.org/deploy360/ipv6/ :

You may want to begin with our "**Where Do I Start?**" page where we have guides for:

- **Network operators**
- **Developers**
- **Content providers / website owners**
- **Enterprise customers**
- **Domain name registrars**
- **Consumer electronics vendors**
- **Internet exchange point (IXP) operators**

- Why do each of these parties have to do something?

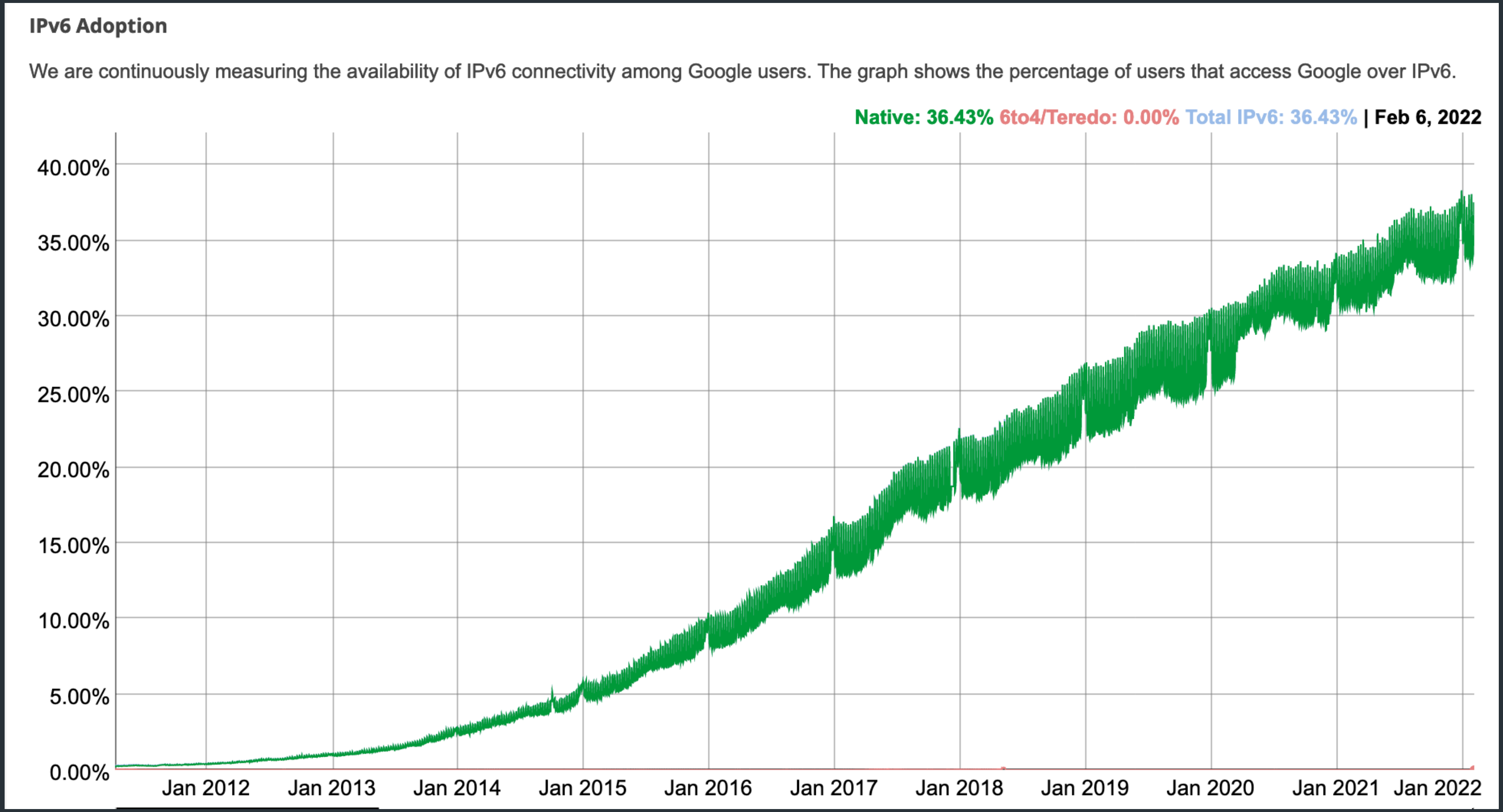
IP version 6



128-bit addresses!
Eg. 2600:3c03::f03c:91ff:fe6e:e3e1

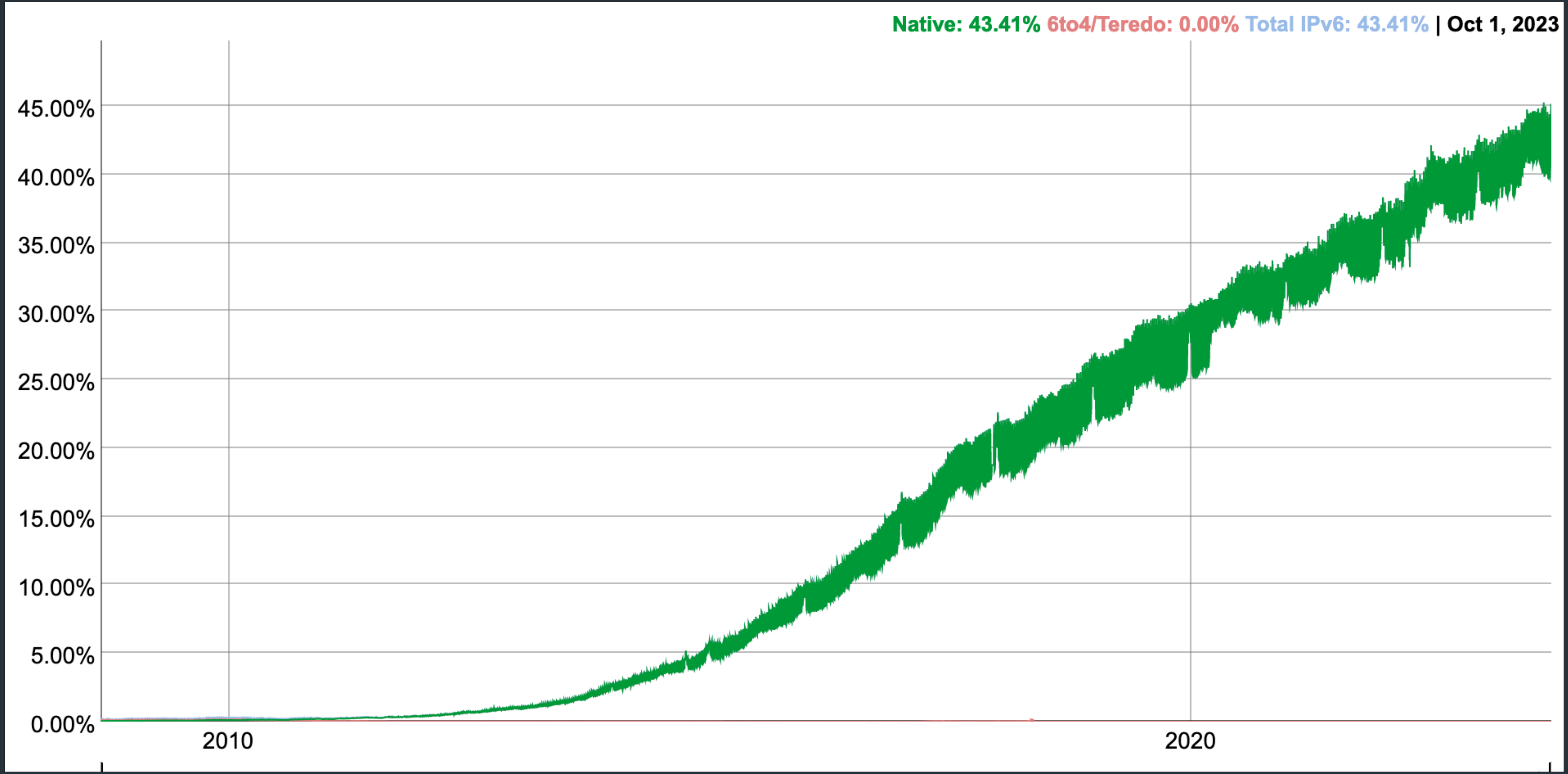
IPv6 Adoption

At Google:

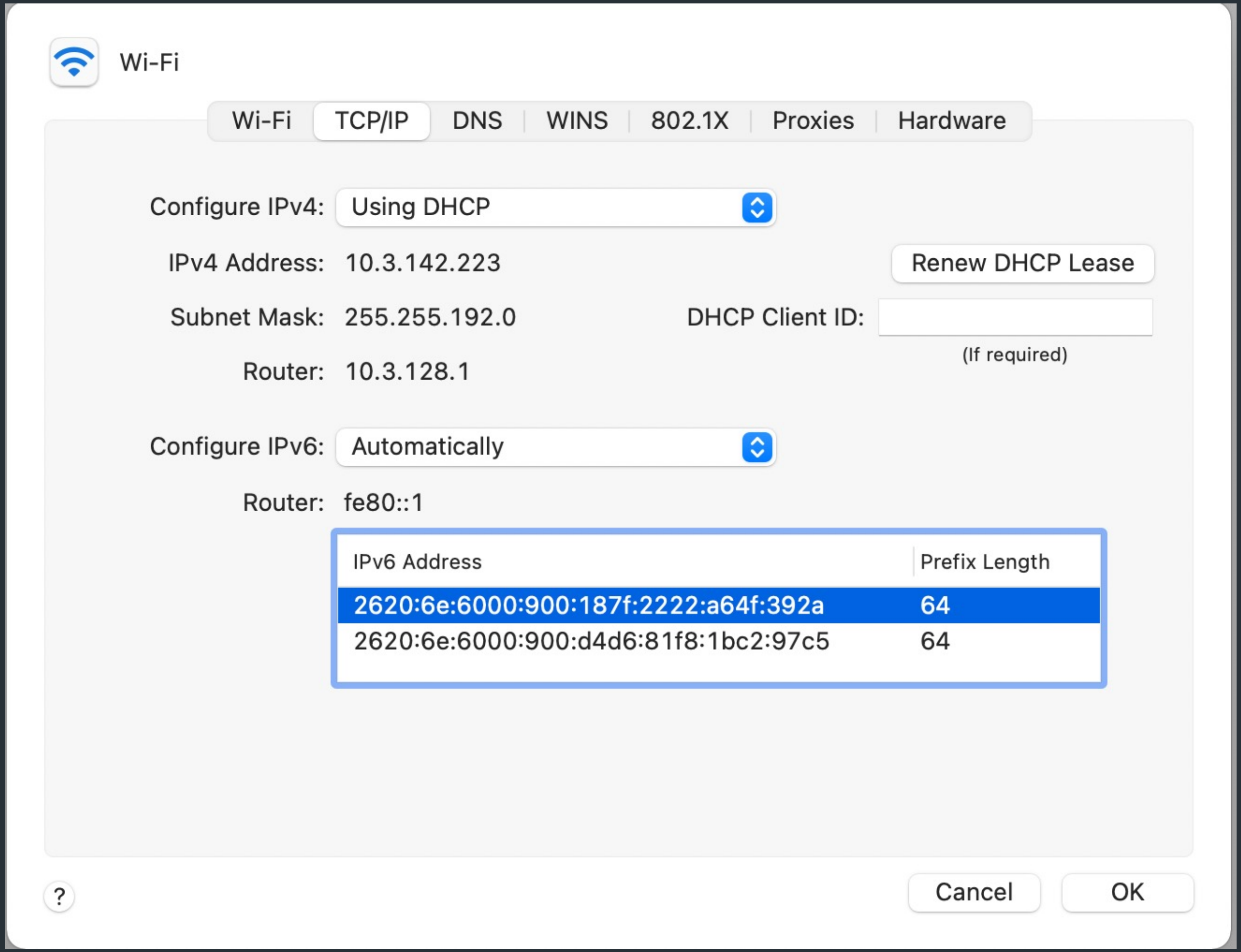


IPv6 Adoption

At Google:



At Brown


Wi-Fi
Wi-Fi
TCP/IP
DNS
WINS
802.1X
Proxies
Hardware
Configure IPv4:
Using DHCP
IPv4 Address:
10.3.142.223
Renew DHCP Lease
Subnet Mask:
255.255.192.0
DHCP Client ID:
(If required)
Router:
10.3.128.1
Configure IPv6:
Automatically
Router:
fe80::1
IPv6 Address
Prefix Length
2620:6e:6000:900:187f:2222:a64f:392a
64
2620:6e:6000:900:d4d6:81f8:1bc2:97c5
64
?
Cancel
OK

IPv6 Key Features

- 128-bit addresses
- Simplifies basic packet format through *extension headers*
  - 40-byte base header (fixed)
  - Make less common fields optional
- Security and Authentication

IPv6 Address Representation

- Groups of 16 bits in hex notation

  47cd:1244:3422:0000:0000:fef4:43ea:0001

- Two rules:
  - Leading 0's in each 16-bit group can be omitted

    47cd:1244:3422:0:0:fef4:43ea:1

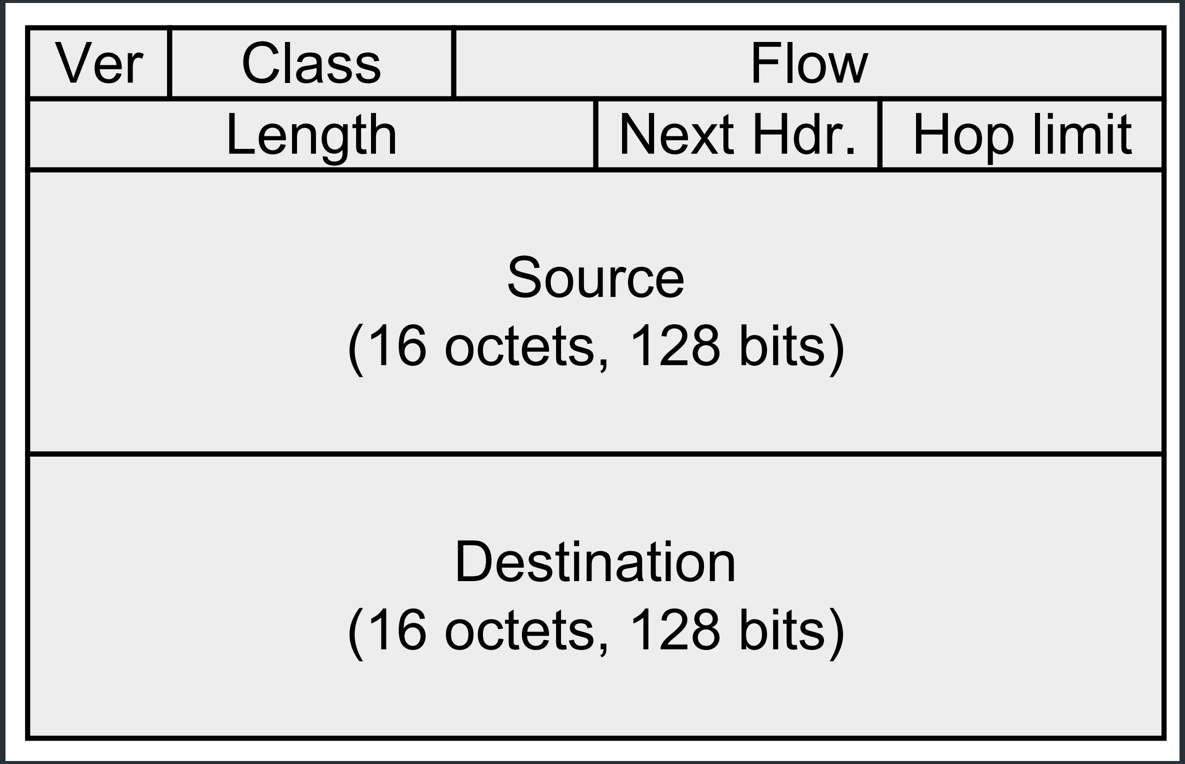
  - One contiguous group of 0's can be compacted

    47cd:1244:3422::fef4:43ea:1

IPv6 Addresses

- Break 128 bits into 64-bit network and 64-bit interface
  - Makes autoconfiguration easy: interface part can be derived from Ethernet address, for example
- Types of addresses
  - All 0's: unspecified
  - 000…1: loopback
  - ff/8: multicast
  - fe8/10: link local unicast
  - fec/10: site local unicast
  - All else: global unicast

IPv6 Header


Ver
Class
Flow
Length
Next Hdr.
Hop limit
Source
(16 octets, 128 bits)
Destination
(16 octets, 128 bits)

IPv6 Header Fields

- Version: 4 bits, 6
- Class: 8 bits, like TOS in IPv4
- Flow: 20 bits, identifies a *flow*
- Length: 16 bits, datagram length
- Next Header, 8 bits: …
- Hop Limit: 8 bits, like TTL in IPv4
- Addresses: 128 bits
- What's missing?
  - No options, no fragmentation flags, *no checksum*

Design Philosophy

- Simplify handling
  - New option mechanism (fixed size header)
  - No more header length field
- Do less work at the network (why?)
  - No fragmentation
  - No checksum
- General flow label
  - No semantics specified
  - Allows for more flexibility
- Still no accountability

With some content from Scott Shenker

Interoperability

- RFC 4038
  - Every IPv4 address has an associated IPv6 address (mapped)
  - Networking stack translates appropriately depending on other end
  - Simply prefix 32-bit IPv4 address with 80 bits of 0 and 16 bits of 1:
  - E.g., ::FFFF:128.148.32.2
- Two IPv6 endpoints must have IPv6 stacks
- Transit network:
  - v6 – v6 – v6 : ✓
  - v4 – v4 – v4 : ✓
  - v4 – v6 – v4 : ✓
  - v6 – v4 – v6 : X!!

Example Next Header Values

- 0: Hop by hop header
- 1: ICMPv4
- 4: IPv4
- 6:TCP
- 17: UDP
- 41: IPv6
- 43: Routing Header
- 44: Fragmentation Header
- 58: ICMPv6

Current State

- IPv6 Deployment picking up
- Most end hosts have dual stacks today (Windows, Mac OSX, Linux, *BSD, Solaris)
- Requires all parties to work!
  - Servers, Clients, DNS, ISPs, all routers
- IPv4 and IPv6 will coexist for a long time

Coming Up

- Routing: how do we fill the routing tables?
  - Intra-domain routing: Tuesday, 10/4
  - Inter-domain routing: Thursday, 10/6

Example

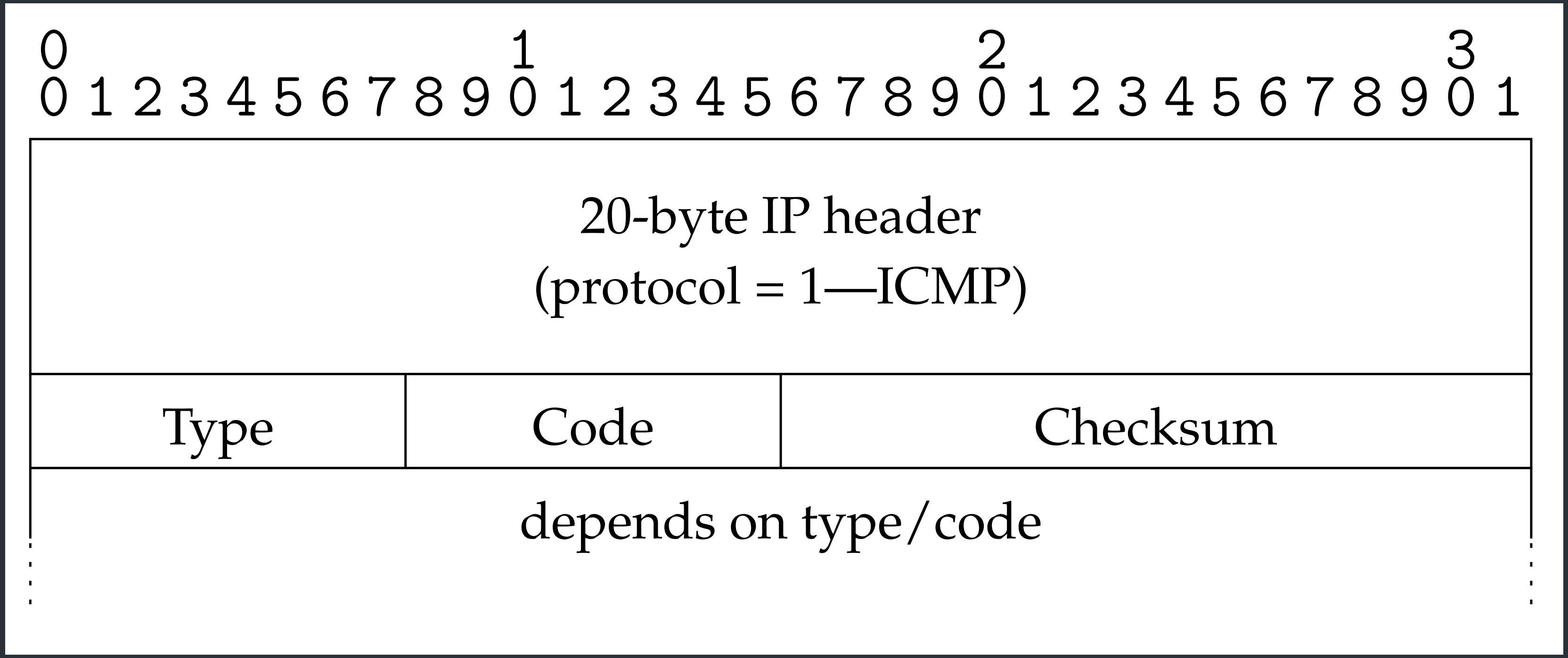
```
# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.17.44.1              ether   00:12:80:01:34:55   C                     eth0
172.17.44.25             ether   10:dd:b1:89:d5:f3   C                     eth0
172.17.44.6              ether   b8:27:eb:55:c3:45   C                     eth0
172.17.44.5              ether   00:1b:21:22:e0:22   C                     eth0
```

```
# ip route
127.0.0.0/8 via 127.0.0.1 dev lo
172.17.44.0/24 dev enp7s0  proto kernel  scope link  src 172.17.44.22  metric 204
default via 172.17.44.1 dev eth0 src 172.17.44.22  metric 204
```

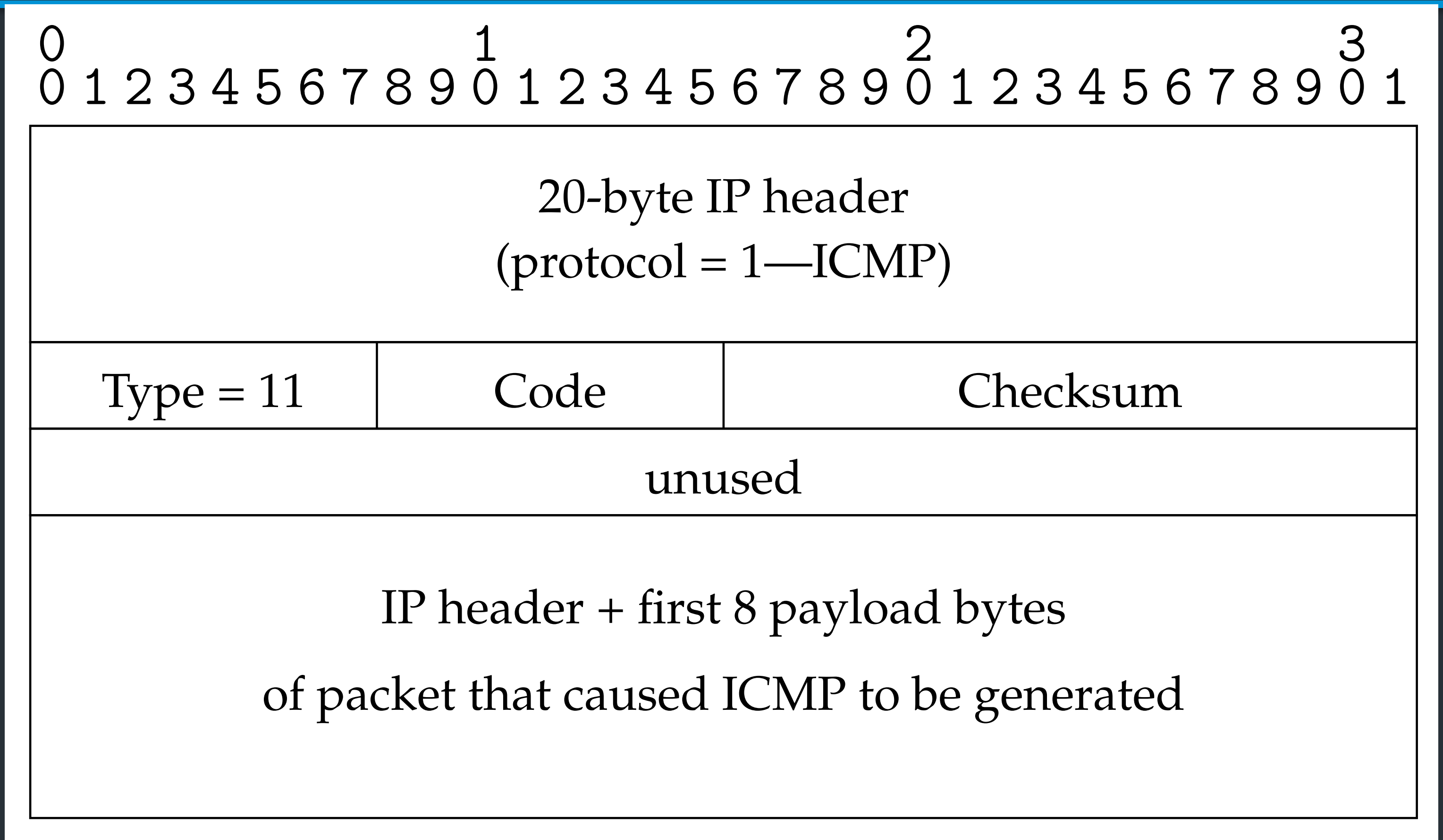
Internet Control Message Protocol (ICMP)

- Echo (ping)
- Redirect
- Destination unreachable (protocol, port, or host)
- TTL exceeded
- Checksum failed
- Reassembly failed
- Can't fragment
- Many ICMP messages include part of packet that triggered them
- See http://www.iana.org/assignments/icmp-parameters

ICMP message format



Example: Time Exceeded



- Code usually 0 (TTL exceeded in transit)
- Discussion: traceroute