CSCI-1680
Network Layer:
Wrapup

John Jannotti

Based partly on lecture notes by Jennifer Rexford, Rob Sherwood, David Mazières, Phil Levis, Rodrigo Fonseca

Administrivia

- **Homework 2 will be out tonight**
- **Will be due “fast” (Tuesday)**
  – So we can post solutions before the midterm!
- **Exam on Thursday**
  – All content up to today
  – Questions similar to the homework
  – Book has some exercises, samples on the course web page (from previous years)
- **Tech talk: Pure Storage, Monday 8pm.**

Today: IP Wrap-up

- **IP Service models**
  - Unicast, Broadcast, Anycast, Multicast
- **IPv6**
  - Tunnels

Different IP Service Models

- **Broadcast: send a packet to *all* nodes in some subnet. “One to all”**
  - 255.255.255.255 : all hosts within a subnet, *never* forwarded by a router
  - “All ones host part”: broadcast address
    - Host address | (255.255.255.255 & ~subnet mask)
    - E.g.: 128.148.32.143 mask 255.255.255.128
    - ~mask = 0.0.0.127 => Bcast = 128.148.32.255
- **Example use: DHCP**
- **Not present in IPv6**
  - Use multicast to link local all nodes group

Anycast

- **Multiple hosts may share the same IP address**
- **"One to one of many" routing**
- **Example uses: load balancing, nearby servers**
  - DNS Root Servers (e.g. f.root-servers.net)
  - Google Public DNS (8.8.8.8)
  - IPv6 6-to-4 Gateway (192.88.99.1)

Anycast Implementation

- **Anycast addresses are /32s**
- **At the BGP level**
  - Multiple ASs can advertise the same prefixes
  - Normal BGP rules choose one route
- **At the Router level**
  - Router can have multiple entries for the same prefix
  - Can choose among many
- **Each packet can go to a different server**
  - Best for services that are fine with that (connectionless, stateless)

Multicast

- **Send messages to many nodes: “one to many”**
- **Why do that?**
  - Snowcast, Internet Radio, IPTV
  - Stock quote information
  - Multi-way chat / video conferencing
  - Multi-player games
- **What’s wrong with sending data to each recipient?**
  - Link stress, especially near origin
  - Have to know address of all destinations

Multicast Service Model

- **Receivers join a multicast group G**
- **Senders send packets to address G**
- **Network routes and delivers packets to all members of G**
- **Multicast addresses: class D (start 1110)**

  224.x.x.x to 229.x.x.x

  – 28 bits left for group address

LAN Multicast

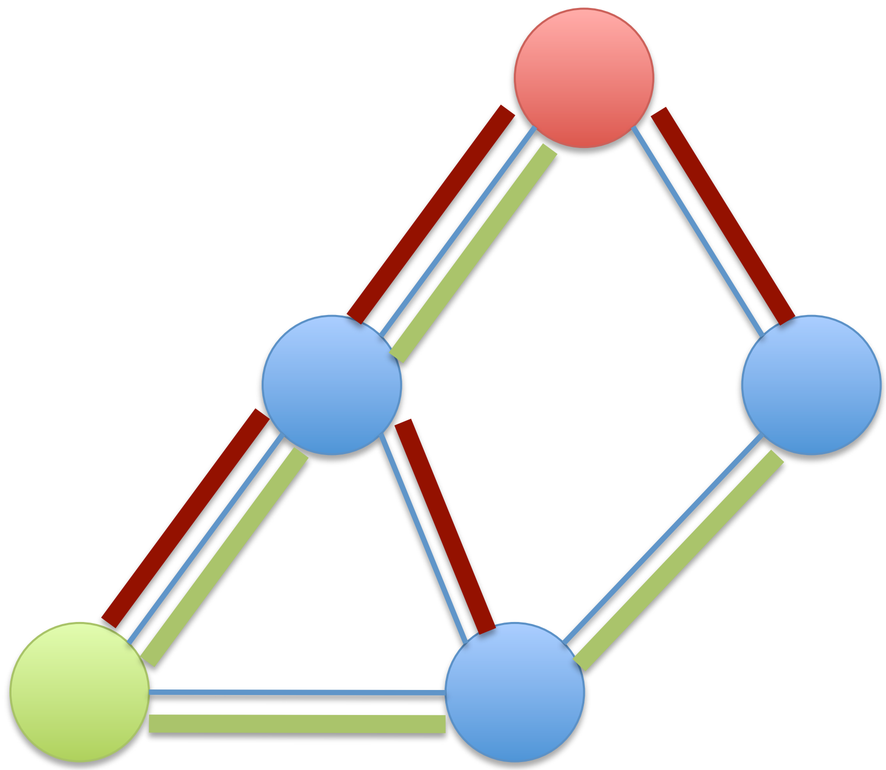
- **Easy on a shared medium**
- **Ethernet multicast address range:**
  - 01:00:5E:00:00:00 to 01:00:5E:7f:ff:ff
- **Set low 23 bits of Ethernet address to low bits of IP address**
  - (Small problem: 28-bit group address -> 23 bits)

How about on the Internet?

Use Distribution Trees

- **Source-specific trees:**
  - Spanning tree over recipients, rooted at each source
  - Best for each source
- **Shared trees:**
  - Single spanning tree among all sources and recipients
  - Hard to find one shared tree that's best for many senders
- **State in routers much larger for source-specific**

Source vs Shared Trees



Building the Tree: Host to Router

- **Nodes tell their local routers about groups they want to join**
  - IGMP, Internet Group Management Protocol (IPv4)
  - MLD, Multicast Listener Discovery (IPv6)
- **Router periodically polls LAN to determine memberships**
  - Hosts are not required to leave explicitly, can stop responding

Building the Tree across networks

- **Routers maintain multicast routing tables**
  - Multicast address -> set of interfaces, or
  - <Source, Multicast address> -> set of interfaces
- **Critical: only include interfaces where there are downstream recipients**

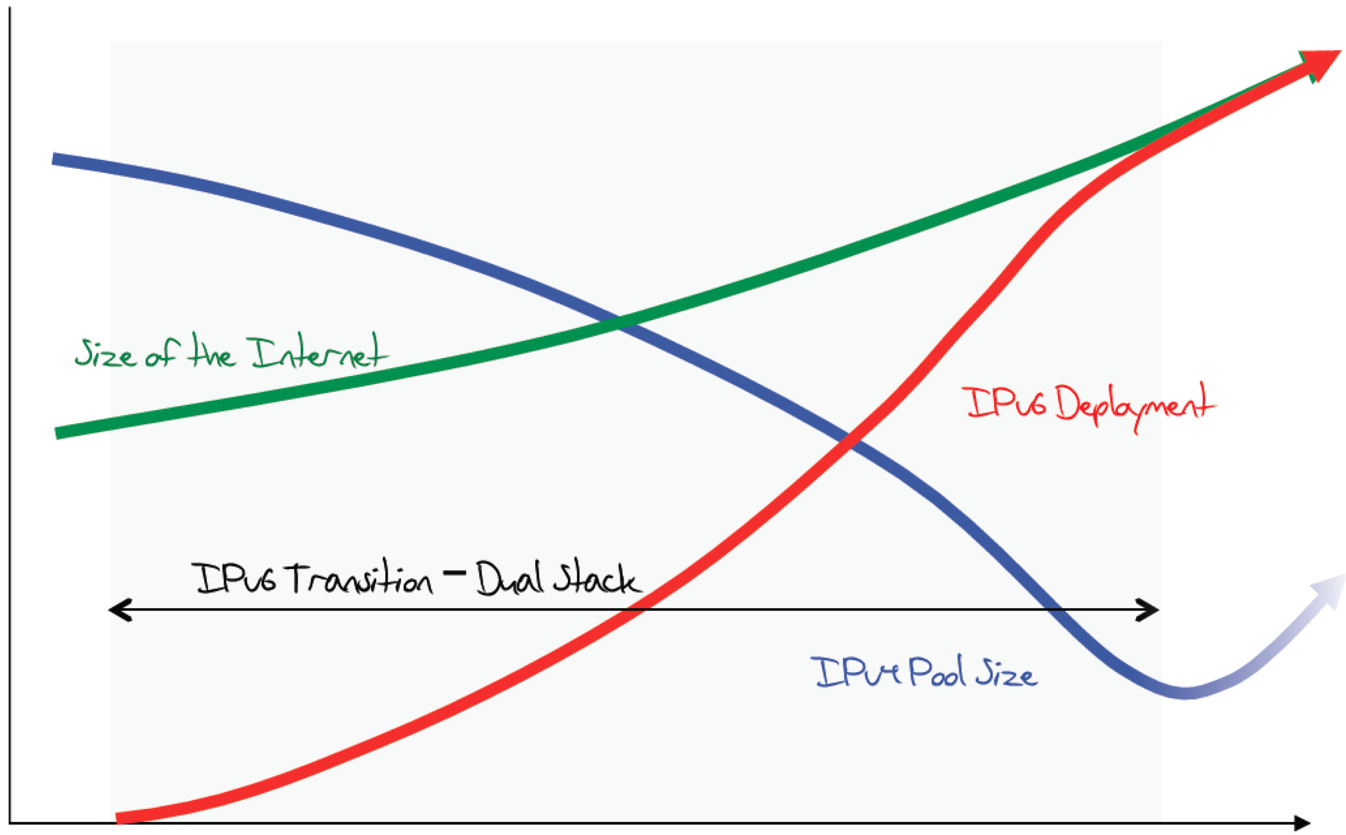
Practical Considerations

- **Multicast protocols end up being quite complex**
- **Introduce a lot of router state**
- **Turned off on most routers**
- **Mostly used within domains**
  - In the department: Ganglia monitoring infrastructure
  - IPTV on campus
- **Alternative: do multicast in higher layers**
  - BitTorrent is an example, sort of.
  - Can’t duplicate packets in routers, but flexibility always wins.

IPv6

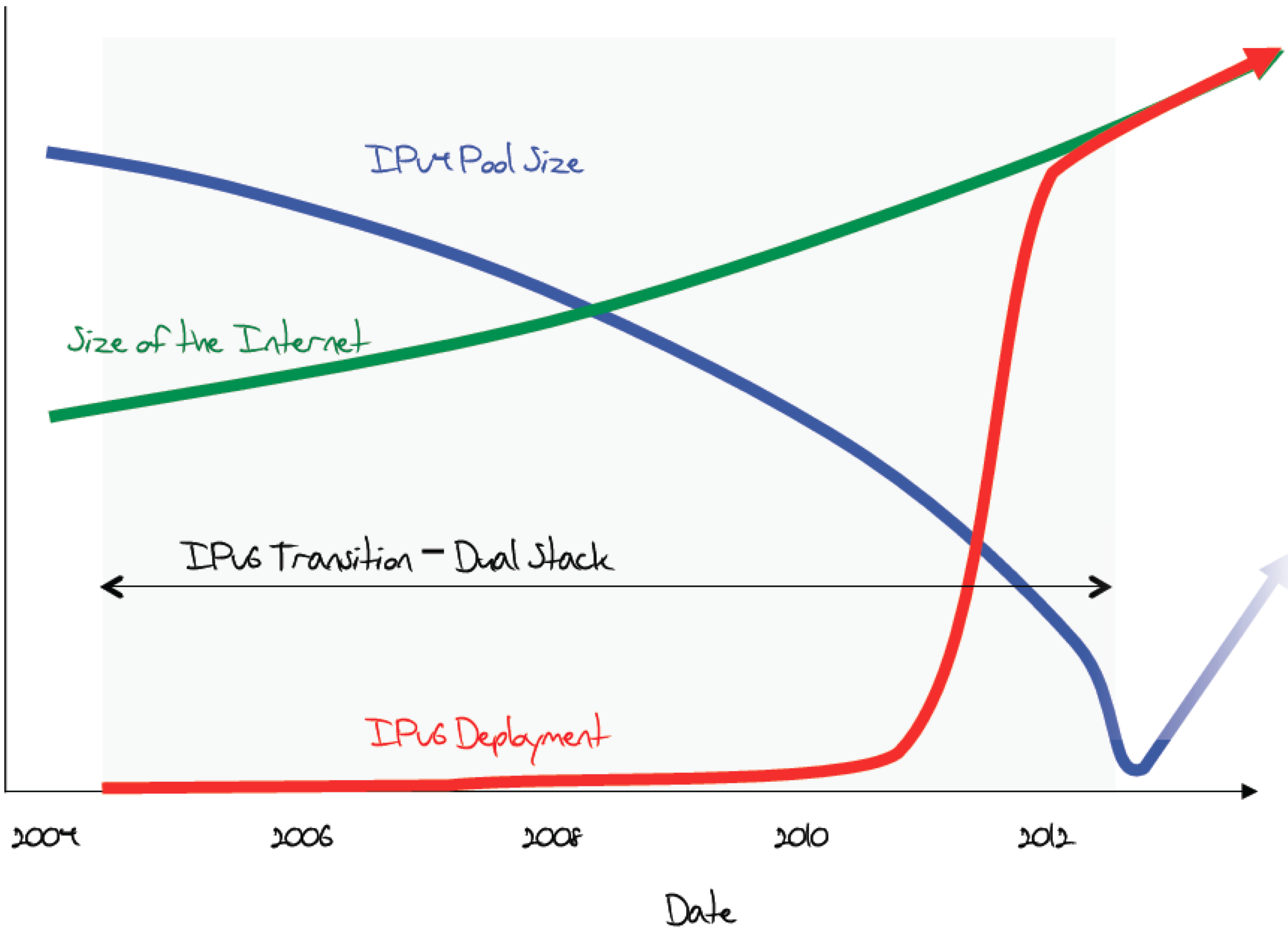
- **Main motivation: IPv4 address exhaustion**
- **Initial idea: larger address space**
- **Need new packet format:**
  - REALLY expensive to upgrade all infrastructure!
  - While at it, why don't we fix a bunch of things in IPv4?
- **Work started in 1994, basic protocol published in 1998**

The original expected plan

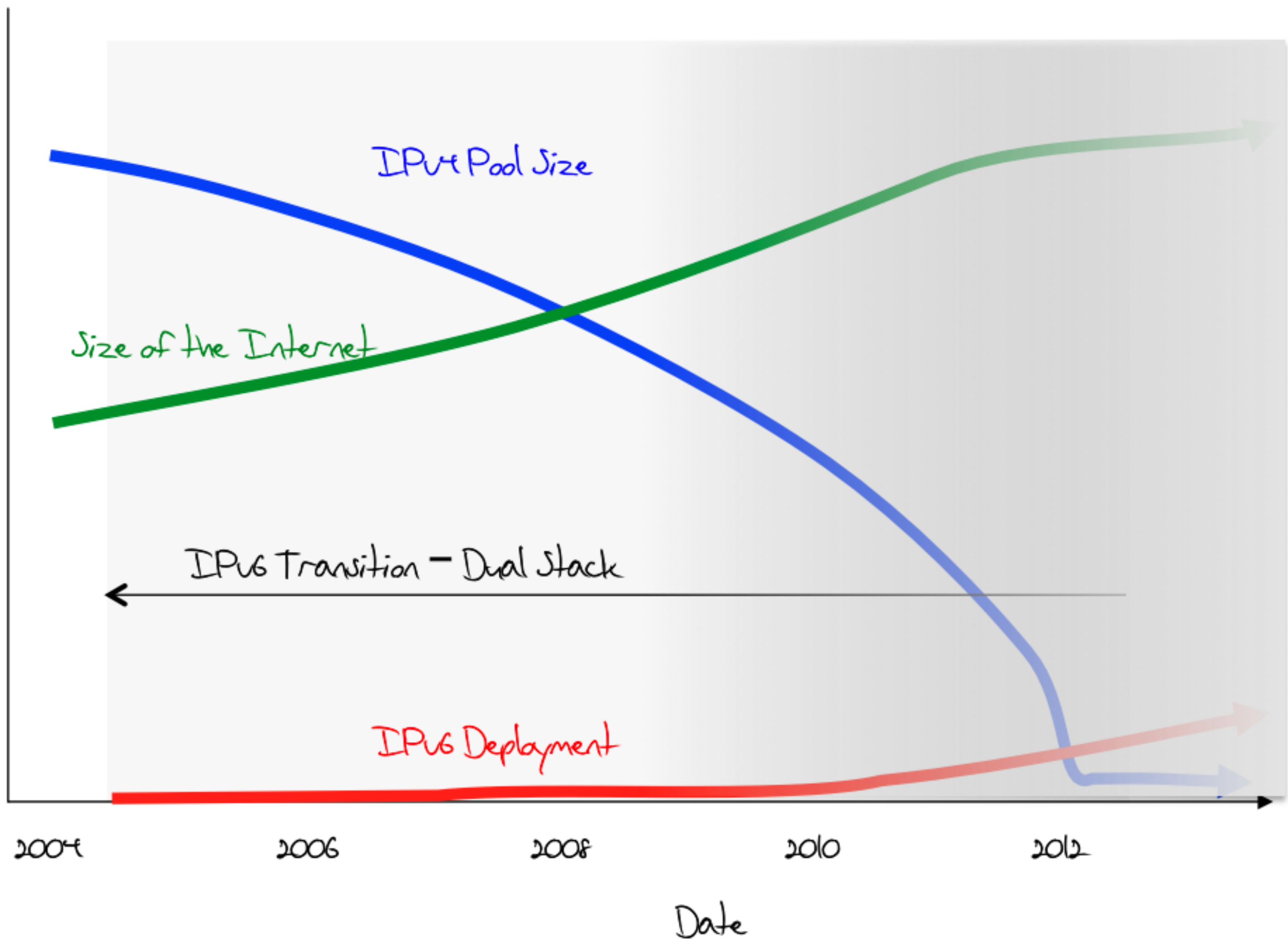


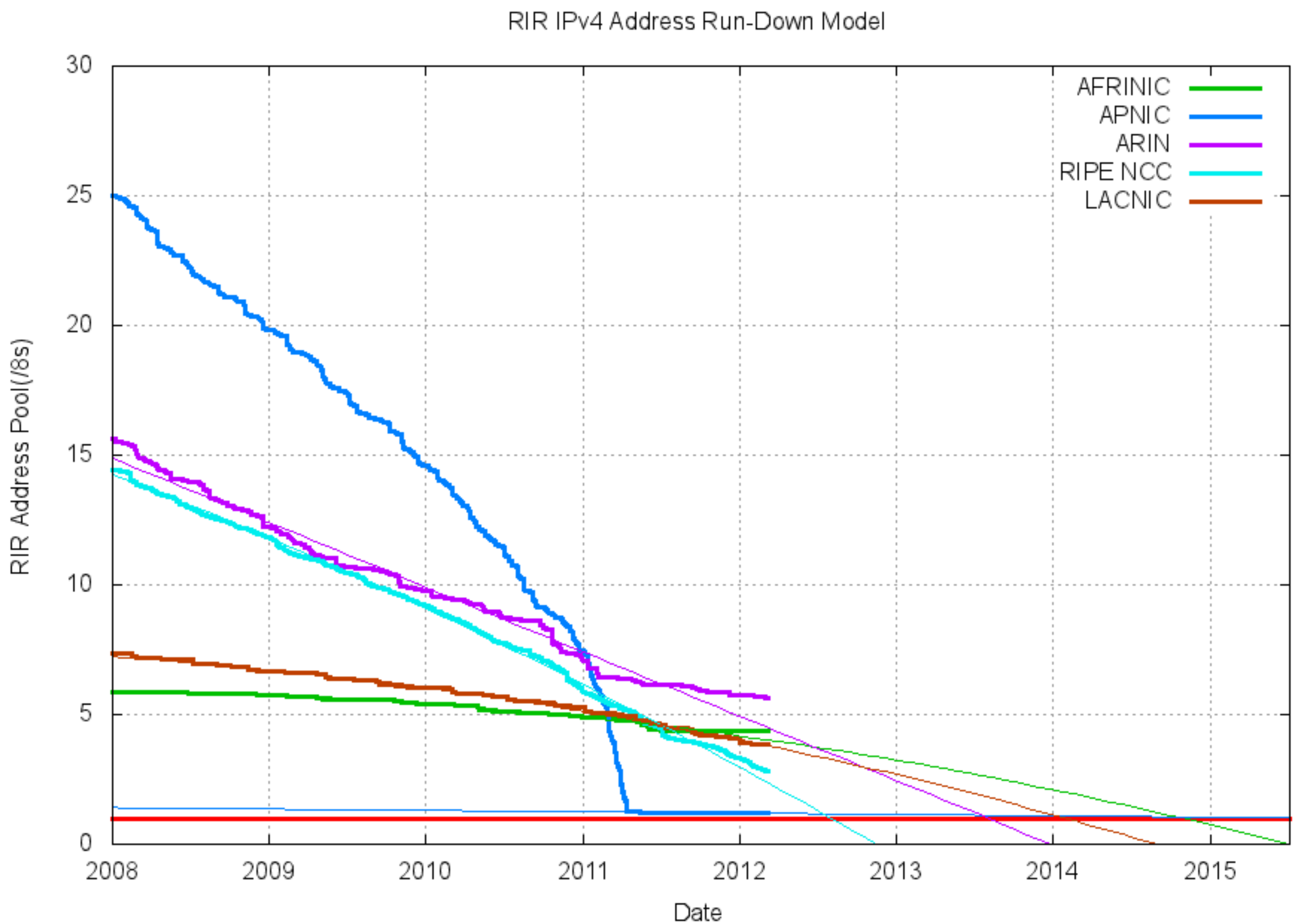
From: http://www.potaroo.net/ispcol/2012-08/EndPt2.html

The plan in 2011

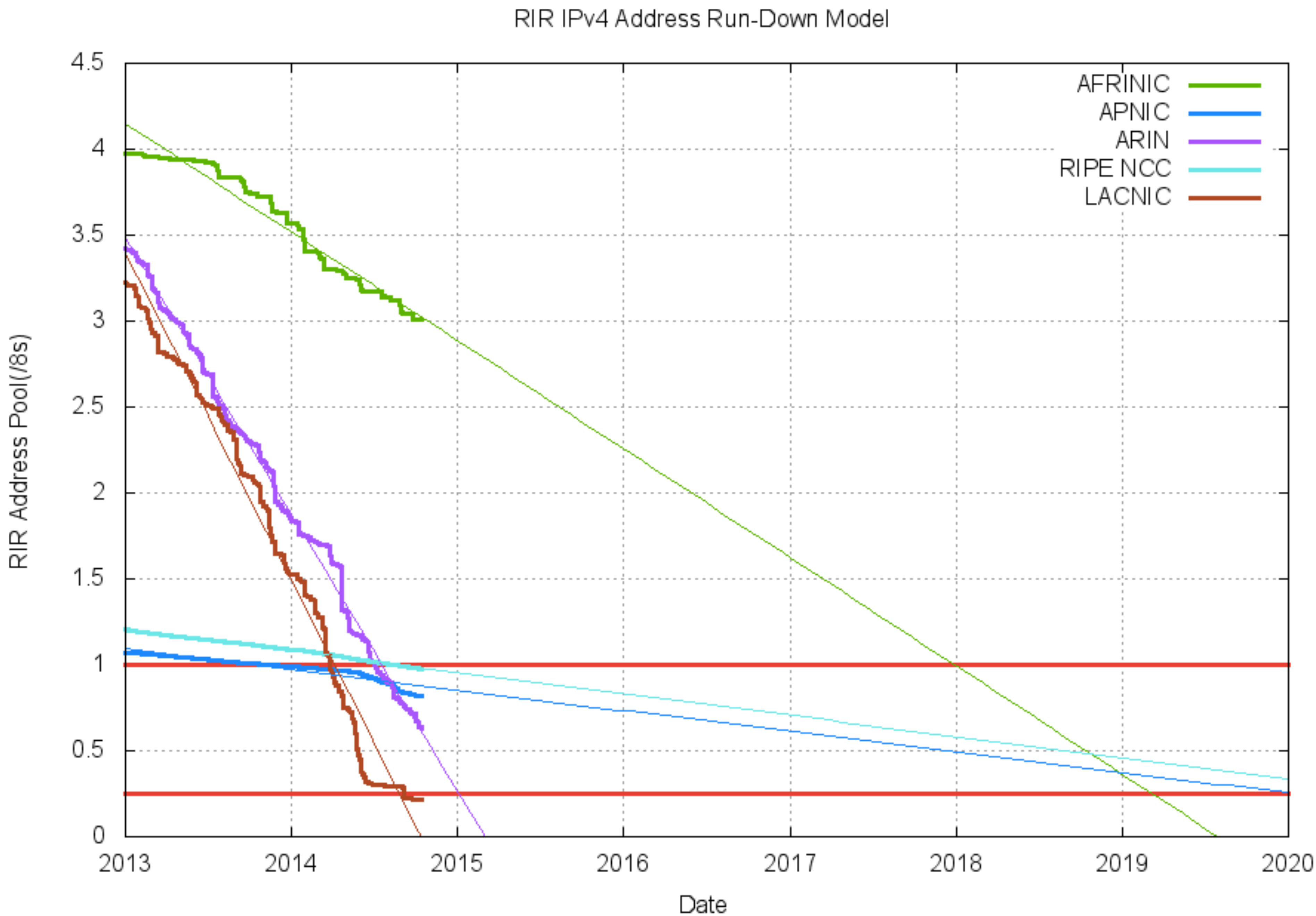


What is really happening



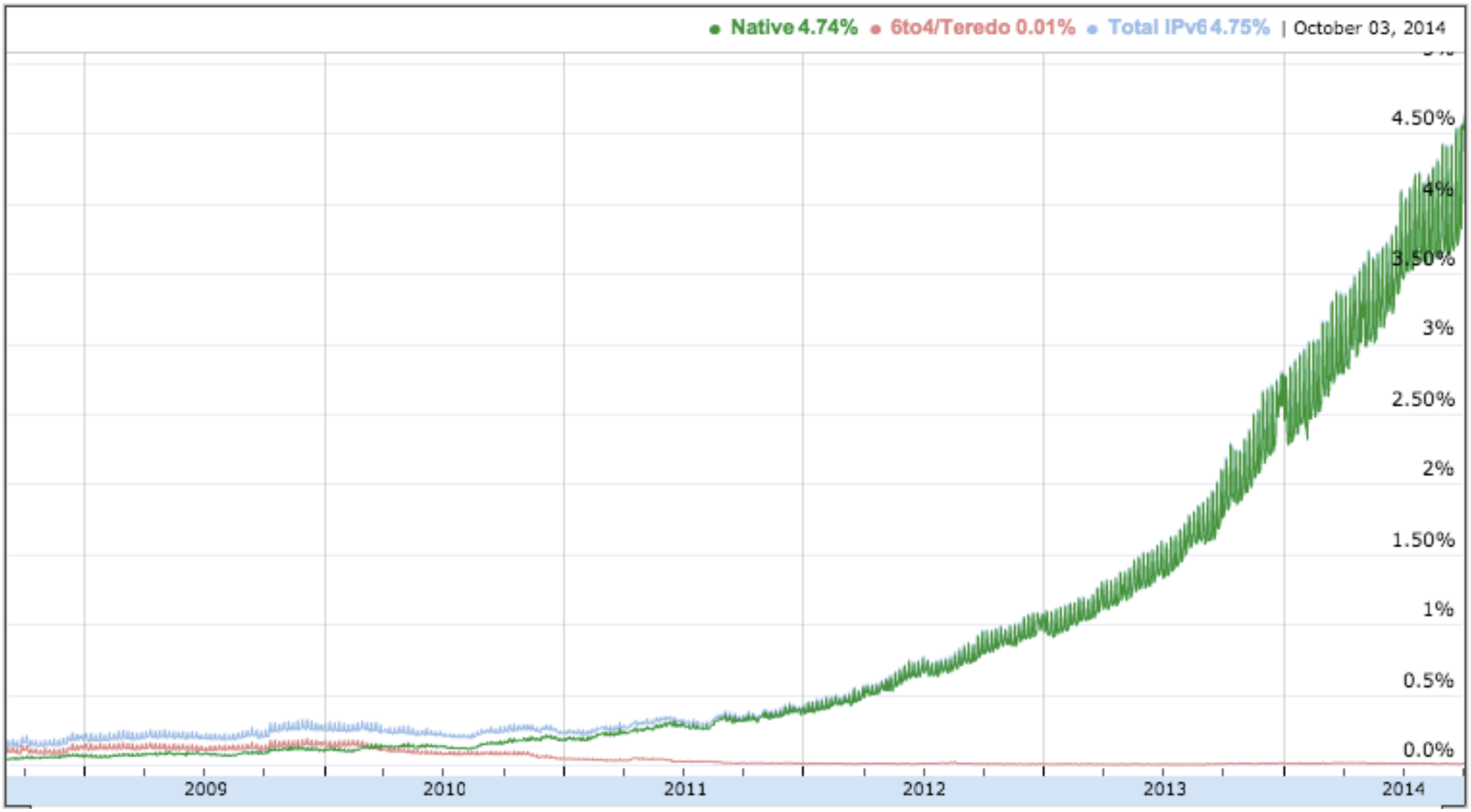


Source: potaroo.net/tools/ipv4



Source: potaroo.net/tools/ipv4

Current Adoption (as seen by Google)



Source: http://www.google.com/ipv6/statistics.html

IPv6 Key Features

- **128-bit addresses**
  - Autoconfiguration
- **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

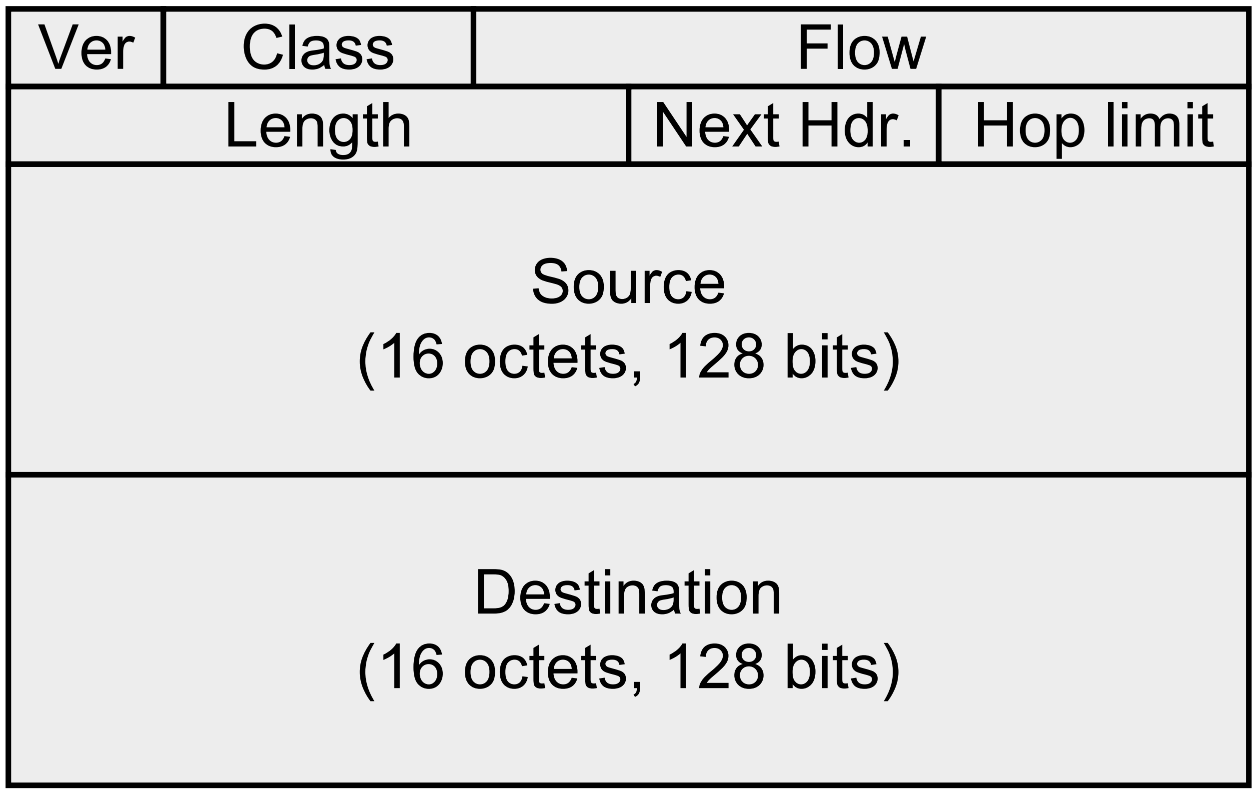
  - One contiguous group of 0's can be compacted

    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**

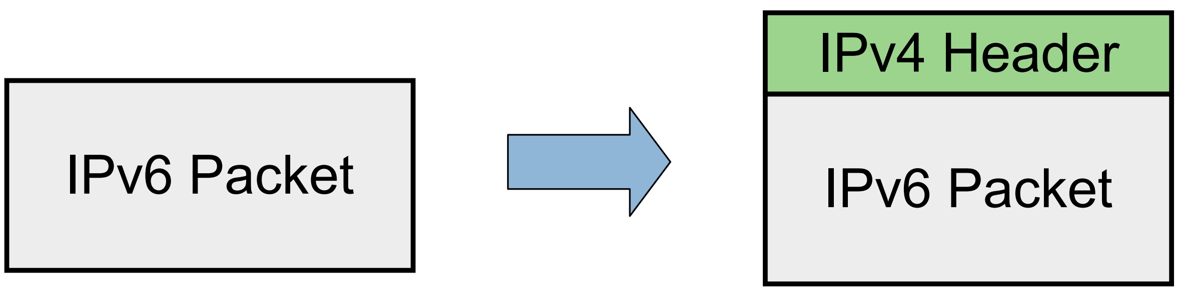
With some content from Scott Shenker

Interoperability

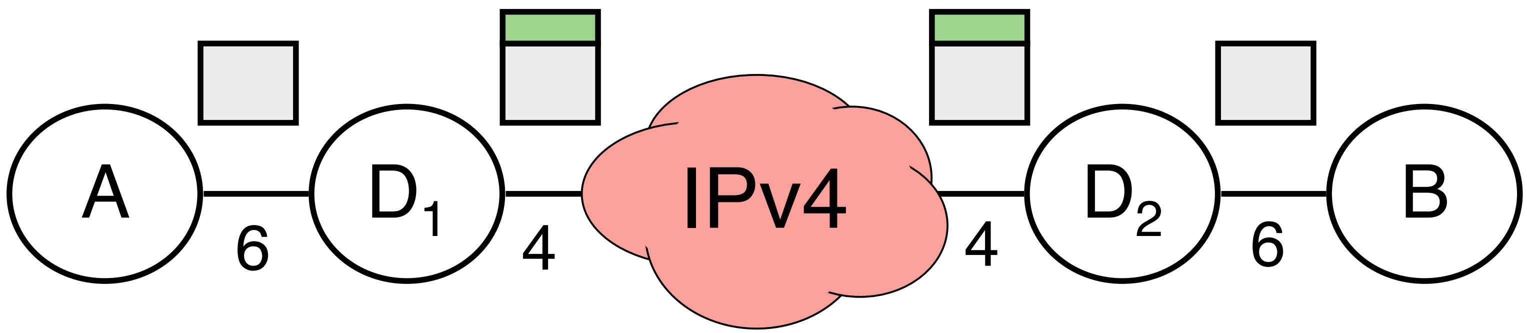
- **RFC 4038**
  - Every IPv4 address has an associated IPv6 address (mapped)
  - Networking stack translates appropriately depending on other end
  - Simply prefix 32-bit IPv4 address with 80 bits of 0 and 16 bits of 1:
  - E.g., ::FFFF:128.148.32.2
- **Two IPv6 endpoints must have IPv6 stacks**
- **Transit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

IP Tunneling

- Encapsulate an IP packet inside another IP packet
- Makes an end-to-end path look like a single IP hop



IPv6 in IPv4 Tunneling



- **Key issues: configuring the tunnels**
  - Determining addresses
  - Determining routes
  - Deploying relays to encapsulate/forward/decapsulate
- **Several proposals, not very successful**
  - 6to4, Teredo, ISATAP
  - E.g., 6to4
    - Deterministic address generation
    - Anycast 192.88.99.1 to find gateway into IPv6 network
    - Drawbacks: voluntary relays, requires public endpoint address

Other uses for tunneling

- **Virtual Private Networks**
- **Use case: access CS network from the outside**
  - Set up an encrypted TCP connection between your computer and Brown's OpenVPN server
  - Configure routes to Brown's internal addresses to go through this connection
- **Can connect two remote sites securely**

Extension Headers

- **Two types: hop-by-hop and end-to-end**
- **Both have a next header byte**
- **Last next header also denotes transport protocol**
- **Destination header: intended for IP endpoint**
  - Fragment header
  - Routing header (loose source routing)
- **Hop-by-hop headers: processed at each hop**
  - Jumbogram: packet is up to 2^{32} bytes long!

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

Fragmentation and MTU

- **Fragmentation is supported only on end hosts!**
- **Hosts should do MTU discovery**
- **Routers will not fragment: just send ICMP saying packet was too big**
- **Minimum MTU is 1280-bytes**
  - If some link layer has smaller MTU, must interpose fragmentation reassembly underneath

Current State

- **IPv6 Deployment has been slow**
- **Most end hosts have dual stacks today (Windows, Mac OSX, Linux, *BSD, Solaris)**
- **2008 Google study:**
  - Less than 1% of traffic globally
- **Requires all parties to work!**
  - Servers, Clients, DNS, ISPs, all routers
- **IPv4 and IPv6 will coexist for a long time**

Final Project

- Beef up Snowcast
- Tunnel TCP over DNS or ICMP
- Implement SSL-like protocol
- Error correcting transport
- Tor-like
- CDN – Akamai-like
- BitTorrent-like

Next time: Midterm

- **After that, transport layer and above!**
  - UDP, TCP, Congestion Control
  - Application protocols
  - …