Lecture #3: IPv6

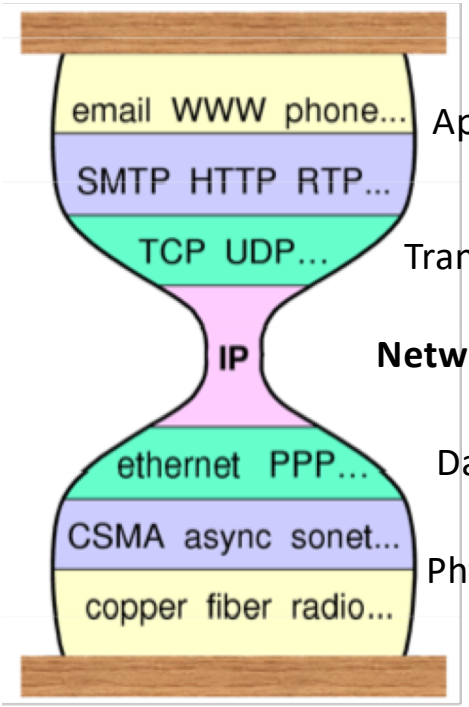
WPI CS4516 Spring 2019 D term

Instructor: Lorenzo De Carli (ldecarli@wpi.edu)

(slides include material from Christos Papadopoulos, CSU and Craig Shue, WPI)

IPv6

- **Network-level protocol** designed to be the successor of IPv4
- Responsibility: **datagram routing** across the Internet
- Implemented on **nodes** and **routers**



Background

- IPng recommended by IETF in July 1994
- Approved by Internet Engineering Steering Group in Nov. 1994
- Published as RFCs in December 1995.
- Formally assigned the name IPv6
- Article published June 1996

Motivation

- **Overcoming pressing IPv4 limitations** due to outdated design
- Introducing support for **new network-level primitives**
- What happened to **IPv5**?
  - Connection oriented experiment
  - Parts of it ended up in MPLS

A bit of history

IPv4

- Proposed in the early ‘80s
  - RFC 791, Sep 1981
- Deployed in 1983
  - Internet Flag Day, remember?
- **Still routes most internet traffic today**

It works… why change it?

- Multiple reasons
- The most pressing (and probably the most valid) one: **address exhaustion**
- IPv4 was designed in the early days of the Internet…
  - …therefore 2^{32} addresses seemed more than enough!
  - But already in the early ‘90s it became clear that this was not the case

IPv4 address space limitation

- In the initial IP design, **every valid address** (i.e., minus reserved ones) is expected to be **globally routable**
  - Every device on the network must have a public IP
- Initially, addresses were partitioned as 8-bit network + 24-bit host address
  - **Extremely wasteful!**
  - 16M host per network

IPv4 class system (1984, RFC 791)

- **Class A:** 8+24, MSB is 0 (128 networks)
- **Class B:** 16+16, MSBs are 10 (16384 networks)
- **Class C:** 24+8, MSBs are 110 (2M networks)
- **Class D:** multicast, MSBs are 1110
- **Class E:** reserved, MSBs are 1111
  - Unusable as some network equipment will refuse to route it
- Designed to **give some flexibility in address assignment** while **keeping HW simple**

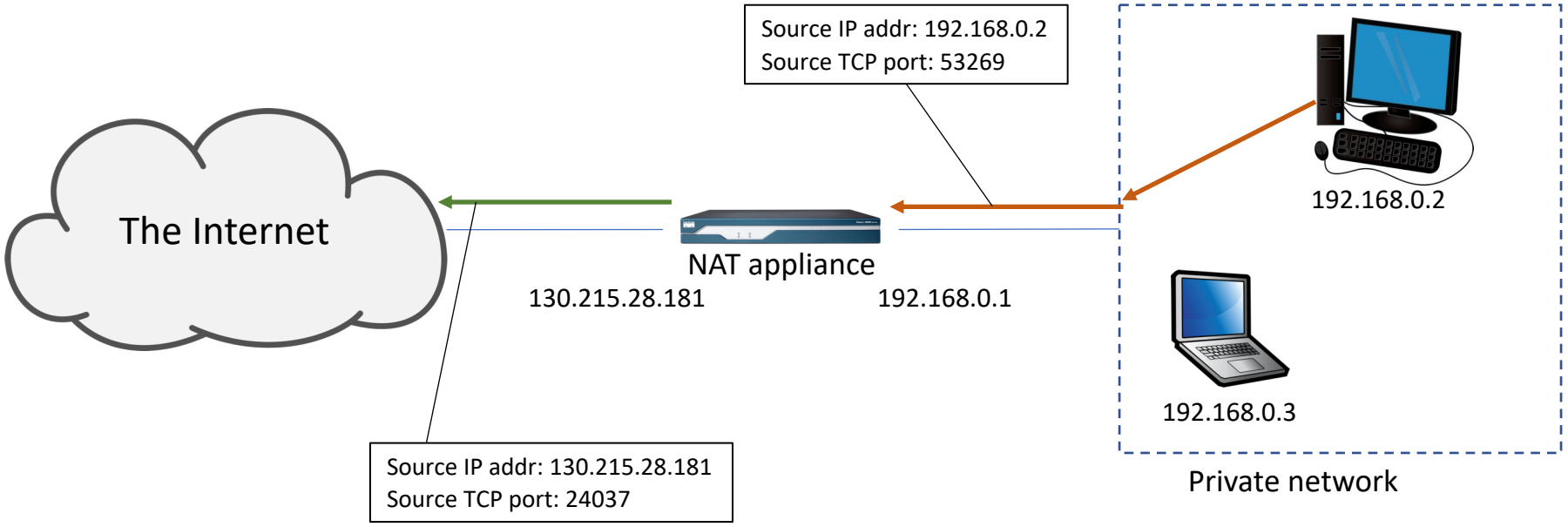
Classless IPv4

- By 1993, it was clear that **even the multi-class system was too wasteful**
  - The address space was used too sparsely and going towards exhaustion
- Solution: **classless IP (CIDR)**
  - Introduced by IETF in 1993
  - Does away with the notion of classes; **allows prefixes of any length to act as the subnet id**
  - Example: 130.215.28.0/24

NAT

- Classless IP works in tandem with the other “emergency measure” introduced to deal with address exhaustion: **NAT** (RFC 1631, 1993)
  - ”Network address translation”
  - In practice, **breaks the Internet into a public scope and many private scopes**
    - Nodes in the public Internet must have public IP addresses
    - All private networks reuse the same blocks of private addresses
    - A dedicated network appliance (NAT) translates private addresses to public addresses (and vice-versa) when nodes in a private network communicate with the public Internet

NAT - II



Note: other types of NAT are possible

Types of NAT (RFC 3489, 3/2003)

- **Full-cone NAT:** 1-to-1 mapping between *(address$_I$, port$_I$)* on internal node and *(address$_G$, port$_G$)* on NAT gateway
  - Any external host can send packets to *(address$_I$, port$_I$)* by addressing them to *(address$_G$, port$_G$)*
  - **Advantage:** hosts are reachable from the public Internet
  - **Disadvantage:** hosts are reachable from the public Internet

Types of NAT - II

- **Restricted-cone NAT:** 1-to-1 mapping between *(address$_I$, port$_I$)* on internal node and *(address$_G$, port$_G$)* on NAT gateway
  - An external host with IP *address$_E$* can only send packets to an internal host only if the latter has previously sent a packet to *address$_E$*
  - “Hides” internal hosts behind the NAT

Types of NAT - III

- **Port-restricted cone NAT:** 1-to-1 mapping between *(address$_I$, port$_I$)* on internal node and *(address$_G$, port$_G$)* on NAT gateway
  - An external host with IP, port *(address$_E$, port$_E$)* can only send packets to an internal host only if the latter has previously sent a packet to *(address$_{E,}$ port$_E$)*

Types of NAT - IV

- **Symmetric NAT:** NAT gateway maps each flow *(address$_I$, port$_I$, address$_E$, port$_E$)* to a specific *(address$_G$, port$_G$)* on the gateway
- Note the difference between *cone NAT and symmetric NAT:
  - The former assigns **gateway address/port based on address source/port**
  - The latter assigns **gateway address/port by flow**
  - **Symmetric NAT** is the trickiest to bypass because it makes mapping unpredictable

NAT pros & cons

- When proposed, **NAT was controversial**
  - Breaks the principle “one node, one (or more) public IP”
- But it has advantages
  - **Decouples** the number of Internet-connected nodes in an organization from the size of its Internet presence
    - An organization (e.g., university) may want all computing nodes to have connectivity, but may only need a few IPs for public Internet presence, email etc.
  - NAT **can hide private network topology** and **make private node unreachable by default**

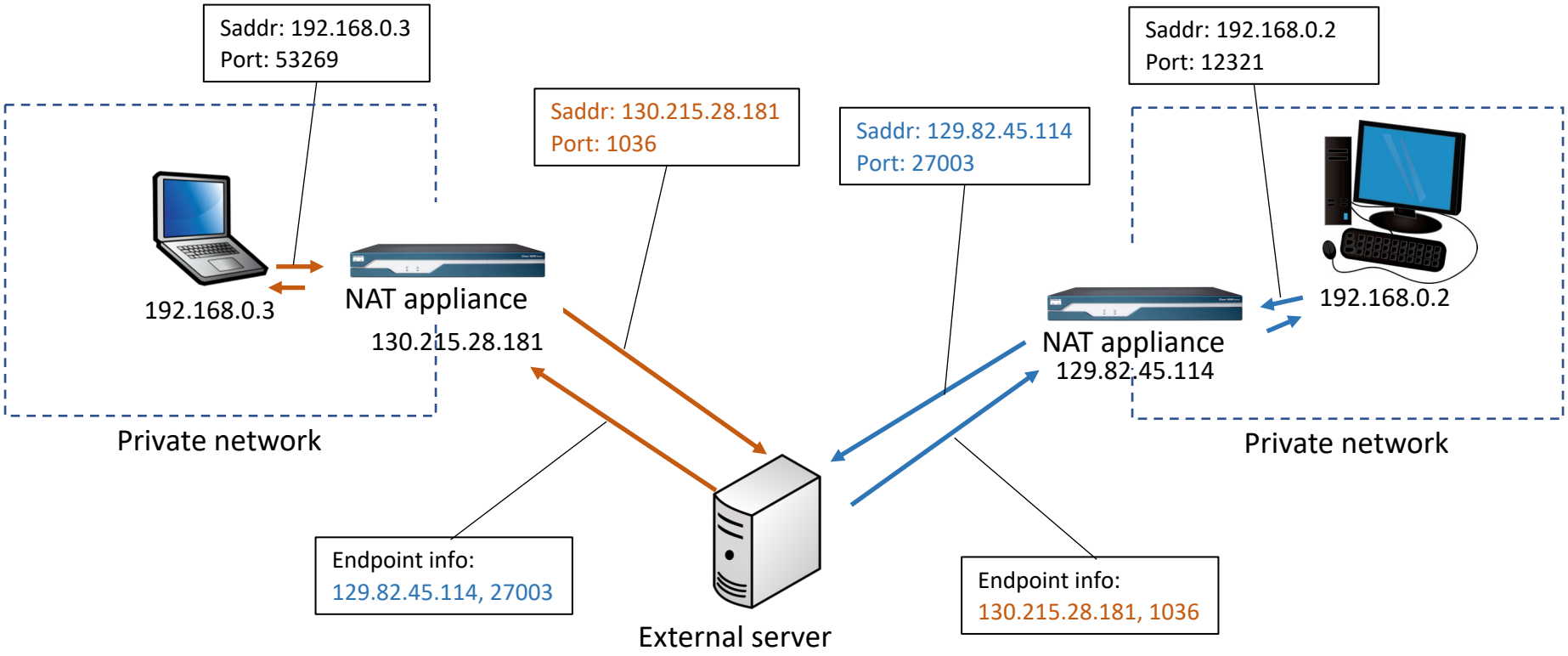
NAT issues

- Oftentimes, it is useful to “hide” a network behind a NAT but it is desirable to **allow some internal host addresses/ports to be reachable from the outside**
- **Various techniques:** hole-punching, port forwarding

NAT hole-punching

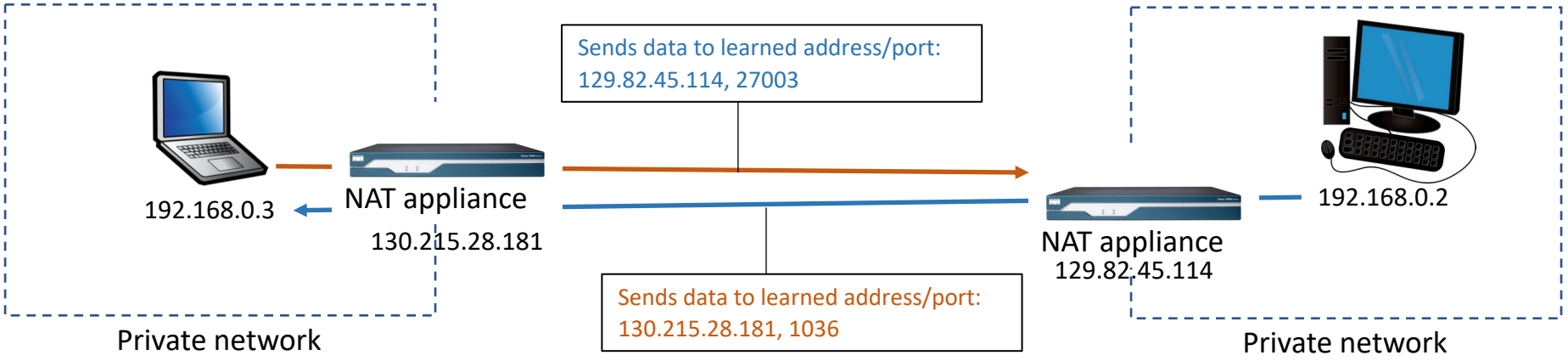
- Works for *cone NAT, where the association between internal and external addresses/ports is **persistent**
- Designed for situations where **two hosts behind different NATs want to communicate**
- Hosts use a third-party - a server w/ IP address – to determine the **external addresses and port to which are mapped**
- A form of this is defined in RFCs 3489/5389

Hole punching example



What's missing?

Hole punching example - II



If restricted NAT, will need to make multiple attempts (why?)

What if it does not work?

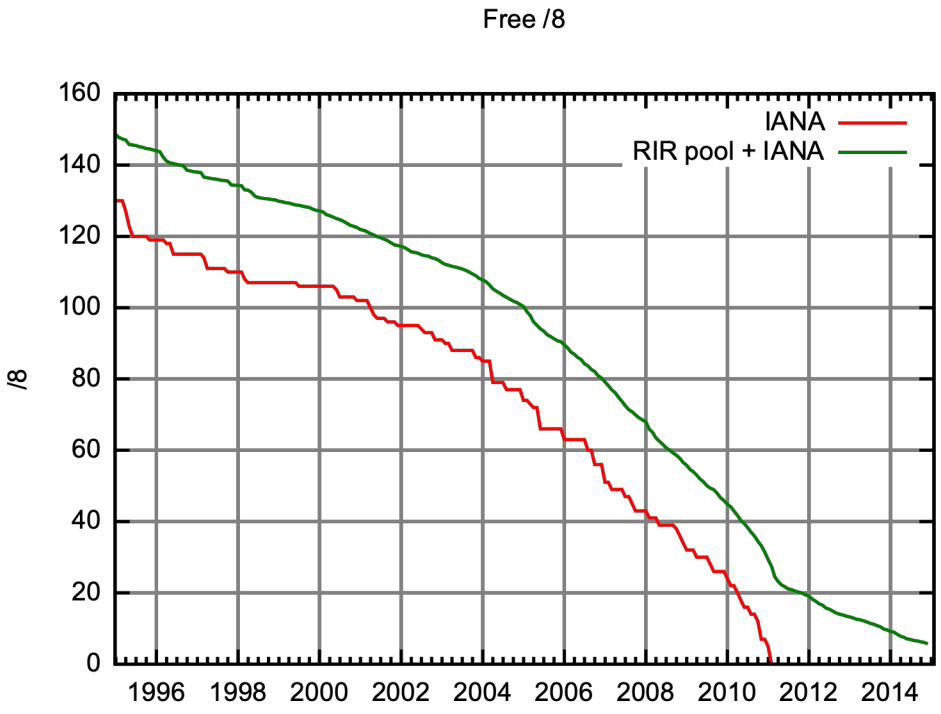
- (E.g.: symmetric NAT, nonstandard configurations, etc.)
- **Port forwarding:** configure an external port on the NAT gateway to **always forward traffic** to an internal host/port
- Those of you who are running their own home server probably do this ☺
- **uPNP** (supported by pretty much all home routers) automates port forwarding

CIDR + NAT

- Summarizing:
  - NAT **lowers the number of public IPs** that an organization needs to have Internet connectivity
  - CIDR **allows assignments of very small blocks of public IPs** to an organization
  - By allowing many nodes to use private addresses, **the pressure on the IP address pool was drastically decreased!**

Yes, but did it work?

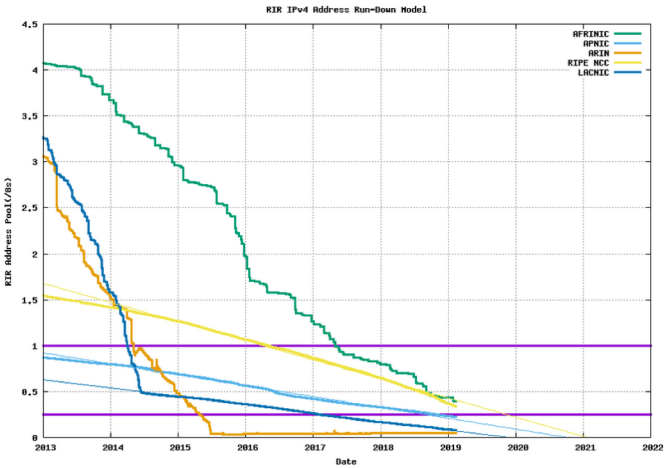
- Quite well, in fact – but we are at the end



By Mro - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=10593349

Projected RIR Address Pool Exhaustion Dates:

RIR	Projected Exhaustion Date	Remaining Addresses in RIR Pool (/8s)
APNIC:	**19-Apr-2011** (actual)	0.2252
RIPE NCC:	**14-Sep-2012** (actual)	0.3364
LACNIC:	**10-Jun-2014** (actual)	0.0806
ARIN:	**24 Sep-2015** (actual)	
AFRINIC:	**18-Sep-2019**	0.3978



Projection of consumption of Remaining RIR Address Pools

Geoff Huston, ipv4.potaroo.net

Now, let’s move on to IPv6

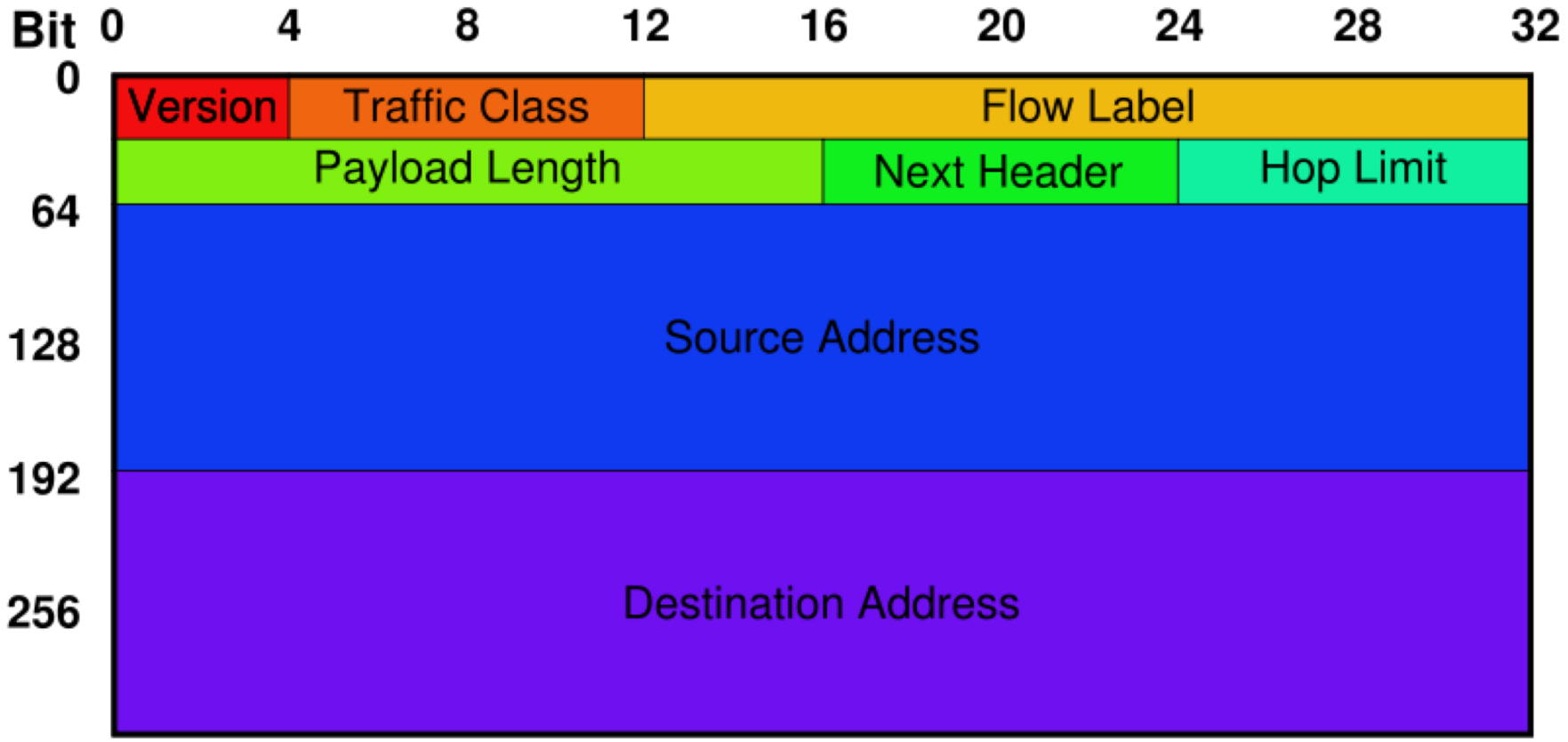
IPv6 design principles

- **More addresses!**
  - Motivation is less obvious than it may seem – we'll go over it
  - **Also, more expressive** addressing scheme
    - IPv6 implements several primitives that in IPv4 require external protocols
- **Ease of transition** from IPv4
- **Better security**
- Better QoS, support for host mobility, etc.

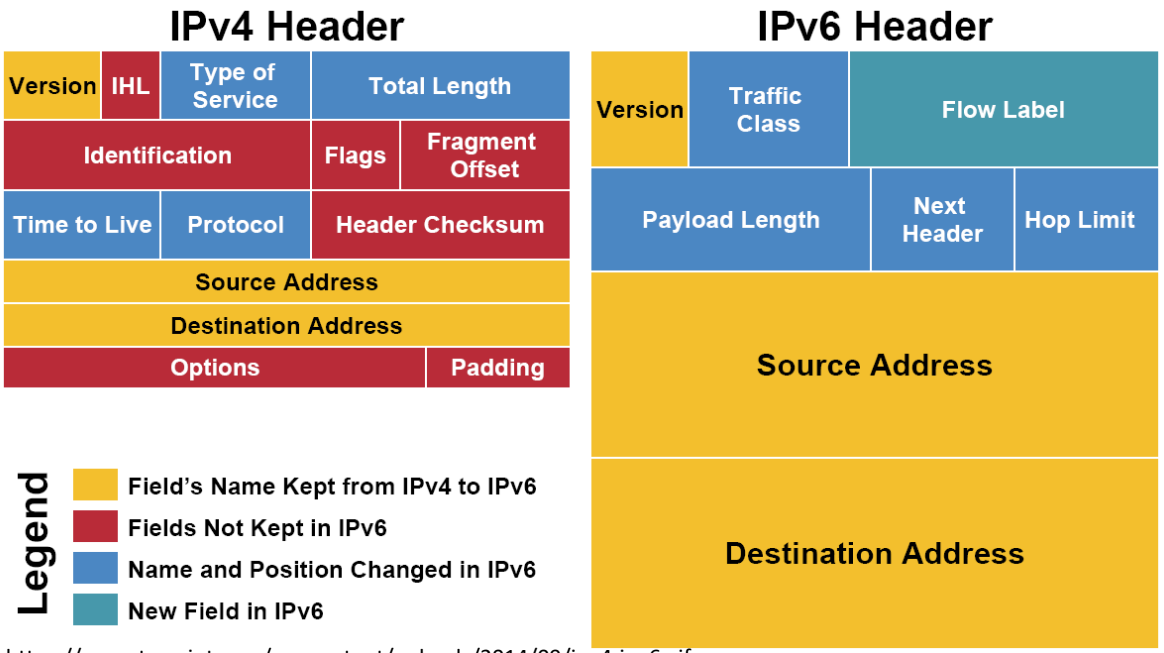
Why more addresses?

- One reason is obvious: **IPv4 address space exhaustion**
- But there is also a vision on the **future of computing** (1996)
  - General-purpose computer market will keep expanding
  - “Nomadic personal computing devices”
  - ”Networked entertainment”
  - “Device control”

IPv6 Header


Bit
0
4
8
12
16
20
24
28
32
0
64
128
192
256
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address

Compare to IPv4 header



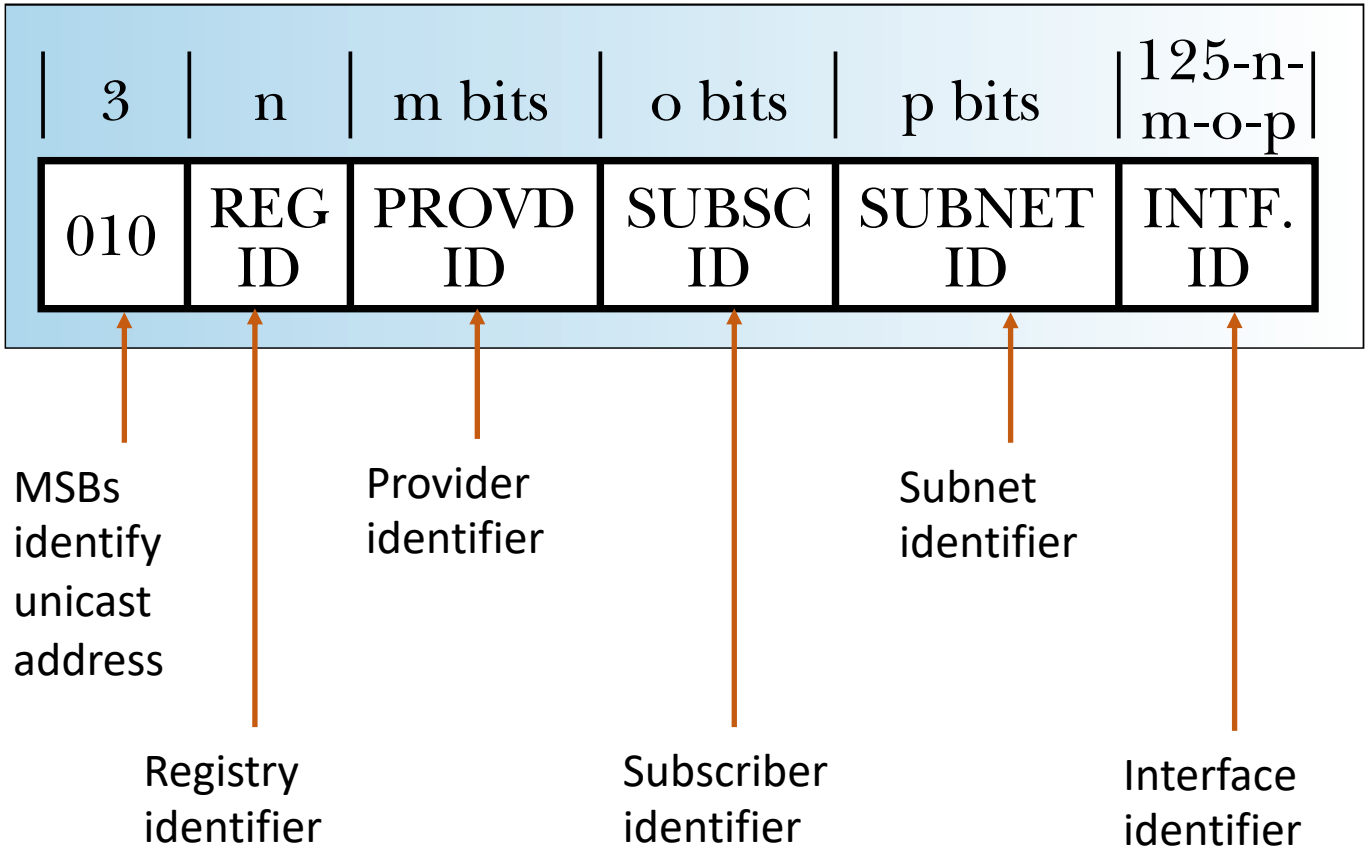
https://www.tecmint.com/wp-content/uploads/2014/09/ipv4-ipv6.gif

- Addresses are 4 times longer (3.4E38 possible values of an address field)
- However, header is only 2 times longer (attempt to keep overhead acceptable)
- (Still, it made the life of router designers quite difficult)

IPv6 Address Space Allocation

- Vast majority is **reserved** for future use
- **Aggregatable Global Unicast Addresses** make up **1/8** of all addresses
- "Link local use" addresses **are usable on local network**, but **not globally unique**
  - **Autoconfiguration**
- "**Site local use**" is used by **private networks**

Global unicast addresses



Local-use addresses

- Designed to be in large part **autogenerated** without the need for external information
  - Enable a node to get an address guaranteed to be (at least locally) unique without relying on other nodes
  - Useful for node **autoconfiguration** when first connected to the network
- How does it work?

Local-use addresses - II

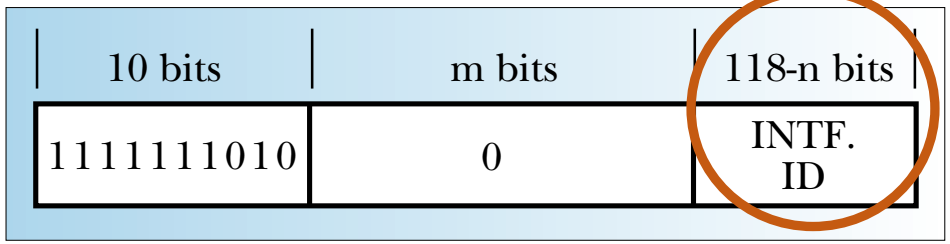


Figure 4. Link-Local address format

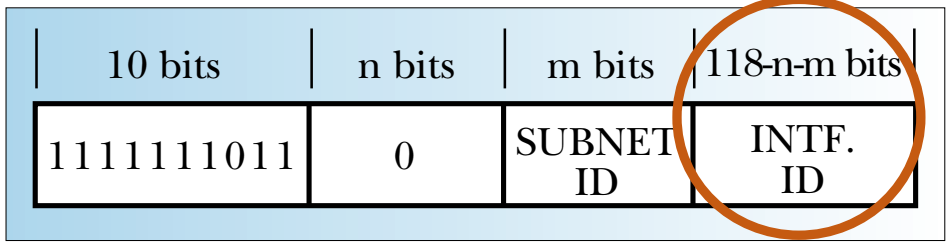


Figure 5. Site-Local address format

Typically constructed from the **layer-II address** (e.g., MAC address which is "guaranteed" to be unique)

Goal: ease of transition

- It was obvious to IPv6 designers that the Internet could not be coerced to switch from IPv4 to IPv6
- In the early 90's, the network was already large and complex enough that **a flag-day style transition would have been untenable**
  - Who remembers the other flag day?
- **Significant capital investment** in IPv4 hardware, software stack, etc.
- IPv6 is designed from the ground up to **coexist** with IPv4 and **make the transition as painless as possible**

IPv4-compatible addressing scheme

1: IPv4-**mapped** IPv6 addresses

- First, IPv6 address can **embed** IPv4 addresses

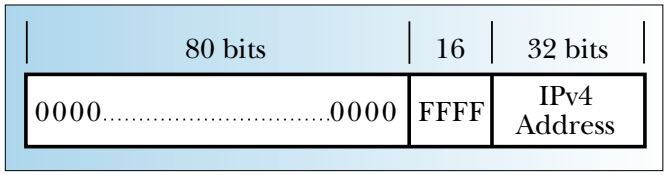
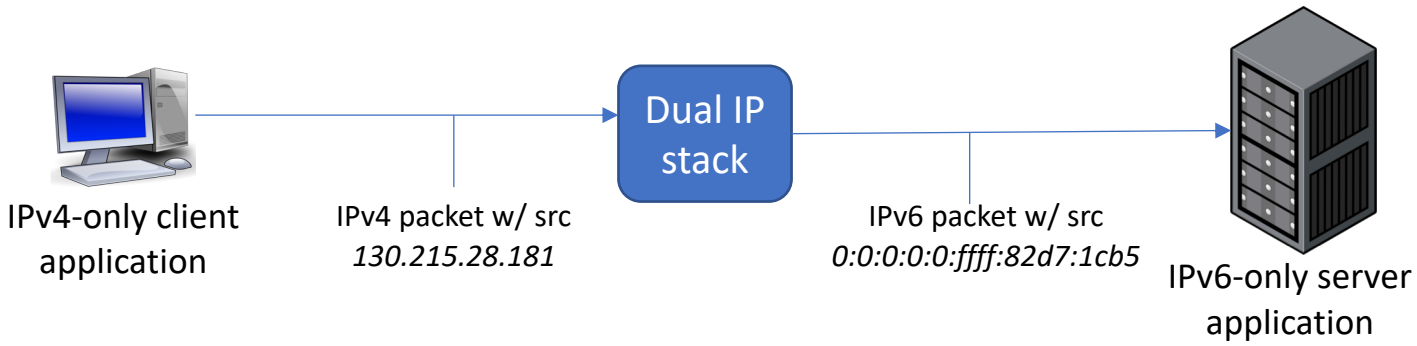


Figure 7. IPv4-mapped IPv6 address

- What is this for? (RFC 4038, "Application Aspects of IPv6 Transition")



2. IPv4-**compatible** IPv6 addresses

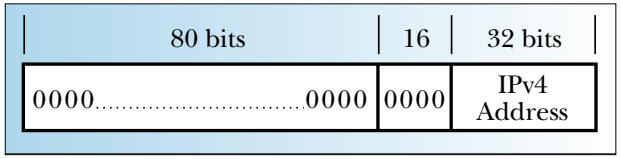
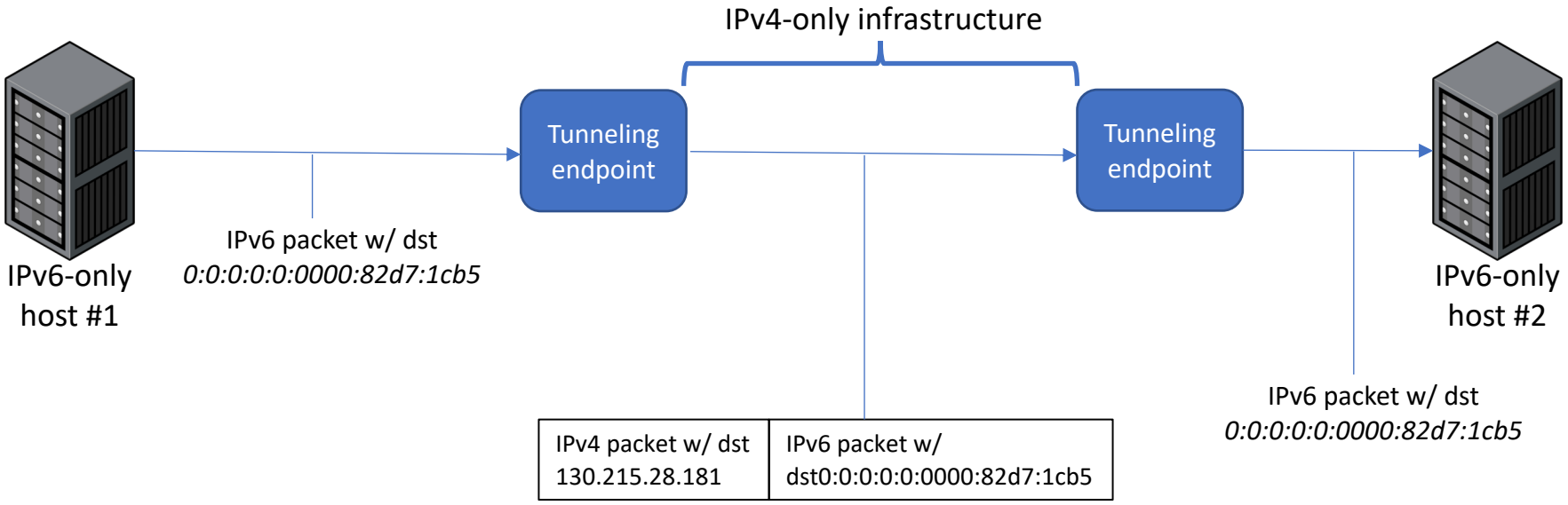


Figure 6. IPv4-compatible IPv6 address

- These ones solves an **infrastructure issue**
  - How to route packets between two IPv6 hosts **through IPv4 infrastructure**
  - Idea:
    - **Tunnel** IPv6 in IPv4
    - Derive destination IPv4 address from destination IPv6 address

IPv6 over IPv4 tunneling

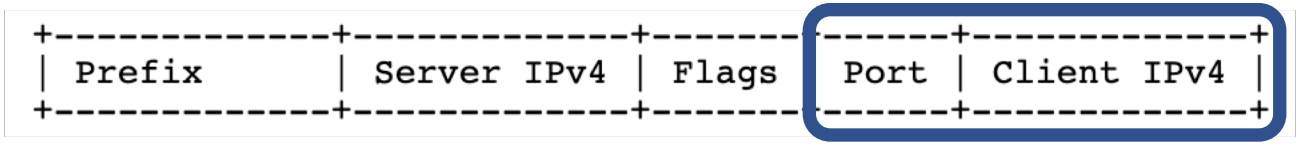


Teredo tunneling

- Another flavor of IPv6-over-IPv4 tunneling
- IPv6 tunneled in UDP over IPv4
- Why? Regular tunneling hides layer 4 and therefore breaks NAT hole punching & related techniques
- Developed at Microsoft, standardized in RFC 4380

How does Teredo work?

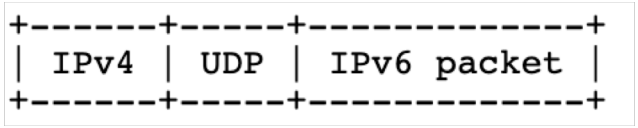
- Client interacts with a **Teredo server** to find out **mapping** between local and NAT address/port (a form of hole punching)
- Server generates **an IPv6 address for the client**. The address encodes the public address/port used on the NAT gateway:



Public IPv4 address/port for the IPv6 client behind NAT

How does Teredo work? - II

- After client is assigned an IPv6 address, it uses a **Teredo relay** to communicate with other IPv6 peers
- The client **encapsulate**s IPv6 packets into UDP over IPv4:

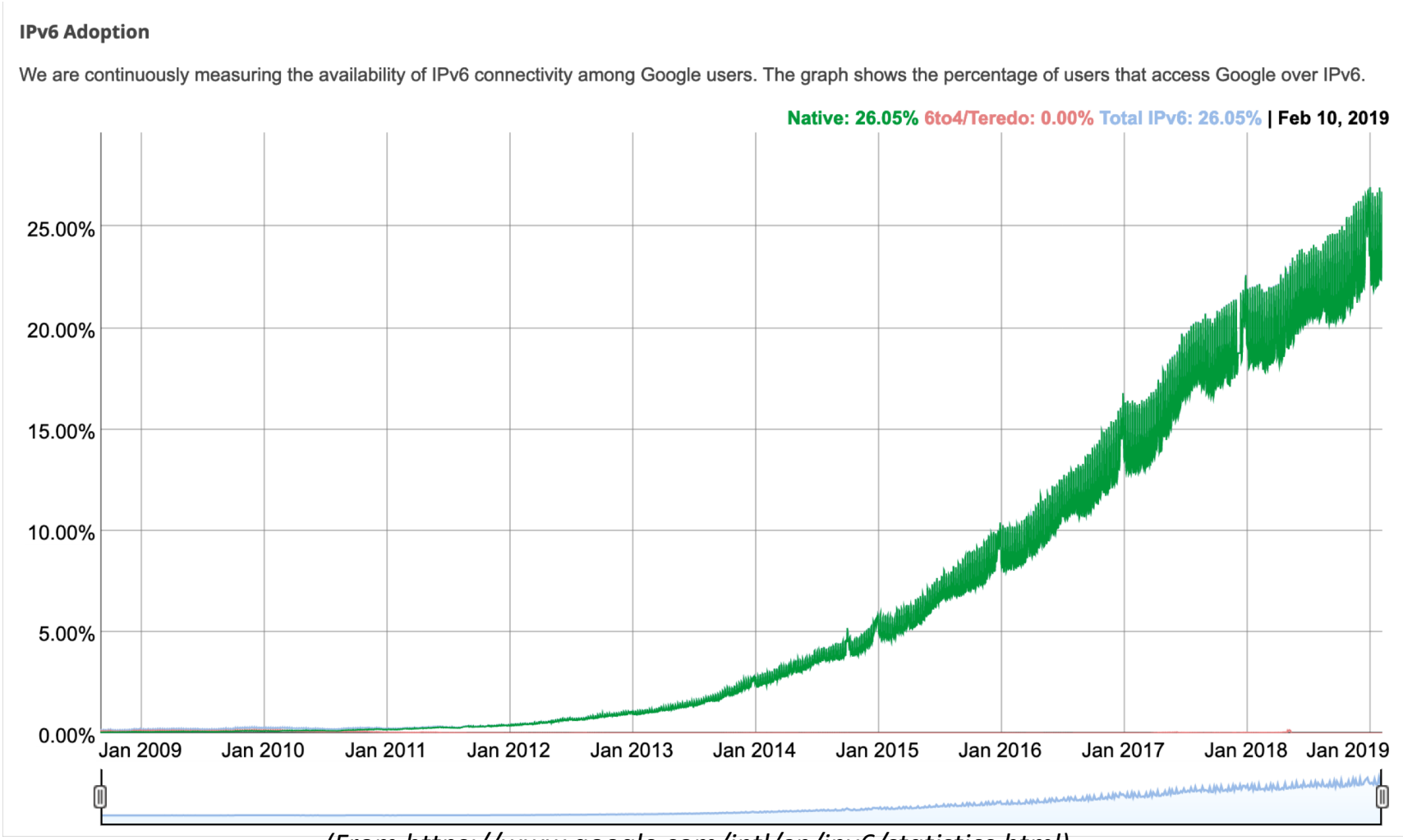


- The relay receives these packets, **decapsulates them**, and sends them to the IPv6 destination
- When receiving packets for the host, the relay performs **encapsulation**

Other tunneling type

- Over time, various **other tunneling techniques** have been defined/used
  - 6rd
  - ISATAP
  - …
- All these technologies are variations of the tunneling principle with modifications to solve specific problems
- Bottom line: **the IPv6 community went to great lengths to ensure smooth transition**

Did they succeed?



(From https://www.google.com/intl/en/ipv6/statistics.html)

Encouraging IPv6 adoption

- World IPv6 day: 6/8/2011
  - Several large companies (Facebook, Google, Akamai, …) enabled IPv6 on their public-facing network for 24 hours
- World IPv6 launch: 6/6/2012
  - Make the activation permanent

Adoption problems

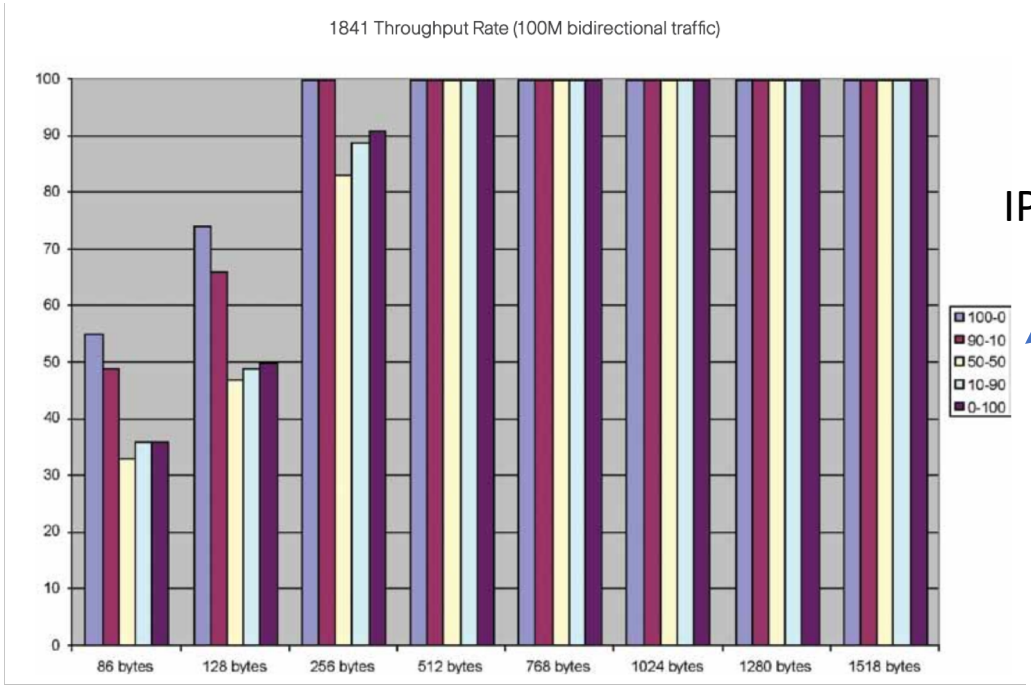
- **Malfunctions**
  - A typical OS networking stack is designed to prefer IPv6 connectivity when available
  - Especially in the early day of IPv6 adoption, a network may advertise IPv6 connectivity but **have it configured incorrectly**
  - **Consequence:** a node may attempt to sends packets via a **broken IPv6 configuration**, rather than a **working IPv4 one**
  - Rare today due to **increased operational experience**/familiarity of operators with IPv6

Adoption problems - II

- Any respectable router implements IP forwarding using **custom hardware logic**
- IPv4 addresses are **substantially different** (shorter, w/ different internal structure) than IPv6 ones
- Implementing IPv6 forwarding requires **redesigned hardware**
  - Early dual IPv4/IPv6 routers implemented IPv4 forwarding in hardware, IPv6 forwarding in software, with **substantial performance drop**

Adoption problems - III

- Even when a router is designed from the ground up to support IPv6, it may be difficult to match performance of IPv4 forwarding
  - IPv6 address lookup requires larger buffers, more operations
  - E.g.: 2007 comparison for CISCO 1841 router:



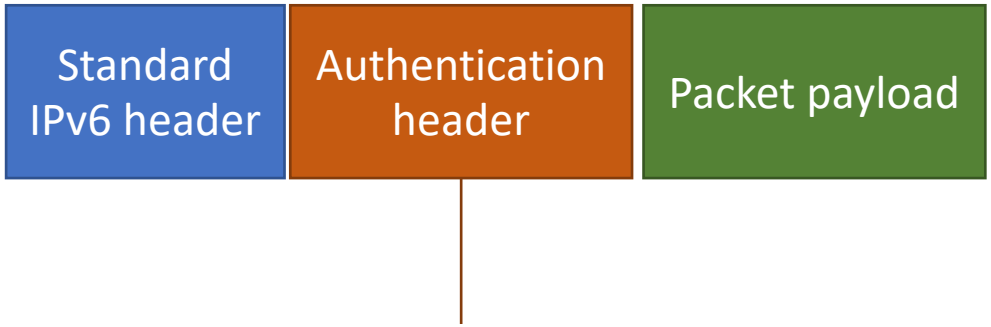
IPv4/IPv6 split

Progress in router design made this a manageable problem

Goal: better security

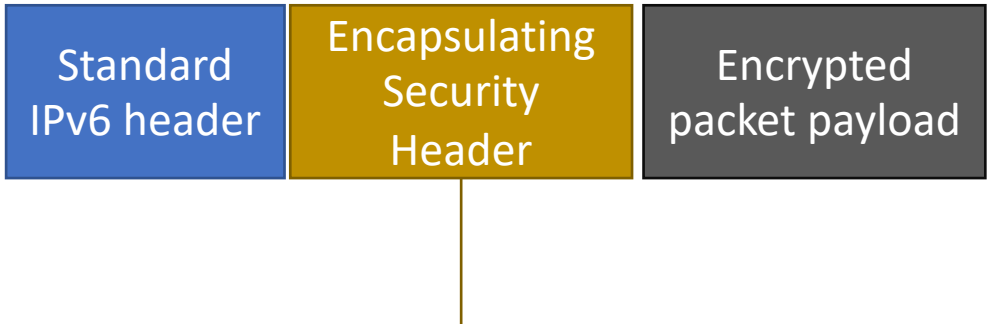
- IPv4 does not have any **security mechanism** built-in
  - Security properties built on top if it
- IPv6 has **IPSec-style security** built in
- What is IPSec?
  - Protocol providing:
    - **Authentication** (i.e. proof of origin) and **integrity** using the IP Authentication Header (AH)
    - **Confidentiality** (encryption) using Encapsulating Security Payload (ESP)

Authentication/Integrity in IPv6



- Covers all immutable fields of IPv6 header (e.g., exclude flow label)
- Includes:
  - Various metadata
  - A sequence number to prevent replay attacks
  - An integrity check vector (hash, keyed hash etc.)

Confidentiality in IPv6



- Provides metadata describing specifics of encryption used on the payload
- Includes:
  - A session identifier (security association identifier)
  - Sequence number

But wait… there is more

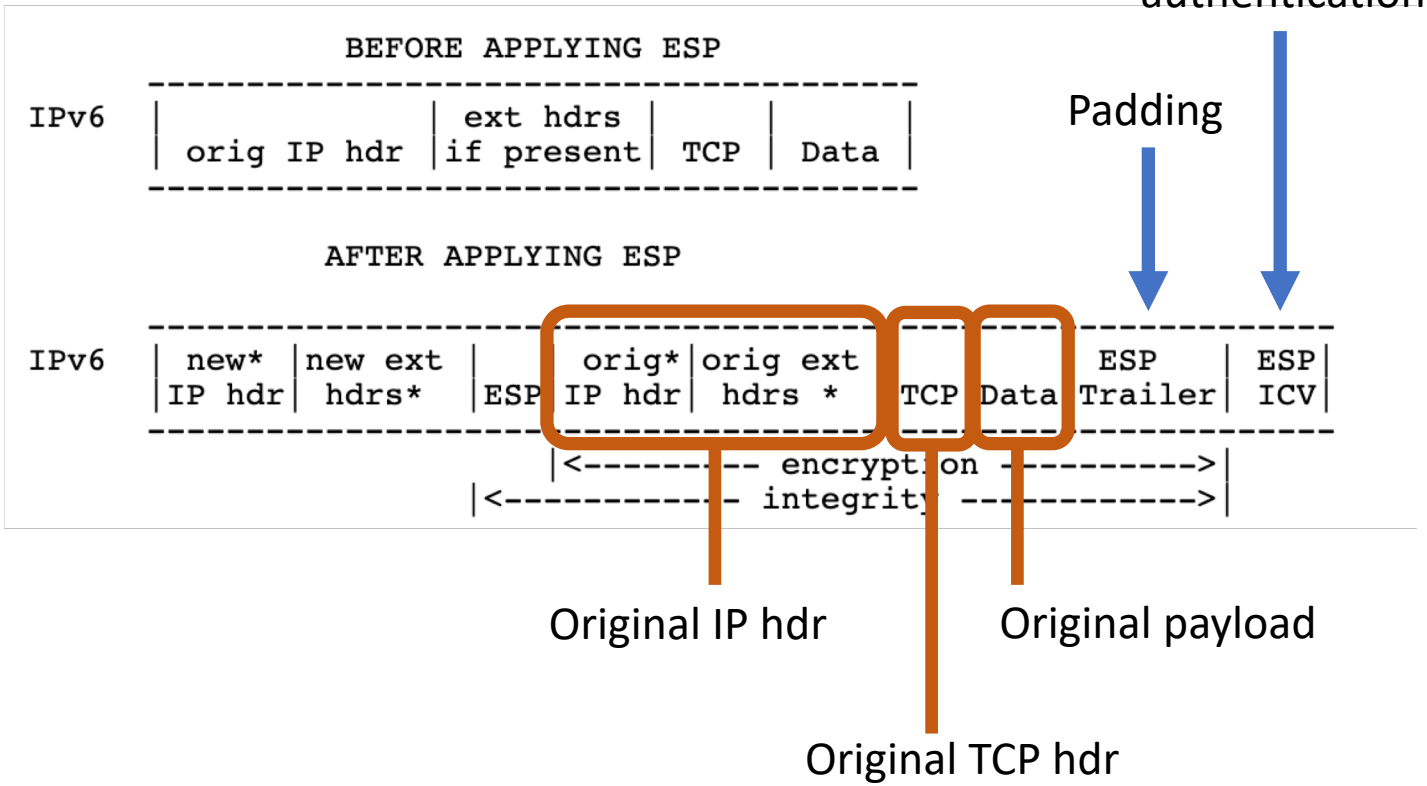
- What we have seen refers to authentication/confidentiality in **transport mode**
- IPv6 also support secure communication in **tunnel mode** (AKA VPN)

IPv6 tunnel mode

- **Tunnel mode** allows to tunnel IPv6 packets within IPv6 packets. Security can be applied so that the ESP header covers the entirety of the tunneled packets (including addresses and other header fields)
  - **Advantage:** source and destination (not jus content) remain secret!

IPv6 tunnel mode - II

(Source: RFC 4303, 12/2005)



Other IPv6 goals

- Fix a bunch of IPv4 shortcomings
  - Better QoS
  - Better multicast/anycast
  - Better option handling

IPv6 QoS/flow labeling

- Providing quality of service at network level it is always been a challenging task
  - **Integrated services** (intserv): per-flow QoS
  - **Differentiated services** (diffserv): per-class QoS
- IPv6 provides **two mechanisms**:
  - **Flow labels**, allowing to mark and identify set of flows that should receive a certain treatment
    - Should be randomly assigned, so that they can act as indices in hash tables
    - Do not specify *how* a flow should be treated – that must be determined using a different protocol
  - **Priorities among traffic classes**

IPv6 multicast

- Propagating packets to **multiple destinations** has obvious benefits for **one-to-many communication**
- IPv4 includes:
  - **Broadcast**: transmit to all hosts on a subnet
  - **Multicast**: transmit to specific sets of hosts
  - Both use sets of reserved addresses
- IPv6 works in the same way but **drops broadcast addresses**
  - Why?
  - Limited usefulness, redundant with layer II, obvious security issues

IPv6 anycast

- Send packets **to one among multiple possible destinations**
- Use cases?
  - **Mobility**
  - **Load-balancing**
- No reserved addresses – any address from the unicast address space can be used for anycast

Better option handling

- **Options** are used in IPv4 and IPv6 to **provide functionality beyond basic routing**
- In IPv4
  - **Limits** on individual and total **option length**
  - Option support required
- In IPv6:
  - ”Next header field” points either to layer-IV packet or option, specifying its type
  - **No limit on number of options**, higher limits on length
  - Routers can skip options they don’t recognize/care about