W9
(2a)

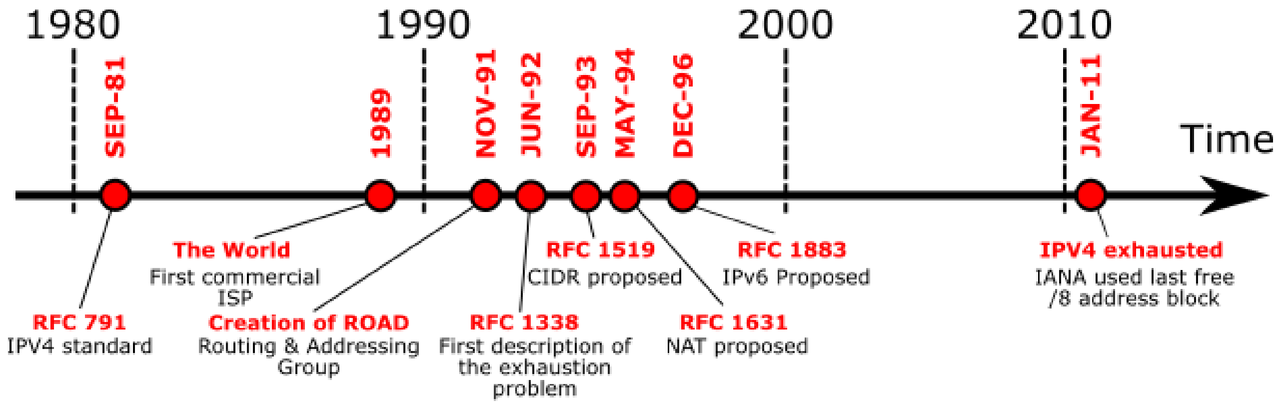
Computer Networks

Network Address Translation and IPv6

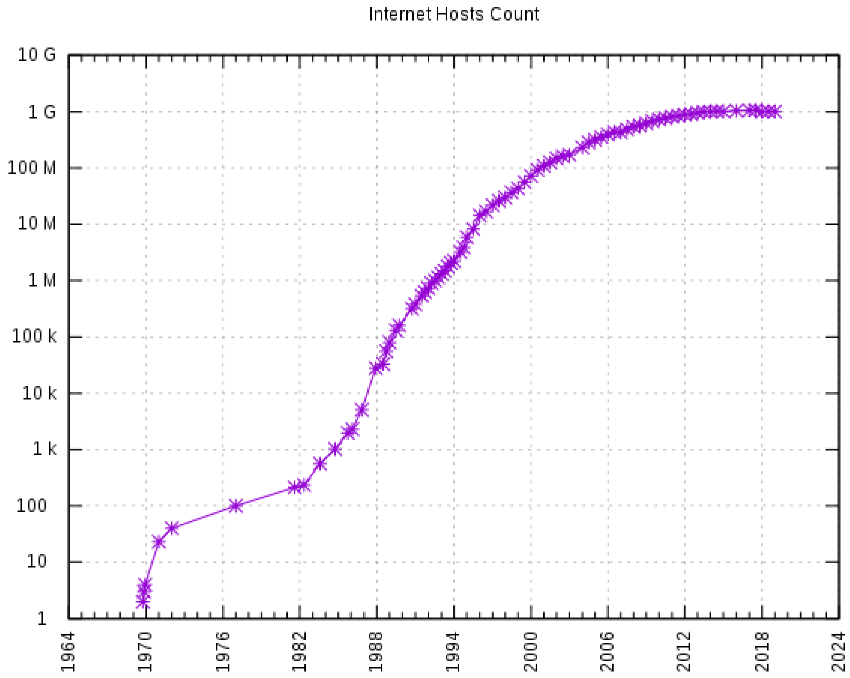
Amitangshu Pal
Computer Science and Engineering
IIT Kanpur

IPv4 Address Space Exhaustion

- IPv4 address space is limited
  - Number of Internet users increased exponentially
  - NAT: Network Address Translation
  - IPv6 addressing



Src: https://commons.wikimedia.org/wiki/File:IPv4_exhaustion_time_line-en.svg



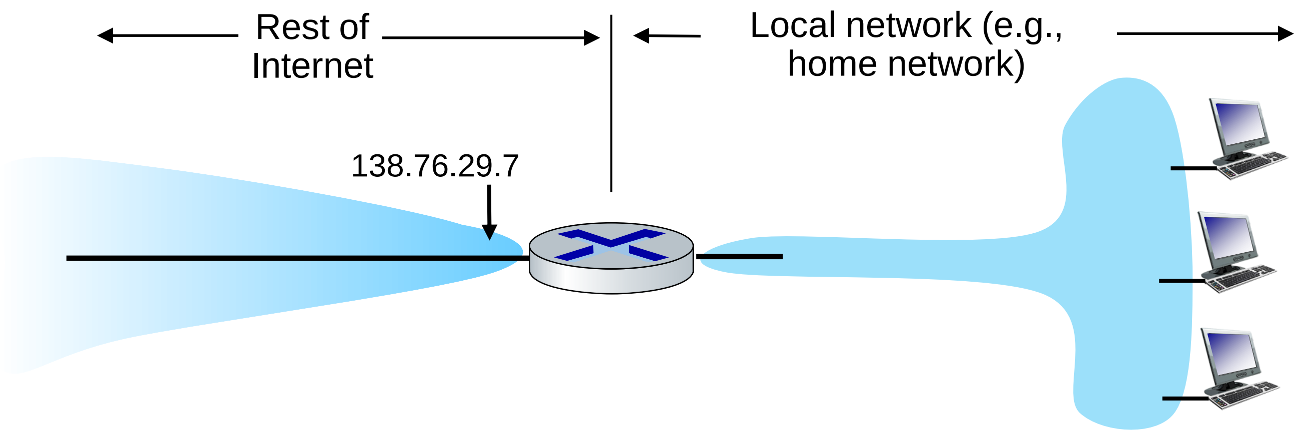
Src: https://commons.wikimedia.org/wiki/File:Internet_Hosts_Count_log.svg

Network Address Translation

NAT: Network Address Translation

Home network:

- One access point has one IPv4 address
- Suppose 10 hosts are connected to the access point



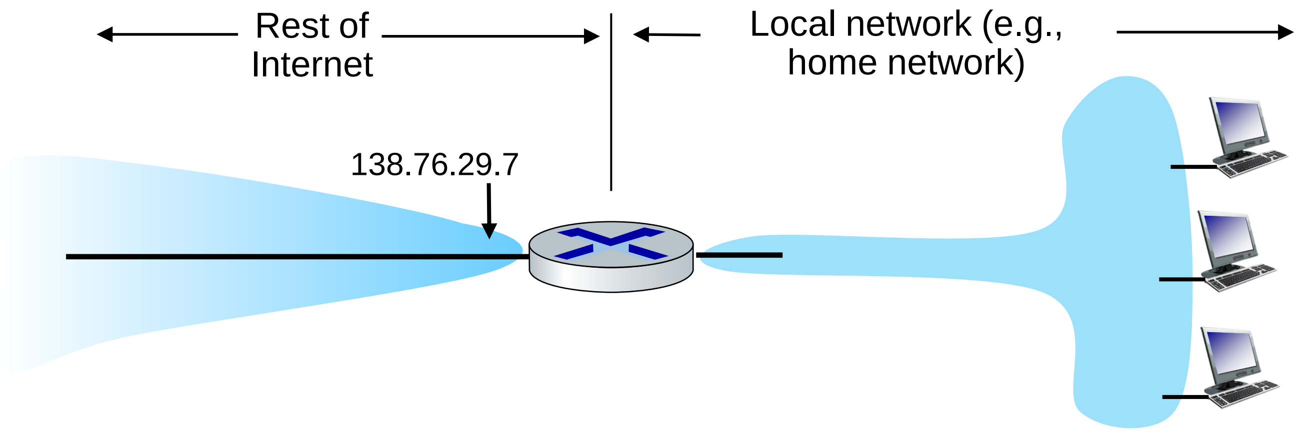
Public and Private IP Address

Private IP address:

- Can be reused
- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 – 192.168.255.255

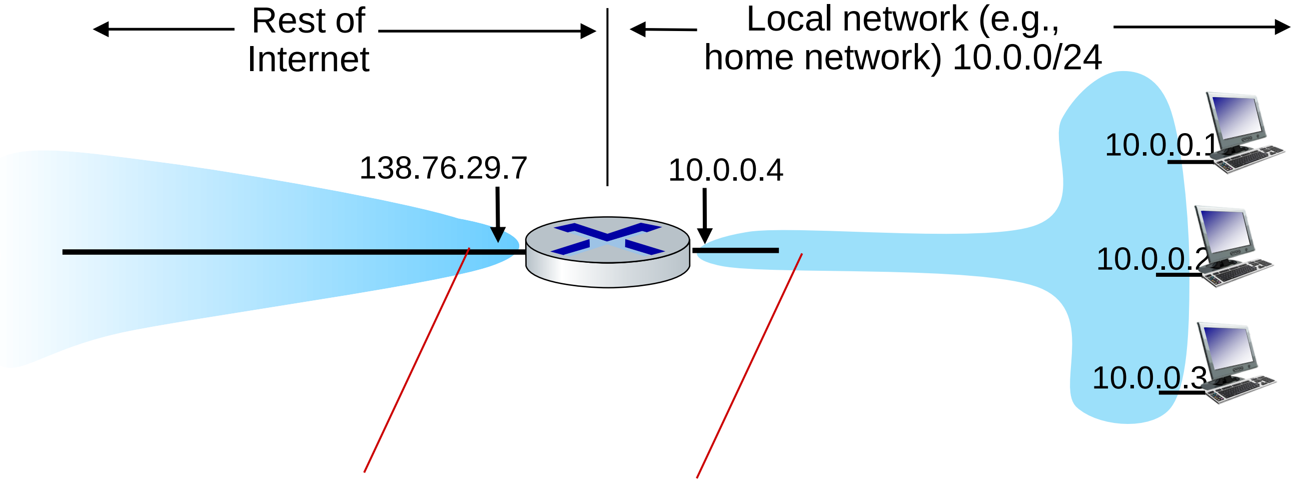
Public IP address:

- Cannot be reused
- Globally unique



NAT: Network Address Translation

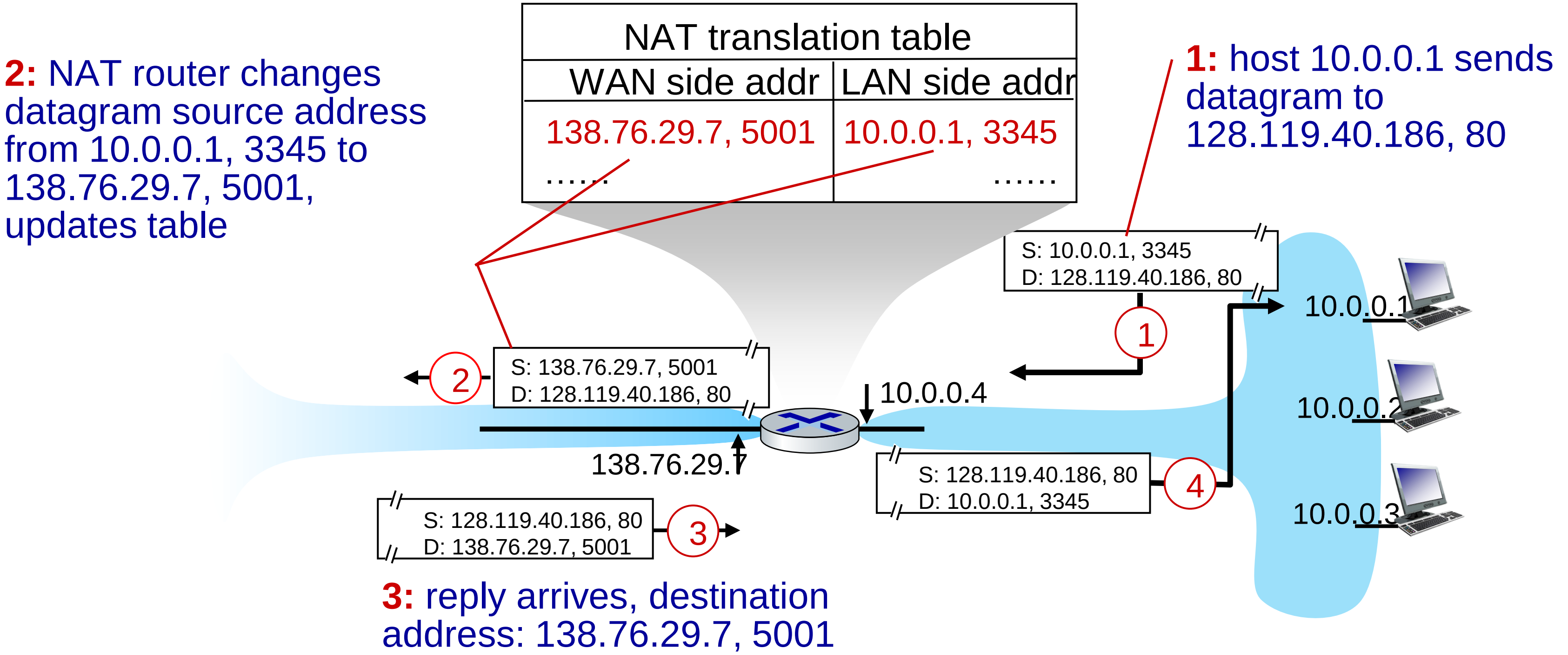
NAT: All devices in local network share just one IPv4 address as far as outside world is concerned



All datagrams leaving local network have same source NAT IP address: 138.76.29.7, but different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation



More info: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html

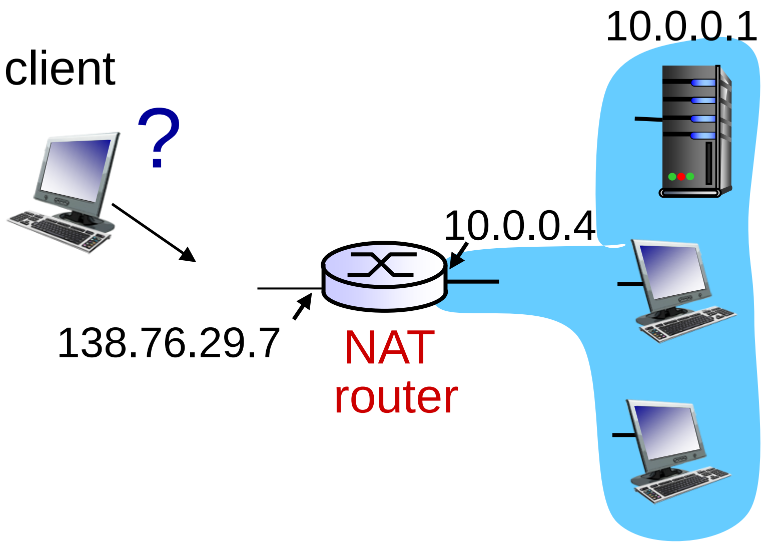
NAT: Network Address Translation

Implementation: NAT router must (transparently):

- Outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination address
- Remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
- Incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT Traversal Problem

- Client wants to connect to server with address 10.0.0.1
  - Server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - Only one externally visible NATed address: 138.76.29.7

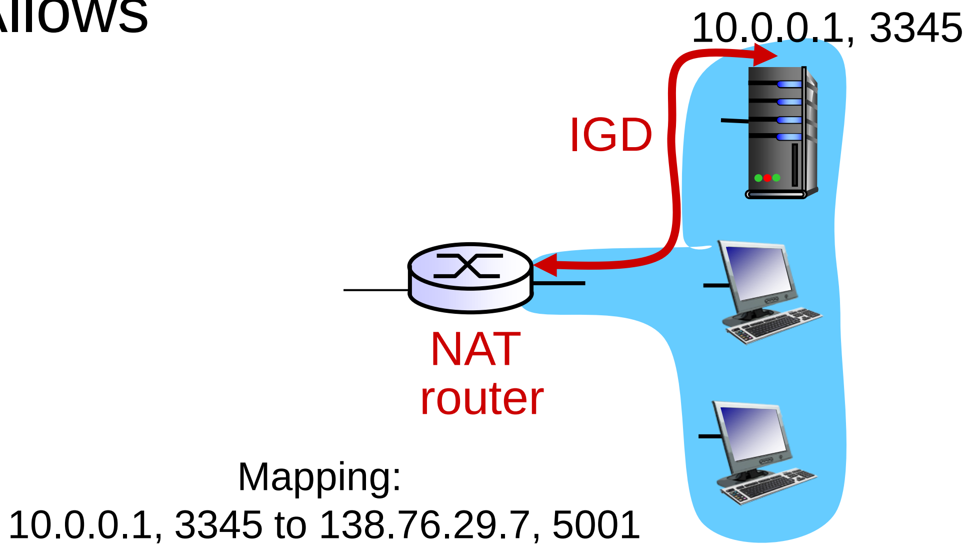


NAT Traversal Problem:

Imagine you have a client (let's call it Client A) on a local network trying to connect to a server (let's call it Server B) with the address 10.0.0.1. However, Server B's address is local to the LAN (Local Area Network), which means Client A can't directly use it as the destination address for the connection. Additionally, there's only one externally visible NATed address (let's say 138.76.29.7) available for the whole network.

NAT Traversal Problem

- Solution 1: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - Learn public IP address (138.76.29.7)
  - Add/remove port mappings (with lease times)



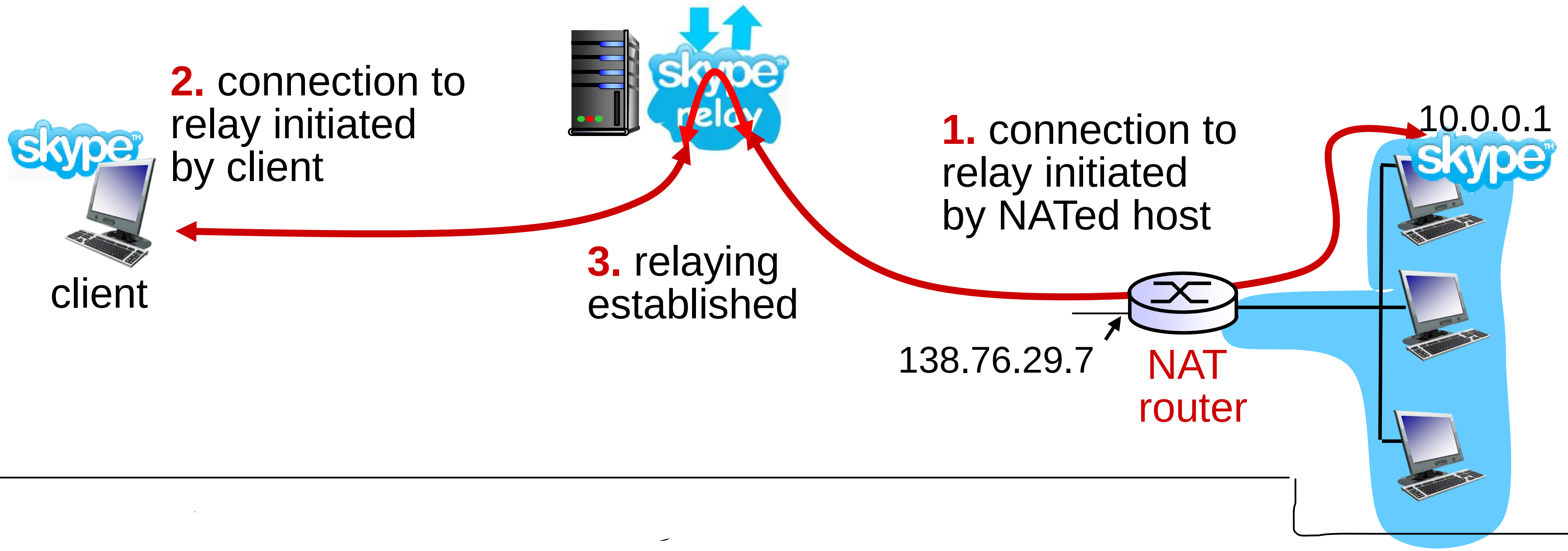
Solution 1: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol:

This solution involves using a protocol called UPnP, which allows devices within the local network to communicate with the NAT router and perform certain actions like learning the public IP address (138.76.29.7) and adding or removing port mappings. These port mappings are important because they tell the NAT router how to correctly route incoming and outgoing traffic for specific applications or services running on devices within the network.

NAT Traversal Problem

- Solution 2: Relaying (used in Skype)

- NATed client establishes connection to relay
- External client connects to relay
- Relay bridges packets between to connections



Solution 2: Relaying:

In this solution, when Client A wants to connect to Server B but can't do so directly due to NAT restrictions, it establishes a connection to a relay server instead. This relay server acts as a middleman. At the same time, an external client (let's call it Client C) that wants to communicate with Client A also connects to the same relay server. The relay server then acts as a bridge, forwarding packets between Client A and Client C.

To put it simply, imagine Client A is sending messages to Server B through a relay runner (the relay server). The relay runner then passes these messages to Server B. Similarly, messages from Server B are relayed back to Client A through the same relay runner.

NAT: Network Address Translation

- Advantages:
  - Just one IP address needed from provider ISP for all devices
  - Can change addresses of host in local network without notifying outside world
  - Can change ISP without changing addresses of devices in local network
  - Security: devices inside local net not directly addressable, visible by outside world

NAT: Network Address Translation

- NAT has been controversial:
  - Routers "should" only process up to layer 3
  - Violates end-to-end argument (port # manipulation by network-layer device)

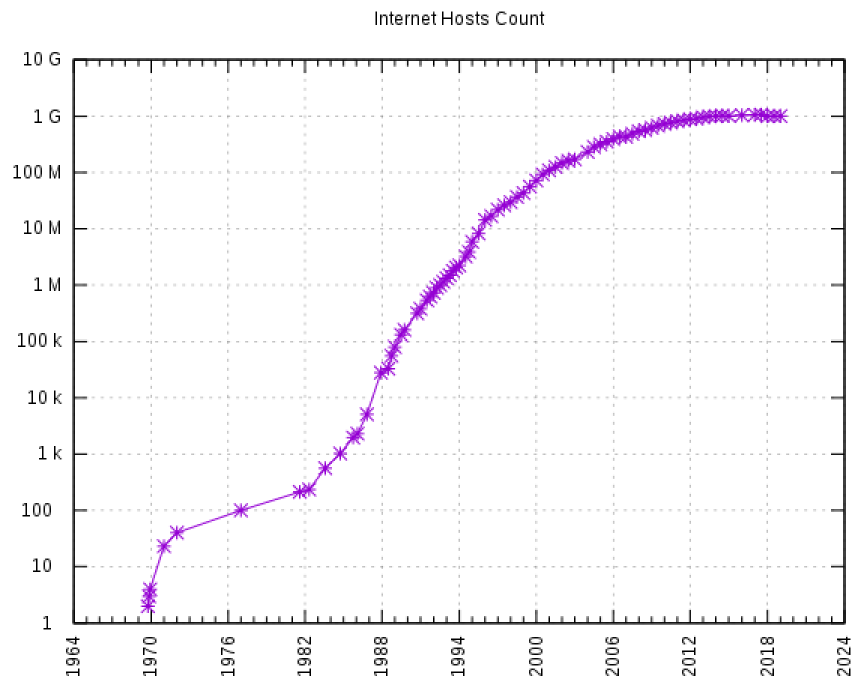
- But NAT is here to stay:
  - Extensively used in home and institutional nets, 4G/5G cellular nets

IPv6

IPv6 Motivation

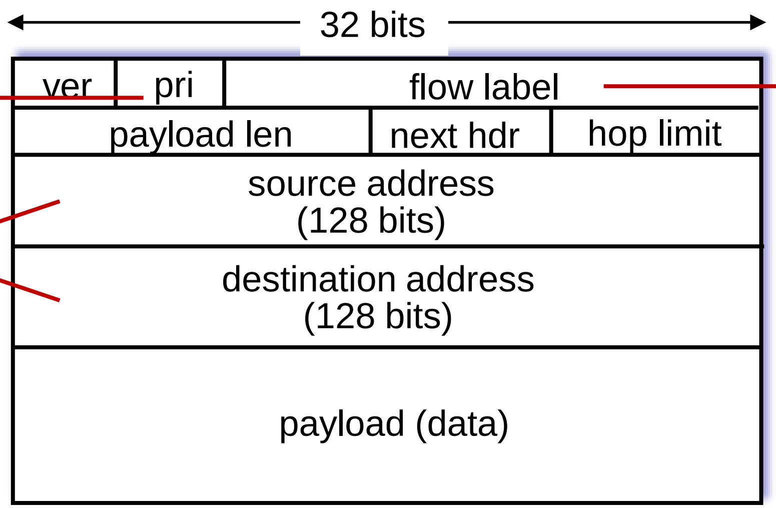
- Initial motivation: 32-bit IPv4 address space would be completely allocated
- Additional motivation:
  - Speed processing/forwarding: 40-byte fixed length header
  - Enable different network-layer treatment of “flows”

- IPv6 has a much larger address space (i.e. 128 bits)



Src: https://commons.wikimedia.org/wiki/File:Internet_Hosts_Count_log.svg

IPv6 Datagram Format



priority: identify priority among datagrams in flow

128-bit IPv6 addresses

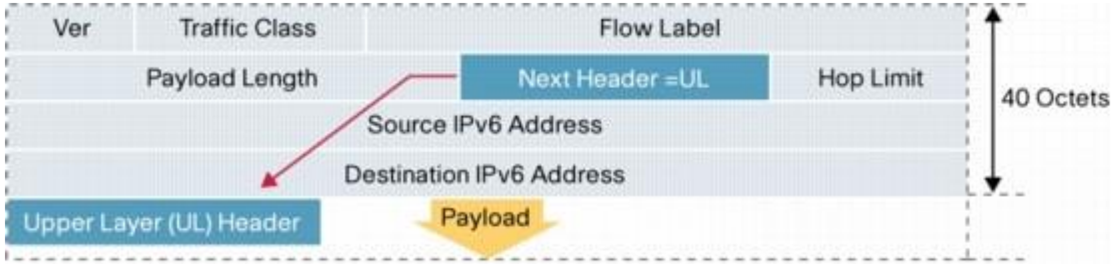
flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

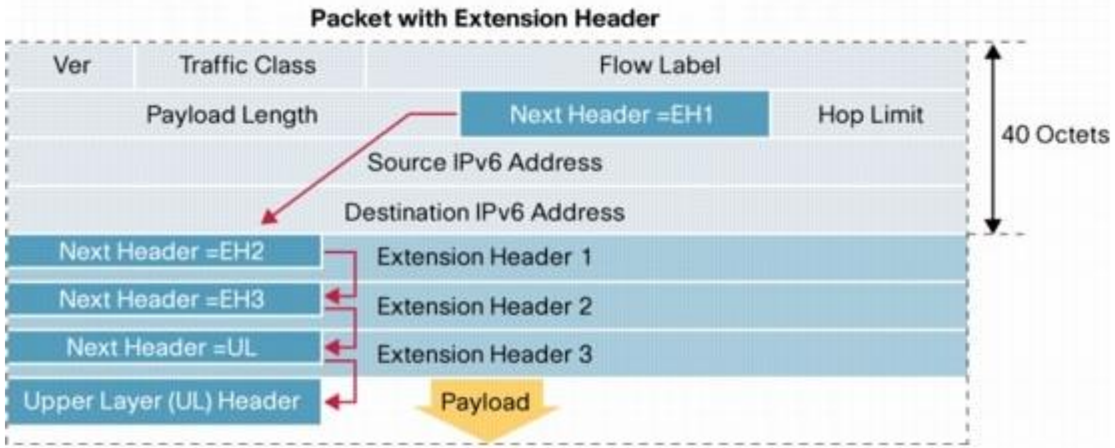
Important features:

- Flow levels for a group of packets
- Better fit for advanced features (e.g. mobility, multicasting, security etc.)

IPv6 Datagram Format

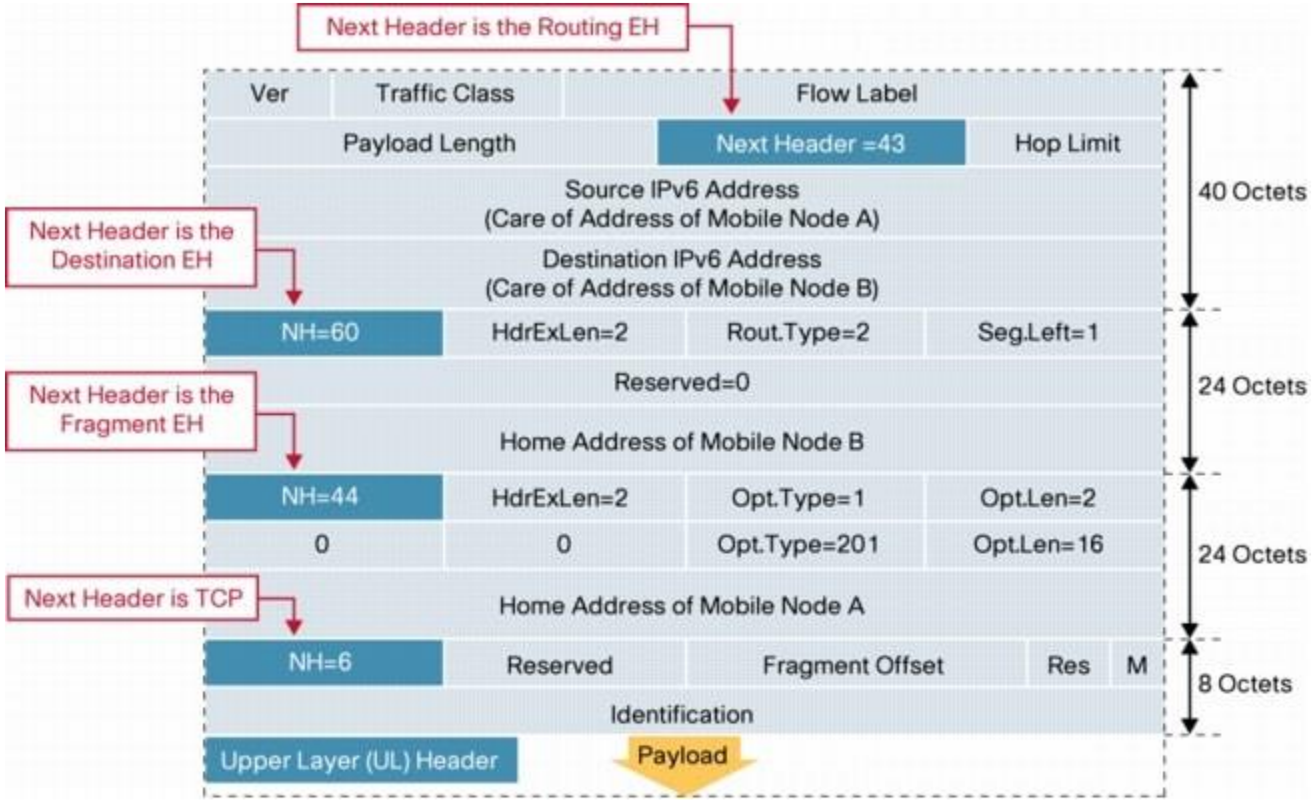
- Extension headers
  - Next header field carries the information of the header following it





Src: https://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html

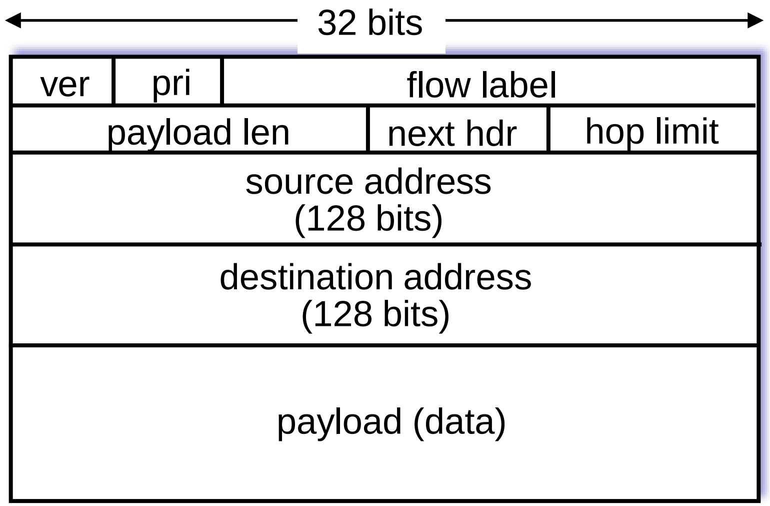
IPv6 Datagram Format



Src: https://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html

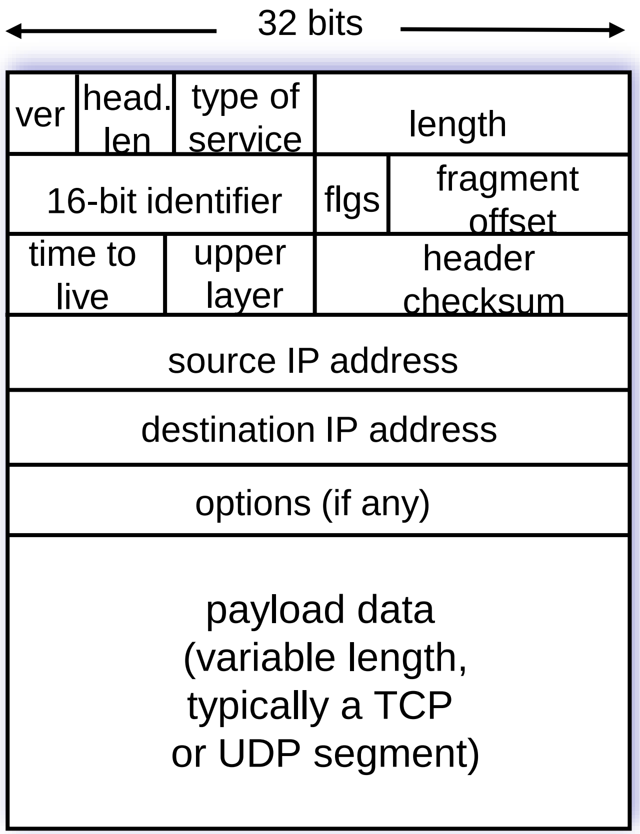
Order	Header Type	Next Header Code
1	Basic IPv6 Header	-
2	Hop-by-Hop Options	0
3	Destination Options	60
4	Routing Header	43
5	Fragment Header	44
6	Authentication Header	51
7	Encapsulation Security Payload Header	50
8	Destination Options	60
9	Mobility Header	135
	No next header	59
Upper Layer	TCP	6
Upper Layer	UDP	17
Upper Layer	ICMPv6	58

IPv6 vs IPv4 Header



What’s missing (compared with IPv4):

- No checksum (to speed processing at routers)
- No fragmentation/reassembly
- No options (available as upper-layer, next-header protocol at router)



IPv6 Addressing

- IPv6 has a much larger address space (i.e. 128 bits)
  - Consists of 8 groups of 4 hex digits (i.e. 16 bits)
- Can be written in compact format
  - Omit leading zeros
  - Omit groups of zeros
- Example: 2001:0db8:0000:0000:0000:ff00:0042:b239

IPv6 Addressing

- IPv6 has a much larger address space (i.e. 128 bits)
  - Consists of 8 groups of 4 hex digits (i.e. 16 bits)
- Can be written in compact format
  - Omit leading zeros
  - Omit groups of zeros
- Example: 2001:0db8:0000:0000:0000:ff00:0000:b239

IPv6 Addressing

- IP addresses have structure:
  - Network part: devices in same network have common high order bits
  - Host part: remaining low order bits

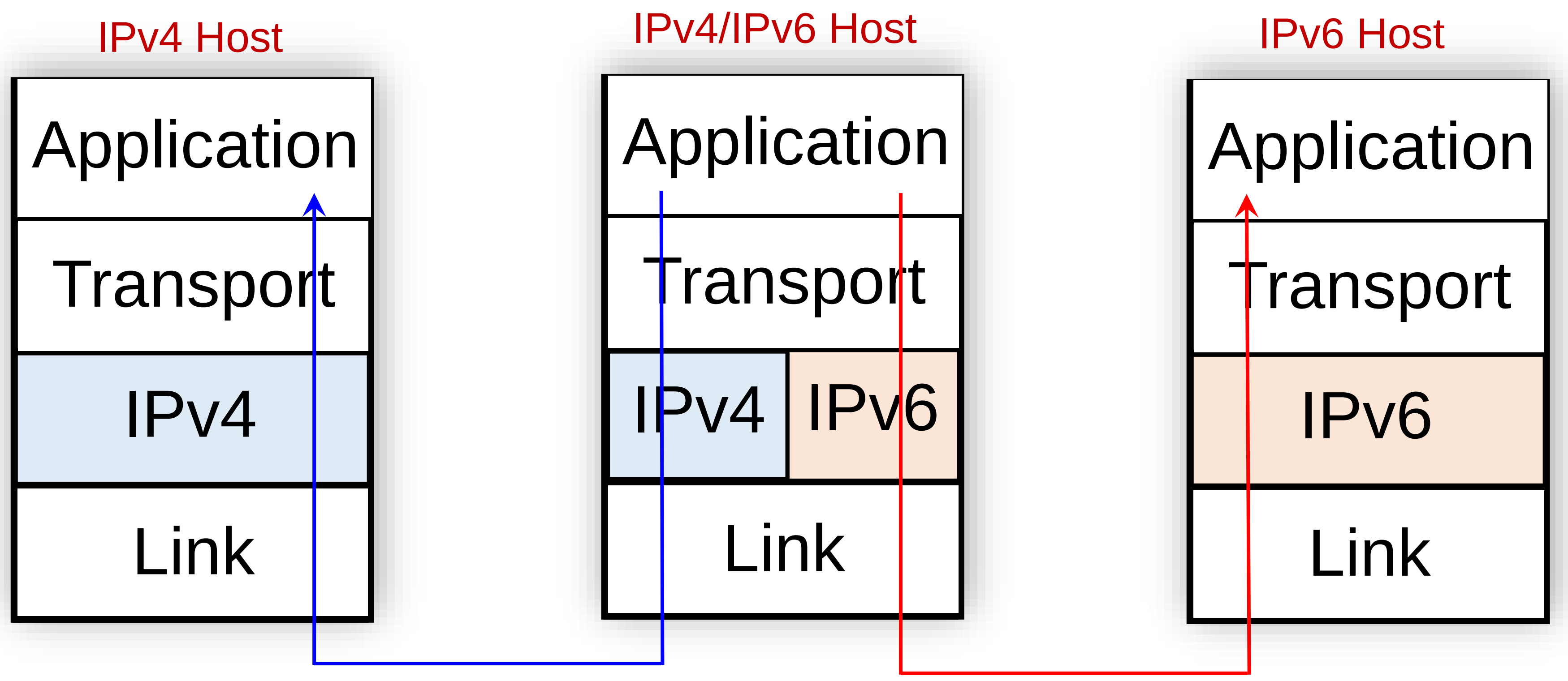
- Example: 2001:0db8:0000:0000:0000:ff00:0000:b239/64

Transition from IPv4 to IPv6

- IPv6 is fundamentally different than IPv4
- Not all routers can be upgraded simultaneously
  - How will the network operate with mixed IPv4 and IPv6 routers?

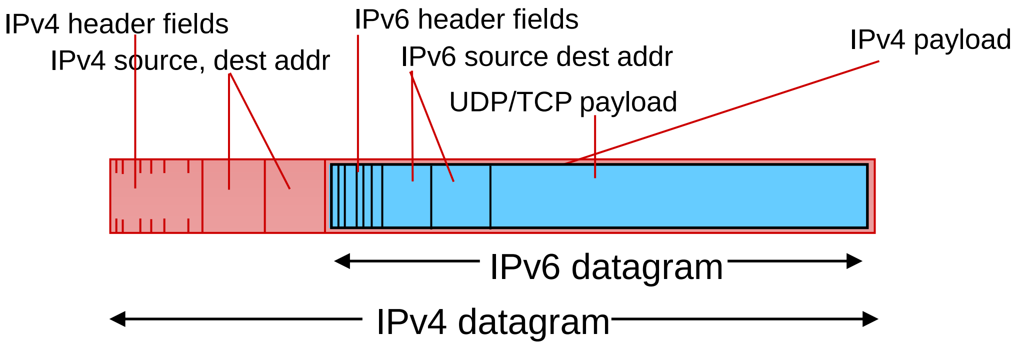
- Well known approaches:
  - Dual stack (supports both IPv4 and IPv6)
  - Tunneling (carries IPv6 over IPv4)

Dual Stack IP Implementation



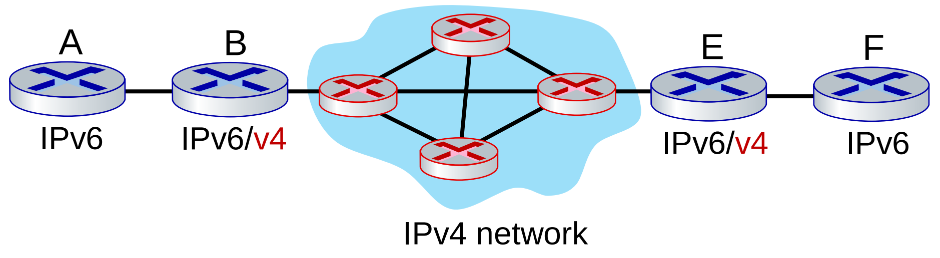
Tunneling and encapsulation

- Tunneling: IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers ("packet within a packet")
  - Tunneling used extensively in other contexts (4G/5G)

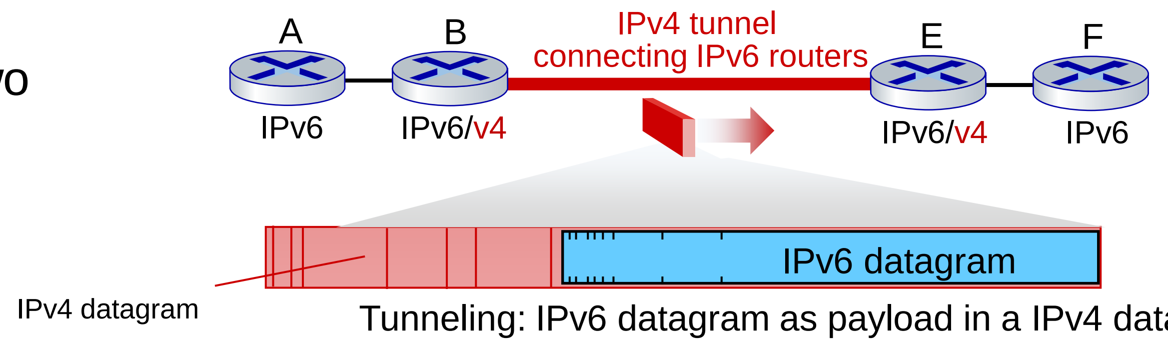


Tunneling and Encapsulation

IPv4 network connecting two IPv6 routers

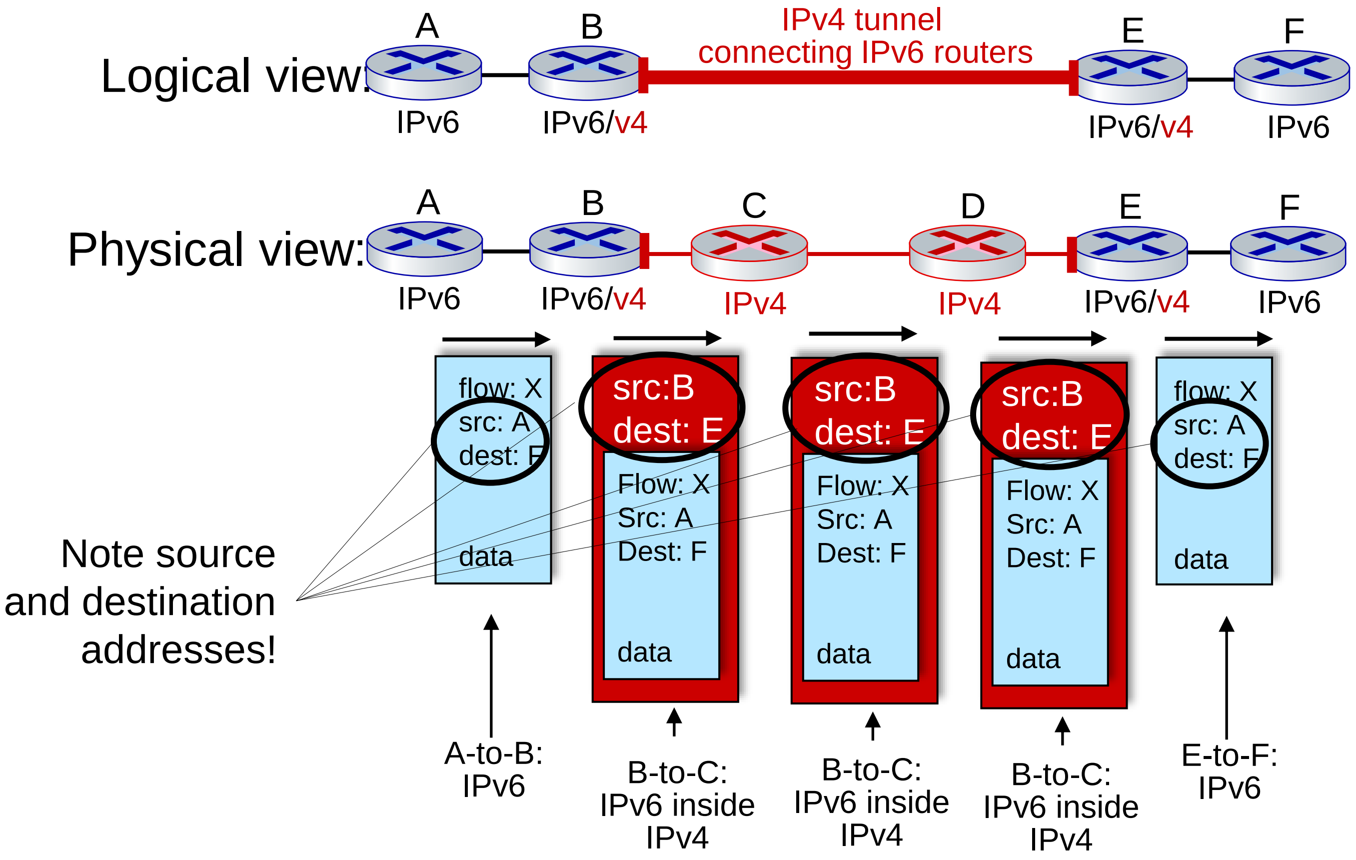


IPv4 tunnel connecting two IPv6 routers



Tunneling: IPv6 datagram as payload in a IPv4 datagram

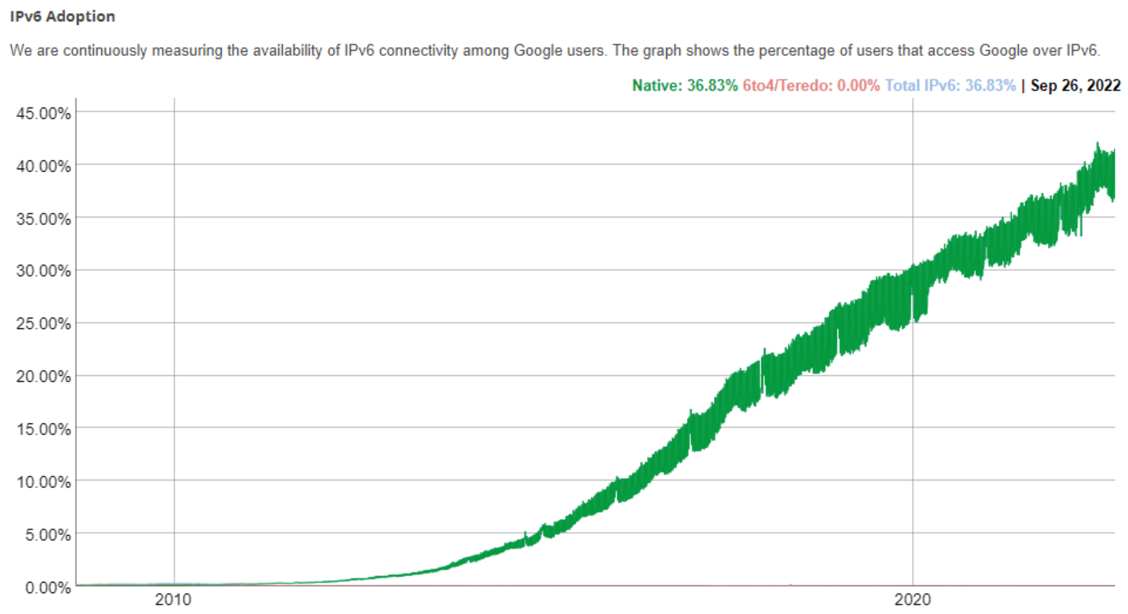
Tunneling



Logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
Physical view:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
Note source
and destination
addresses!
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

IPv6: Adoption

- Google: ~40% of clients access services via IPv6 (Sep, 2022)
- NIST: 1/3 of all US government domains are IPv6 capable



Src: https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption

Summary

- **Network Address Translation:**
  - Helps IPv4 address space exhaustion

- **IPv6 Addressing:**
  - IPv6 datagram format
  - Tunnelling and Encapsulation