Other approaches to solve address depletion problem:

• IPv6

→ 128-bit addresses

→ proposed mid-1990s

→ IPv6 overhead and complexity

→ resistance to wide-spread adoption

→ IPv4 still dominant

IPv4 has found real-world workarounds limiting necessity of IPv6 deployment

→ IPv6 has found niche in ISP core/periphery

→ e.g., periphery: cellular devices

IPv6 header format:

<table>
<tr><td>version
4</td><td>traffic class
8</td><td colspan="3">flow label
20</td></tr>
<tr><td colspan="3">payload length
16</td><td>next header
8</td><td>hop limit
8</td></tr>
<tr><td colspan="5">source address
128</td></tr>
<tr><td colspan="5">destination address
128</td></tr>
</table>

- traffic class: similar role as TOS field in IPv4
- flow label: flow label + source address

  → per-flow traffic management

  → significant extra bits

  → header size twice as large: 40 bytes

- next header: similar to IPv4 protocol field

  → plus double duty for option headers

  → integrated with IPsec: authentication, encryption

- hop limit: same role TTL
- missing fields

  → fragmentation header optional: only allowed at source

Network/socket programming

→ slight differences compared to IPv4

Key features of IPv4 global Internet:

• Classless (vs. classful) IP addressing

→ variable length subnetting

→ that is, $a.b.c.d/x$ (x: mask length)

→ e.g., 128.10.0.0/16, 128.210.0.0/16, 204.52.32.0/20

Prefix specifies organization: autonomous system

→ IPv4 and IPv6 addresses allocated to autonomous systems

→ Purdue University: ASN 17

→ AT&T: ASN 7018 (and others)

→ used in inter-domain routing

→ CIDR (classless inter-domain routing)

→ de facto global Internet addressing standard

- Dynamically assigned IP addresses

→ share an IP address pool

→ addresses are temporary

→ used in access ISPs, enterprises, home networks, etc.

→ e.g., WiFi hot spots

Past: Internet access ISPs exploit that only a fraction requires global Internet access at the same time

→ serve large customer base with small IP address space

Today: residential customers expect to stay online continually

→ limited saving effect of IP address space

DHCP (Dynamic Host Configuration Protocol):

→ UDP-based client/server: server port 67, client port 68

→ 4-way handshake (called DORA)

- Discovery: DHCP client broadcasts request

  → destination IP 255.255.255.255, MAC address all 1's, source IP 0.0.0.0

- Offer: DHCP server responds with IP address (and other relevant info)

  → client's MAC address (Ethernet type field 0x0800, IP protocol field 17)

  → UDP payload contains DHCP packet

- Request: client accepts offered IP address

- Ack: server confirms assignment

- Network address translation (NAT)

→ use of both permanent private IP address and dynamic public IP address

→ address translation from private IP to public IP when accessing global Internet

→ useful for enterprise networks, home networks, etc.

In practice: additional name translation layer

→ configure local DNS (Domain Name System) server with private IP addresses

→ local machines can communicate with each other using symbolic names

→ DNS: global distributed name resolution database system

Example: recent change at Purdue to assign private addresses to lab and instructional machines

→ amber01.cs.purdue.edu: 10.168.53.10

→ borg01.cs.purdue.edu: 10.168.53.41

→ data.cs.purdue.edu: 128.10.2.13

When amber01 accesses global Internet

→ e.g., run web browser

→ 10.168.53.10 translated to 128.10.127.250

Note: amber01.cs.purdue.edu not meaningful outside of Purdue

→ only Purdue's DNS server configured to translate amber01.cs.purdue.edu to 10.168.53.10

. . . IPv4 Address depletion problem remains

- NAPT: NAT + port number

→ variant of NAT: borrow src port field as address bits

Ex.: 192.168.10.10 and 192.168.10.11 both map to single public address 128.10.27.10; in addition

→ 192.168.10.10 maps to 128.10.26.10:6001

→ 192.168.10.11 maps to 128.10.26.10:6002

What about port numbers of 192.168.10.10 and 192.168.10.11?

→ e.g., client process bound to 192.168.10.10:22222

→ e.g., client process bound to 192.168.10.11:33333

Does not matter: NAPT translation table entries

→ 192.168.10.10:22222 maps to 128.10.26.10:6001

→ 192.168.10.11:33333 maps to 128.10.26.10:6002

Example:

if 192.168.10.10:22222 is a web browser (say Firefox) downloading web page from https://www.purdue.edu:443

→ web server knows client as 128.10.27.10:6001

NAPT yields huge increase in effective IP address space

→ IP address bits are increased to 48 (= 32 + 16)

→ biggest factor preventing IP address depletion

Well-suited for systems with asymmetric traffic

→ problem when running servers

→ in general, need permanent IP addresses

Methods to host servers/peers behind DHCP gateway
→ NAPT traversal problem

- Proxies

→ e.g., Internet telephony service: clients contact well-known server—server knows their dynamic addresses

→ server informs client its peer's dynamic IP address and port number

→ peers talk directly to each other

→ called UDP hole punching (can be extended to TCP)

- Relays

→ peers communicate through intermediary

→ full-fledged service: VPN (virtual private network)

- Gateway configuration protocols

→ e.g., UPnP IGD (Internet Gateway Device)

→ limitations

CIDR and dynamically assigned IP addresses with NAPT alleviated IPv4 address depletion problem

→ significant increase of Internet's effective address space

→ IPv4 still dominant

Last free IPv4 address block allocated by IANA (suborganization of ICANN) to regional registries early 2011

→ RIRs: ARIN, RIPE, APNIC, LACNIC, AFRINIC

Last available/recovered address pool allocated mid-2014

→ from central Internet authorities to autonomous systems

→ ISPs manage their own address blocks

IPv6 has found foothold in ISP intranet and explosion of customer premises equipment (e.g., mobile devices)

$\rightarrow$ struggled to find relevance since introduction in mid-1990s

$\rightarrow$ draft standard 1998

$\rightarrow$ full standard 2017

IPv6 benefit in ISP environment

$\rightarrow$ no need for NAT for mobile-to-mobile and mobile-to-IPv6 server communication

In general

$\rightarrow$ dual stack overhead

$\rightarrow$ e.g., my cell phone: 12.75.17.199, 2600:387:11:39d::e (with a few bit flips)