Anycast

- Communicate with “any” one of a set of nodes
- Can do this with DNS

$ dig www.google.com

;; ANSWER SECTION:
www.l.google.com. 300 IN A 74.125.19.103
www.l.google.com. 300 IN A 74.125.19.104
www.l.google.com. 300 IN A 74.125.19.147
www.l.google.com. 300 IN A 74.125.19.99

Anycast at IP layer

- DNS allows anycast through name → address mappings
- Sometimes we need it at layer 3 itself
  - Single IP address refers to multiple hosts
  - Need to talk to any one of them
- Example: DNS root servers
  - Would like to scale number of root servers with Internet
  - Can’t use DNS (remember root servers hard-coded)
  - Want to query closest root server

Anycast in Forwarding Table

- Remember, forwarding is longest-prefix-match
- An anycast address is a /32 address
- A single router may have multiple entries for the address
- Anycast best used in services where separate packets might go to different destinations

The Cost

- A /32 routing entry!
- Multiple /32 routing entries!

Further Advantages

- Geographic scoping
- Distributed Denial of Service (DDoS)
  - Load from DDoS is distributed across many anycast nodes
- F root server (192.5.5.241) now in 46 locations!

I think we have a problem

- Projected use of /8 blocks
- From “A Pragmatic Report on IPv4 Address Space Consumption,” Tony Main, Cisco Systems.

Try the following:

dig +norec @f.root-servers.net hostname.bind chaos txt
IPv6

- Work started in 1994
- Basic protocol published in 1998 [RFC 2460]
- Brief lull, the progress in 2003-6
- Hard push within IETF today for adoption

IPv6 Addresses [RFC 4291]

<table>
<thead>
<tr>
<th>n bits</th>
<th>128-n bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet prefix</td>
<td>interface ID</td>
</tr>
</tbody>
</table>

- Written as 8, ‘:’-separated 16-bit hex numbers
  - Example: 2001:470:806d:1:0:0:0:9
  - Can omit a single run of 0s with “::”
  - Use brackets in URLs: http://[2001:470:806d:1::9]:80/
  - Can write low 32-bits like IPv4: 64:ff9b::171.66.3.9
- Like IPv4, specify subnet prefix with ‘/’
  - E.g., 2001:db8:122:344::/64
- Most IPv6 networks use 64-bit subnet prefix, and end users should receive multiple /64s [RFC 6177]

IPv6 Key Features

- 128 bit addresses
  - Autoconfiguration
- Simplifies basic packet format through *extension headers*
  - 40 byte “base” header
  - Make uncommonly used fields optional

IPv6 multicast addresses

<table>
<thead>
<tr>
<th>8</th>
<th>4</th>
<th>4</th>
<th>112 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111</td>
<td>0RPT</td>
<td>scop</td>
<td>group ID</td>
</tr>
</tbody>
</table>

- T: 1 = *transient*, 0 = group ID assigned by IANA
- P: 1 = address embeds global IPv6 prefix (T must also be 1)
- R: 1 = (requires T = P = 1) encodes rendezvous point
- Scope 1 = interface-local, 2 = link-local, …
- Some groups assigned by IANA:
  - ff02::1 = all nodes, ff02::2 = routers, ff02::1:1 = DHCP
  - ff02::1:ffxx:yyyy = nodes w/ unicast address …xx:yyyy
- Send to Ethernet address 33:33:*low-4-bytes-of-IP*

IPv6 address allocation

- Normal global unicast addresses start 2000::/3
  - IANA doles out *unicast prefixes* to RIRs
- A few other *special prefixes* are assigned
  - :: (all 0s) is unspecified address, ::1 is localhost
  - Rest of 0::/8 used for IPv4 compatibility
  - fc00::/7 used for local addresses [RFC 4193] (kind of like IPv4 addresses 10/8, 172.16/20, 192.168/16 [RFC 1918])
  - fe80::/10 used for link-local addresses
  - ff00::/8 used for multicast
- Over 85% of address space reserved
  - In the unlikely event we exhaust 2000::/8, can be more parsimonious with some other slice

Deriving interface IDs from Ethernet addr

- A 48-bit Ethernet MAC address looks like this:
```
|cccccc0gcccccccc|cccccccc11111111|111110mmmmmmmm|mmmmmmmmmmmmmmmm|
```
  - c is manufacturer’s organizationally unique identifier
  - 0 identifies this as a globally unique address
  - g is 0 for unicast MAC addresses
  - m are address bits assigned by manufacturer
- Convert MAC addr to 64-bit interface ID by flipping 0, sticking hex fffe in middle [RFC 4291]:
```
|cccccc1gcccccccc|cccccccccccccccc|cccccccccccccccc|cccccccccccccccc|
```
- c is manufacturer’s organizationally unique identifier
  - 0 identifies this as a globally unique address
  - g is 0 for unicast MAC addresses
  - m are address bits assigned by manufacturer
Interface IDs in IPv6 addresses

- 64-bit subnets allow use of derived interface IDs
  - Using Ethernet address reduces the need for DHCP
  - Manually assigned addresses (with global bit 0) won’t conflict with ones derived from Ethernet addresses
  - E.g., use interface ID 1 for default router, won’t conflict with any derived interface IDs

- Link-local subnet fe80::/64 is important
  - Means you are guaranteed an address on every interface
  - Look on your machine… ifconfig will show IPv6 address
  - But can’t route to fe80::/64 without knowing interface

IPv6 Header [RFC 2460]

<table>
<thead>
<tr>
<th>Ver</th>
<th>Class</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 bits: 6 for IPv6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 bits: like TOS in IPv4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 bits: identifies a flow [RFC 6437], but not really used yet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 bits: datagram length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 bits: more later</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 bits: like TTL in IPv4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Certain packets (e.g., redirect) must have Hop limit 255</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ensures will be ignored if not from local net</td>
<td></td>
</tr>
</tbody>
</table>

- Addresses: 128 bits each

IPv6 API [RFC 3493]

```c
struct sockaddr_in6 {
    sa_family_t sin6_family; /* AF_INET6 */
    in_port_t sin6_port; /* transport layer port */
    uint32_t sin6_flowinfo; /* IPv6 flow information */
    struct in6_addr sin6_addr; /* IPv6 address */
    uint32_t sin6_scope_id; /* set of interfaces for a scope */
};
```

- sin6_scope_id specifies interface
  - New library calls if_nametoindex, etc., to get values
- In address conversion, specify interface w. ‘%’
  - E.g., ping6 fe80::230:48ff:fe8e:d7a0%eth0

Autoconfiguration [RFC 4862]

- radvd advertises prefixes with ICMP [RFC 4861]
  - Program run by one or more routers on link
  - Lets clients be configured without running DHCP
  - But ICMP message also has bit to say DHCPv6 available
- ICMP contains prefixes + per-prefixed info:
- Valid lifetime and preferred lifetime:
  - Longer valid than preferred lets address become deprecated
- Autonomous config bit
  - 1 means receiving kernel immediately assigns address based on prefix and derived interface ID
- On-link bit – says whole prefix reachable on link

Prefixes vs. links

- In IPv4, address/prefix says what’s on link
  - E.g., Your address is 192.168.1.101/24
  - Sending to 192.168.1.102? Use ARP to get link-level address
  - Sending to 192.168.2.102? Send to router
- IPv6 decouples address from what’s on-link
  - E.g., can reach on-link prefixes without address in prefix
  - Can advertise prefix for autoconfig, but not on-link
  - ICMP redirects can tell you about more on-link nodes
Neighbor discovery [RFC 4861]
- Recall IPv4 uses ARP to get Ethernet from IP addr
- IPv6 doesn’t need a non-IP protocol like ARP
- If address is known on-link, use ICMP
  - Recall multicast address ff02::1:ffxx:yyzz (solicited node address) goes to all nodes with addresses ending ...xx:yyzz
  - Hence they listen to link-layer address 33:33:ff:xx:yy:zz
  - Multiple addrs, same interface ID? Just one 33:33: addr
  - Also learn of neighbors from their traffic
- Nodes aggressively time out valid neighbor info
  - If prefix not on-link, send to router again
  - Router can always send you a redirect

Example Next Header Values
- Assigned from same space as protocol numbers
- 0: Hop-by-hop options header
- 4: IPv4
- 6: TCP
- 17: UDP
- 41: IPv6
- 43: Routing header
- 44: Fragmentation header
- 58: ICMPv6
- 60: Destination options

Extension Headers
- Two types: destination and hop-by-hop
- Both have a next header byte
- Destination headers: intended for IP endpoint
  - Fragment header, Routing header (loose source routing)
  - Destination options
- Hop-by-hop options header (type 0)
  - Must be first extension header in packet
  - Contains series of options for processing at each forwarder
  - Unknown options handled depending on top two bits of 8-bit option type (ignore, drop packet, send ICMP)
  - 3rd bit says if option can change in transit

MTU Requirement
- IPv4 requires a 576-byte link MTU
- IPv6 requires 1280-byte MTU
- If link MTU is smaller, then it MUST support sub-IP fragmentation and assembly to provide a 1280-byte MTU
- It SHOULD provide a 1500-byte MTU; nodes MUST receive 1500 byte packets

Fragmentation Revisited
- High-loss links (e.g., wireless) can be a problem
- 10-hop route, each link has a 10% drop rate (90% success rate)
  - Probability one fragment arrives is $0.9^{10} \approx 35\%$
  - Each fragment is transmitted
    \[ 1 + 0.9 + 0.9^2 + 0.9^3 \ldots 0.9^9 \approx 6.5 \text{ times along the route} \]
  - 100% chance on first hop, 90% on second hop, 81% on third hop, etc.

Fragmentation Revisited, Continued
- If a packet has four fragments, delivery probability is $0.35^4 \approx 1.4\%$
- Total transmissions/delivery = $\frac{1}{0.014} \cdot \sum_{i=0}^{9} 0.9^i$
- Total transmissions/delivery = 65 · 6.5 = 423
- Fragmentation header in IPv6 is a destination header
  - Fragmentation is possible, but must be done at the source
Link-layer reliability

- High-loss link layers usually have single-hop acks and retransmissions
  - End-to-end argument: when can layer 2 reliability fail end-to-end?
- 10-hop route, each link has a 10% drop rate
  - Expect $\frac{10}{10} \approx 1.1$ transmissions/link
  - 10 links, 11 transmissions
  - 44 transmissions/delivery

v4 API Interoperability

- How to make code work w. IPv4 and IPv6?
  - Client side relatively simple
    - getaddrinfo can return IPv6 or IPv4 address [RFC 3493]
    - A1_ADDRCONF flag says only return addresses in family $n$
    - Just create socket of right type when connecting
- Or use IPv4-mapped IPv6 addresses [RFC 4038]
  - getaddrinfo returns (w. A1_V4MAPPED), & works for servers
  - Disabled by IPV6_V6ONLY socket option

Tunneling IPv6 over IPv4

- How to join IPv6 given IPv4 connection?
  - Idea: Tunnel IPv6 inside IPv4 packets [RFC 4213]
    - IP protocol 41 is IPv6—lets you put IPv6 packet as payload of IPv4 (or IPv6) packet
    - Known as Simple Internet Transition or sit tunnels

How can an ISP deploy IPv6?

- Already invested in expensive IPv4 infrastructure
  - Too expensive to replace core routers (even if they claim to support v6, might be too slow or buggy)
- What if ISP controls Customer Edge (CE) routers?
  - Often embedded Linux boxes using software routing, so can just upgrade the software
- 6rd (rapid deployment) provides IPv6 on IPv4 infrastructure [RFC 5569][RFC 5969]
  - ISP Free offered 1.5M customers IPv6 in 5 weeks using 6rd
  - Changed CE software, installed border relays (BRs), but left their expensive v4 infrastructure untouched

Tunneling IPv6 over IPv4

- Can get account from a tunnel broker [RFC 3053]
  - Multiple free options available, e.g.
    - Hurricane Electric
    - Sixxs
  - Easy to setup
    - Run radvd on one machine
    - Most machines support IPv6, will start using it when they get route advertisement ICMPs
    - Instant benefit for reaching machines behind NAT w. v4

How 6rd works

- Embed customer’s IPv4 address in IPv6 address
  - 6rd prefix | IPv4 address | subnet ID | interface ID
  - 6rd prefix is globally routable prefix belonging to ISP
  - E.g., 32-bit prefix + 0-bit subnet ID = 1 /64 per customer
- CE routers statelessly translate v6->v4
  - Embed full IPv6 packet in a v4 packet, using sit
  - Send sit packet to nearest BR using IPv6 anycast (e.g., might use address in 192.88.99.0/24 [RFC 3068])
- BRs determine incoming sit IPv4 from IPv6 dest
6rd and [RFC 1918] private addresses

- What if ISP has more customers than IPv4 addr?
- E.g., say multiple domains all re-use 10.0.0.0/8
  - Then don’t need first octet of IPv4
- Configure CE with IPv4MaskLen
  - Number of bits that get stripped off IPv4 address when embedded in IPv6
  - Use extra bits to encode address domain in v6 address
- DHCP option 212 lets CE routers learn parameters

IPv4 Status

- IANA is already out of IP addresses

![IPv4 Status Graph](image)

A Market in Addresses?

- A market won’t solve the problem, though
- IPv4 addresses can’t be legally owned in some regions
- Trading will further fragment the address space

Internet of Things

- Increasing connectivity: wireless controllers, light switches, etc.
- Home area networks, personal area networks
- Today: vertically integrated, separate technologies
- Goal: connect them with IP
- Imagine every light has an IP address…

6lowpan and the Internet of Things

![6lowpan Diagram](image)
6lowpan

- IETF working group on IPv6 for low-power personal area networks (PANs)
- Tiny, energy constrained, wireless devices: smart homes, ubiquitous computing
- Link layers have tiny MTUs: (802.15.4 is 127 bytes)
- [RFC 4944]

6lowpan Header Compression

- 6lowpan tries to compress common cases: TCP, UDP, etc.
- Example: address compression
  - 6lowpan must allow full 128-bit addresses
  - Address fields alone are 32 bytes!
  - But often they can be shortened…

- [RFC 6282]

6lowpan Compression Flags

- SAC: Source address (stateful?)
- DAC: Destination address (stateful?)
- SAM/DAM: compression scheme used, for stateless:
  - 00: Full 128 bit address
  - 01: 64-bit address, other 64 are link-local prefix padded with zeros
  - 10: 16-bit address, other 112 are as above
  - 00: 0-bit address, 64-bit link local prefix + 64-bit link layer address

Multicast

- Problem: want to send a packet to many nodes
  - Examples: IP-TV, large audio stream
- Using \( n \) unicast packets means the same packet can traverse a single link many times

Multicast Approach

- Nodes can join a multicast group
- Denoted by a multicast IP address
- Routers build a routing topology
  - Link state vs. distance vector
- IGMP: Internet Group Management Protocol
  - Protocol for hosts to manage membership in multicast groups
  - Hosts talk to local multicast routers
Example: Link State Tree

- Routers exchange link state
- Node advertise presence in group
- Routers compute shortest-path multicast tree
- Very expensive!

Practical considerations

- Multicast protocols end up being very complex
- Introduce a lot of router state
- Turned off on most routers
- Used within a domain, not between domains
- How does one handle congestion control?