CSCI-1680
Network Layer:
IP & Forwarding
Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti
Administrivia

• Sign up for IP milestone meetings, preferably with your mentor TA on or before next Tuesday (Oct 11)
  – You don’t need to show an implementation, but you are expected to talk about your design
  – Look for calendar link on EdStem

• HW2: Out after class
Today

More topics on IP forwarding

- IP and the link layer
- Network Address Translation (NAT)
- DHCP
- Next 2 classes: Routing (ie, RIP)
What happens at the link layer?

What does it mean to send to IF1?

<table>
<thead>
<tr>
<th>Prefix</th>
<th>IF/Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.1.0/24</td>
<td>IF1</td>
</tr>
<tr>
<td>1.2.2.0/24</td>
<td>IF2</td>
</tr>
<tr>
<td>8.0.0.0/30</td>
<td>IF0</td>
</tr>
<tr>
<td>Default</td>
<td>8.0.0.2</td>
</tr>
</tbody>
</table>
ARP: Address resolution protocol

• Maps IP addresses to mac addresses
  – Request: “Who has 1.2.3.4?”
  – Response: “aa:bb:cc:dd:ee:ff has 1.2.3.4”

• ARP table: hosts cache IP->mac mappings

• Requests send to broadcast address: ff:ff:ff:ff:ff:ff:ff
  – Anyone can respond: problem?
# arp -n

<table>
<thead>
<tr>
<th>Address</th>
<th>HWtype</th>
<th>HWaddress</th>
<th>Flags</th>
<th>Mask</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.17.44.1</td>
<td>ether</td>
<td>00:12:80:01:34:55</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
<tr>
<td>172.17.44.25</td>
<td>ether</td>
<td>10:dd:b1:89:d5:f3</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
<tr>
<td>172.17.44.6</td>
<td>ether</td>
<td>b8:27:eb:55:c3:45</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
<tr>
<td>172.17.44.5</td>
<td>ether</td>
<td>00:1b:21:22:e0:22</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
</tbody>
</table>
End-to-end Principle

- Keep the network layer simple
- Application-specific features/requirements should be implemented by end hosts
  - Reliability, security
  - Application-specific functionality

Why?
- Easier to implement, e.g., reliability with end-to-end view
- Easier for network layer to scale
- Can implement new protocols without changing network
IP challenge: Address space exhaustion

• IP version 4: ~4 billion IP addresses
  – World population: ~8 billion
  – Est. number of devices on Internet (2021): >10-30 billion

• Since 1990s: various tricks
  – Smarter allocations by registrars
  – Address sharing: Network Address Translation (NAT)
  – DHCP
  – Reclaiming unused space

• Long term solution: IP version 6
Obtaining Host IP Addresses - DHCP

- Networks are free to assign addresses within block to hosts
- Tedious and error-prone: e.g., laptop going from CIT to library to coffee shop
- **Solution**: Dynamic Host Configuration Protocol
  - Client: DHCP Discover to 255.255.255.255 (broadcast)
  - Server(s): DHCP Offer to 255.255.255.255 (why broadcast?)
  - Client: choose offer, DHCP Request (broadcast, why?)
  - Server: DHCP ACK (again broadcast)
- **Result**: address, gateway, netmask, DNS server
Problems with DHCP?

• What happens if a random host decides to be a DHCP server?
Story time
About those home routers...
Private IPs

Some IP ranges are reserved:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.0/8</td>
<td>“Loopback” address—always for current host</td>
</tr>
<tr>
<td>10.0.0.0/8</td>
<td></td>
</tr>
<tr>
<td>192.168.0.0/16</td>
<td>Reserved for private internal networks (RFC1918)</td>
</tr>
<tr>
<td>172.16.0.0/12</td>
<td></td>
</tr>
</tbody>
</table>

- Many networks will use these blocks internally
- These IPs should never be routed over the Internet!
  - What would happen if they were?
Network Address Translation

- What happens when hosts need to share an IP address?
- How to map private IP space to public IPs?
Network Address Translation (NAT)

• Despite CIDR, it’s still difficult to allocate addresses ($2^{32}$ is only 4 billion)
• NAT “hides” entire network behind one address
• Hosts are given private addresses
• Routers map outgoing packets to a free address/port
• Router reverse maps incoming packets
• Problems?
Problems with NAT

• Breaks end-to-end connectivity!
• Technically a violation of layering
• Need to do extra work at end hosts to establish end-to-end connection
  – VoIP (Voice/Video conferencing)
  – Games
NAT Traversal

Various methods, depending on the type of NAT

Examples:
• ICE: Interactive Connectivity Establishment (RFC8445)
• STUN: Session Traversal Utilities for NAT (RFC5389)

One idea: connect to external server via UDP, it tells you the address/port
Internet Control Message Protocol (ICMP)

- Echo (ping)
- Redirect
- Destination unreachable (protocol, port, or host)
- TTL exceeded
- Checksum failed
- Reassembly failed
- Can’t fragment
- Many ICMP messages include part of packet that triggered them
- See [http://www.iana.org/assignments/icmp-parameters](http://www.iana.org/assignments/icmp-parameters)
ICMP message format

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

depends on type/code

20-byte IP header
(protocol = 1—ICMP)
**Example: Time Exceeded**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |

<table>
<thead>
<tr>
<th>Type = 11</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>unused</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- IP header + first 8 payload bytes of packet that caused ICMP to be generated

**Discussion:**

- Code usually 0 (TTL exceeded in transit)
- Discussion: traceroute
So what happened when we ran out of IPv4 addresses?

- It’s not completely gone just yet, but close
- Address block fragmentation
  - Secondary market for IPv4
  - E.g., in 2011 Microsoft bought >600K US IPv4 addresses for $7.5M
- NATs galore
  - Home NATs, carrier-grade NATs
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 was happening (late 2012)
June 6th, 2012
Transition is not painless


You may want to begin with our "Where Do I Start?" page where we have guides for:

- Network operators
- Developers
- Content providers / website owners
- Enterprise customers
- Domain name registrars
- Consumer electronics vendors
- Internet exchange point (IXP) operators

• Why do each of these parties have to do something?
IP version 6

<table>
<thead>
<tr>
<th>Version</th>
<th>Traffic class</th>
<th>Flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload length</td>
<td>Next header</td>
<td>Hop limit</td>
</tr>
</tbody>
</table>

Source address

Destination address

128-bit addresses!
Eg. 2600:3c03::f03c:91ff:fe6e:e3e1
IPv6 Adoption

At Google:

IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.

Native: 36.43% 6to4/Teredo: 0.00% Total IPv6: 36.43% | Feb 6, 2022
At Brown

Wi-Fi

Configure IPv4: Using DHCP
IPv4 Address: 10.3.142.223
Subnet Mask: 255.255.192.0
Router: 10.3.128.1

Configure IPv6: Automatically
Router: fe80::1

IPv6 Address | Prefix Length
-------------|----------------
2620:6e:6000:900:187f:2222:a64f:392a | 64
2620:6e:6000:900:d4d6:81f8:1bc2:97c5 | 64
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

• 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

<table>
<thead>
<tr>
<th>Ver</th>
<th>Class</th>
<th>Flow</th>
<th>Length</th>
<th>Next Hdr.</th>
<th>Hop limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>(16 octets, 128 bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination</td>
<td>(16 octets, 128 bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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: X!!
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
Current State

- IPv6 Deployment picking up
- Most end hosts have dual stacks today (Windows, Mac OSX, Linux, *BSD, Solaris)
- Requires all parties to work!
  - Servers, Clients, DNS, ISPs, all routers
- IPv4 and IPv6 will coexist for a long time
Coming Up

• Routing: how do we fill the routing tables?
  – Intra-domain routing: Tuesday, 10/4
  – Inter-domain routing: Thursday, 10/6
Example

# ARP table

<table>
<thead>
<tr>
<th>Address</th>
<th>HWtype</th>
<th>HWaddress</th>
<th>Flags</th>
<th>Mask</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.17.44.1</td>
<td>ether</td>
<td>00:12:80:01:34:55</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
<tr>
<td>172.17.44.25</td>
<td>ether</td>
<td>10:dd:b1:89:d5:f3</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
<tr>
<td>172.17.44.6</td>
<td>ether</td>
<td>b8:27:eb:55:c3:45</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
<tr>
<td>172.17.44.5</td>
<td>ether</td>
<td>00:1b:21:22:e0:22</td>
<td>C</td>
<td></td>
<td>eth0</td>
</tr>
</tbody>
</table>

# IP route

```
# ip route
127.0.0.0/8 via 127.0.0.1 dev lo
172.17.44.0/24 dev enp7s0 proto kernel scope link src 172.17.44.22 metric 204
default via 172.17.44.1 dev eth0 src 172.17.44.22 metric 204
```