Lecture 11: Addressing
Today’s Lecture

- Addressing in internetworking
  - How long should addressees be?
  - How do you address both hosts and networks?

- How does a router look up where to route a packet?
Router needs to know where to forward a packet

Forwarding table contains:
- List of network names (e.g., LANs) and next hop routers
- Local networks have entries specifying which interface
  » Link-local hosts can be delivered with Layer-2 forwarding

Address of incoming internetwork packet needs to say:
- What is the destination network?
- What is the destination host?
Addressing Considerations

- Fixed length or variable length addresses?

- Issues:
  - Flexibility
  - Processing costs
  - Header size

- Engineering choice: IP uses fixed length addresses
IP Addresses

- 32-bits in an IPv4 address
  - Dotted decimal format a.b.c.d
  - Each represent 8 bits of address

- Hierarchical: Network part and host part
  - E.g. IP address 128.54.70.238
  - 128.54 refers to the UCSD campus network
  - 70.238 refers to the host ieng6.ucsd.edu

- Which part is network (e.g., LAN) vs. host?
Class-based Addressing

- Most significant bits determines “class” of address
  
  **Class A**
  
  Network | Host
  
  127 nets, 16M hosts
  
  **Class B**
  
  Network | Host
  
  16K nets, 64K hosts
  
  **Class C**
  
  Network | Host
  
  2M nets, 254 hosts

- Special addresses
  - Class D (1110) for multicast, Class E (1111) experimental
  - 127.0.0.1: local host (a.k.a. the loopback address)
  - Host bits all set to 0: network address
  - Host bits all set to 1: broadcast address
Subnetting

- Individual networks may be composed of several LANs
  - Only want traffic destined to local hosts on physical network
  - Routers need a way to know which hosts on which LAN

- Networks can be arbitrarily decomposed into subnets
  - Each subnet is simply a prefix of the host address portion
  - Subnet prefix can be of any length, specified with netmask

<table>
<thead>
<tr>
<th>Network</th>
<th>Subnet</th>
<th>Host</th>
</tr>
</thead>
</table>

Prefix

CSE 123 – Lecture 11: Addressing and IPv6
Subnet Addresses

- Every (sub)network has an address and a netmask
  - Netmask tells which bits of the network address is important
  - Convention suggests it be a proper prefix

- Netmask written as an all-ones IP address
  - E.g., Class B netmask is 255.255.0.0
  - Sometimes expressed in terms of number of 1s, e.g., /16

- Need to size subnet appropriately for each LAN
  - Only have remaining bits to specify host addresses
IP Address Problem (1991)

- Address space depletion
  - In danger of running out of classes A and B

- Why?
  - Class C too small for most organizations (only ~250 addresses)
  - Very few class A – very careful about giving them out (who has 16M hosts anyway?)
  - Class B – greatest problem
Classless Inter-Domain Routing (1993)

- Networks described by variable-length prefix and length
- Allows arbitrary allocation between network and host address

<table>
<thead>
<tr>
<th>Network</th>
<th>Host</th>
</tr>
</thead>
</table>

Prefix  Mask=# significant bits representing prefix

- e.g. 10.95.1.2 contained within 10.0.0.0/8:
  - 10.0.0.0 is network and remainder (95.1.2) is host

- Pro: Finer grained allocation; aggregation
- Con: More expensive forwarding table lookup: longest prefix match
Longest Matching Prefix

- Forwarding table contains many prefix/length tuples
  - They *need not* be disjoint!
  - E.g. 200.23.16.0/20 and 200.23.18.0/23
  - What to do if a packet arrives for destination 200.23.18.1?
  - Need to find the longest prefix in the table which matches it (200.23.18.0/23)

- Not a simple table, requires multiple memory lookups
  - Lots and lots of research done on this problem
  - Lots of this work was historically done by UCSD faculty
Summary

- You can’t route efficiently on flat address spaces
  - You’d need a table the size of all hosts on the Internet
  - You’d need to send updates about that table to everyone

- Network-layer addressing is done hierarchically
  - Routing prefix + host suffix
  - Originally, this split was done statically (class-based addressing)
  - Now it is done dynamically (CIDR)
  - Requires more complex forwarding table lookup
  - Allows contiguous chunks of address space to be aggregated (for the purposes of routing) into fewer prefixes
Route Aggregation

- Combine adjacent networks in forwarding tables
  - Helps keep forwarding table size down

```
<table>
<thead>
<tr>
<th>Organization 0</th>
<th>Organization 1</th>
<th>Organization 2</th>
<th>Organization 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>200.23.16.0/23</td>
<td>200.23.18.0/23</td>
<td>200.23.20.0/23</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>
```

```
“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16”
```

CSE 123 – Lecture 11: IPv6
But what if address range is not contiguous?

- Organization 0
  - 200.23.16.0/23
- Organization 2
  - 200.23.20.0/23
- Organization 7
  - 200.23.30.0/23
- Organization 1
  - 200.23.18.0/23

Fly-By-Night-ISP

Send me anything with addresses beginning 200.23.16.0/20

Internet

ISP-R-Us

Send me anything with addresses 200.23.18.0/23
Forwarding example

• Packet to 10.1.1.6 arrives

• Path is R2 – R1 – H1 – H2
Forwarding example (2)

- Packet to 10.1.1.6
- Matches 10.1.0.0/23

### Forwarding table at R2

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>loopback</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.16.1</td>
</tr>
<tr>
<td>10.1.8.0/24</td>
<td>interface1</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>interface2</td>
</tr>
<tr>
<td><strong>10.1.0.0/23</strong></td>
<td><strong>10.1.2.2</strong></td>
</tr>
<tr>
<td>10.1.16.0/24</td>
<td>interface3</td>
</tr>
</tbody>
</table>

---

CSE 123 – Lecture 11: IPv6
### Routing table at R1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>loopback</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.2.1</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>interface1</td>
</tr>
<tr>
<td><strong>10.1.1.0/24</strong></td>
<td>interface2</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>interface3</td>
</tr>
<tr>
<td><strong>10.1.1.4/30</strong></td>
<td>10.1.1.101</td>
</tr>
</tbody>
</table>
Forwarding example (4)

- Packet to 10.1.1.6
- Direct route
  - Longest prefix match

Routing table at H1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>loopback</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.1.1</td>
</tr>
<tr>
<td>10.1.1.0/24</td>
<td>interface1</td>
</tr>
<tr>
<td>10.1.1.4/30</td>
<td>interface2</td>
</tr>
</tbody>
</table>
For Next Time

- Project 2 and Homework 2!
- Midterm not next Monday but the following MONDAY
  - Online on GradeScope:
- Read 3.2.6, 9.3.1 for next Lecture
The space crunch…

- Still running out of IP addresses… what to do?
- Two solutions
  - Network Address Translation – multiple multiple hosts on a single IP address (future class)
  - Get bigger addresses -> IPv6
- IPv6: 128bit addresses… we won’t run out
  - 64bit routing prefix, 64bit host id

An IPv6 address (in hexadecimal)

\[ \text{2001:0DB8:AC10:FE01:0000:0000:0000:0000} \]

Zeroes can be omitted

\[ \text{2001:0DB8:AC10:FE01::} \]

\[ \text{00100000000001:0000110111011100:1010110000010000:1111110000000001:} \]
IPv6 Addresses

- Colon-Hex notation
  - 8 groups of four HEX digits separated by colons, e.g.
    - FEDC:0000:0000:0065:4321:0000:DEAD:BEEF
  - Can drop leading zeros:
  - Can even skip **first** sequence of all zeros w/ ::
  - FEDC::65:4321:0000:DEAD:BEEF
  - Every IPv4 address is a IPv6 address:
  - E.g., ::222.173.190.239 (prepended w/zeros)

- Network names expressed as prefix/length:
  - FEDC::65:43/50
Address Types

- Each interface has multiple different addresses
  - Link local, prefixed with FE80::/10 (1111 1110 10)
    » Used only for communication between adjacent IPv6 devices
    » Packets are NOT forwarded by routers
    » Automatically assigned upon boot
  - Unique local, prefixed with FC00::/7 (1111 110 )
    » Used only internal to one network
    » Not routable on the global Internet
  - Global
    » Like an IPv4 address
## IPv6 vs IPv4 header

<table>
<thead>
<tr>
<th>Ver.</th>
<th>Traffic Class</th>
<th>Flow Label</th>
<th>Payload Length</th>
<th>Source Address</th>
<th>Next Header</th>
<th>Hop Limit</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### IPv6 header:

- **Ver.**: Version number
- **Traffic Class**: Type of service
- **Flow Label**: Flow label
- **Payload Length**: Length of the payload
- **Source Address**: Address of the sender
- **Destination Address**: Address of the recipient

### IPv4 header:

- **Ver.**: Version number
- **Hdr Len**: Header length
- **Type of Service**: Type of service
- **Total Length**: Total length of the packet
- **Identification**: Identification number
- **Flags**: Flags
- **Fragment Offset**: Fragment offset
- **Time to Live**: Time to live
- **Protocol**: Protocol number
- **Header Checksum**: Header checksum
- **Source Address**: Address of the sender
- **Destination Address**: Address of the recipient
- **Options...**: Options

### Differences:

- **Gray bits are unique to each header**
- **Changes**
  - Eliminate fragmentation-related fields
  - Eliminate header checksum
  - Added flow label
  - Quadruple size of addresses
  - IPv6 header (40 bytes) vs IPv4 (20 bytes)
Extension Headers

- Effectively a linked list of headers
  - The “next header” field is the pointer

- Two different types
  - **Destination**, intended for the IP end point. E.g.,
    - 44: Fragmentation Header (it’s baaack!)
    - 43: Routing header (dictates how to route the packet)
  - **Hop-by-hop**, processed by each node on the path
IPv6 Transition is slow

- Need to support both protocols at the same time
  - Complicated… if a destination has both a IPv4 and IPv6 address which to use?
- Less need in developed world -> slower adoption
- That said
  - All major operating systems now support IPv6
  - All major router vendors
  - US Mobile carriers (e.g., Tmobile, Verizon, etc)
  - Offered as option by many US ISPs
- In your lifetime it is likely that IPv6-based addressing will start to dominate
Aside: ICMP

- What happens when things go wrong?
  - Need a way to test/debug a large, widely distributed system

- ICMP = Internet Control Message Protocol (RFC792)
  - Companion to IP – required functionality

- Used for error and information reporting:
  - Errors that occur during IP forwarding
  - Queries about the status of the network
ICMP Error Message Generation

Error during forwarding!

source

IP packet

ICMP

IP packet

dest
Common ICMP Messages

- **Destination unreachable**
  - “Destination” can be host, network, port, or protocol
- **Redirect**
  - To shortcut circuitous routing
- **TTL Expired**
  - Used by the “traceroute” program
    - traceroute traces packet routes through Internet
- **Echo request/reply**
  - Used by the “ping” program
    - ping just tests for host liveness
- **ICMP messages include portion of IP packet that triggered the error (if applicable) in their payload**
ICMP Restrictions

- The generation of error messages is limited to avoid cascades … error causes error that causes error…

- Don’t generate ICMP error in response to:
  - An ICMP error
  - Broadcast/multicast messages (link or IP level)
  - IP header that is corrupt or has bogus source address
  - Fragments, except the first

- ICMP messages are often rate-limited too
  - Don’t waste valuable bandwidth sending tons of ICMP messages