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
Route Aggregation

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

```
200.23.16.0/23
200.23.18.0/23
200.23.20.0/23
200.23.30.0/23
```

```
Organization 0
Organization 1
Organization 2
Organization 7

Fly-By-Night-ISP
```

```
ISPs-R-Us
```

```
Send me anything with addresses beginning 200.23.16.0/20
Send me anything with addresses beginning 199.31.0.0/16
```

```
Internet
```

CSE 123 – Lecture 11: IPv6
Most Specific Route

- 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

ISP-R-Us

Internet
```

“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16 or 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
Forwarding example (3)

- Packet to 10.1.1.6
- Matches 10.1.1.4/30
  - Longest prefix match

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>10.1.1.0/24</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><strong>10.1.1.101</strong></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>
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

IPv6: 128bit addresses… we won’t run out
- 64bit routing prefix, 64bit host id

An IPv6 address (in hexadecimal)

```
2001:0DB8:AC10:FE01:0000:0000:0000:0000
```

Zeroes can be omitted
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

### IPv6 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>IPv6</th>
<th>IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver.</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Traffic Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow Label</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next Header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hop Limit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### IPv4 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver.</td>
<td>4</td>
</tr>
<tr>
<td>Type of Service</td>
<td></td>
</tr>
<tr>
<td>Total Length</td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td></td>
</tr>
<tr>
<td>Flg Fragment Offset</td>
<td></td>
</tr>
<tr>
<td>Time to Live</td>
<td></td>
</tr>
<tr>
<td>Protocol Header Checksum</td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
</tr>
<tr>
<td>Options...</td>
<td></td>
</tr>
</tbody>
</table>

- 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
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
For Next Time

• Finish up Project 1!

• Midterm MONDAY
  • One 8.5x11 sheet of paper with notes
  • Nothing else—no calculators!

• Read 3.2.6, 9.3.1 for next Lecture