Administivia

• Snowcast: was due last night
• HW1: Out now, due next Wed (Feb 23)
• IP Project: Out tomorrow (Feb 18)
  – Fill out group preference form by 11:59pm tomorrow (Feb 18)
Start of network layer

• Network layer: Internet Protocol (IP) (v4)
• Mechanics of IP forwarding
• Intro to IP project
Layers, Services, Protocols

- **Application**
  - Service: user-facing application.
  - Application-defined messages

- **Transport**
  - Service: multiplexing applications
  - Reliable byte stream to other node (TCP),
  - Unreliable datagram (UDP)

- **Network**
  - Service: move packets to any other node in the network
  - Internet Protocol (IP)

- **Link**
  - Service: move frames to other node across link.
  - May add reliability, medium access control

- **Physical**
  - Service: move bits to other node across link
Internet Protocol (IP) Goals

How to connect everyone?
• Glue lower-level networks together
• A network of networks!
Internet Protocol (IP) Goals

How to connect everyone?
• Glue lower-level networks together
• A network of networks!
• Router: device that forwards packets between networks
Internet Protocol (IP) Goals

How to connect everyone?

• Glue lower-level networks together
• A network of networks!
• Router: device that forwards packets between networks

Doesn’t this sound like switching?
Inter-networking Challenges

- Networks are heterogeneous (e.g., Wifi vs. Ethernet)
  - Different frame formats
  - Different service models
  - Different packet sizes/bandwidths
Inter-networking Challenges

- Networks are heterogeneous (e.g., Wifi vs. Ethernet)
  - Different frame formats
  - Different service models
  - Different packet sizes/bandwidths
- Scaling
  - Link-layer forwarding strategies don’t scale to Internet!
Map of the Internet, 2021 (via BGP)
OPTE project
How would you design such a protocol?

- Circuits or packets?
  - Predictability
How would you design such a protocol?

- Circuits or packets?
  - Predictability

- Service model
  - Reliability, timing, bandwidth guarantees
How would you design such a protocol?

• Circuits or packets?
  – Predictability

• Service model
  – Reliability, timing, bandwidth guarantees

• Any-to-any communication
  – How do you find a particular host?
  – How do you get a message there?
  – What happens when a host joins/leaves?
IP’s Decisions
IP’s Decisions

• Packet switched
  – Unpredictability, statistical multiplexing
IP’s Decisions

• Packet switched
  – Unpredictability, statistical multiplexing

• Service model
  – Lowest common denominator: best effort, connectionless datagram
IP’s Decisions

• Packet switched
  – Unpredictability, statistical multiplexing

• Service model
  – Lowest common denominator: best effort, connectionless datagram

• Any-to-any communication
  – IP header: common message format
  – IP address: each host has an address, based on hierarchical structure of network
A Bit of History

• Packet switched networks: Arpanet’s IMPs
  – Late 1960’s
  – RFC 1, 1969!
  – Segmentation, framing, routing, reliability, reassembly, primitive flow control

• Network Control Program (NCP)
  – Provided connections, flow control
  – Assumed reliable network: IMPs
  – Used by programs like telnet, mail, file transfer
A Bit of History

• Packet switched networks: Arpanet’s IMPs
  – Late 1960’s
  – RFC 1, 1969!
  – Segmentation, framing, routing, reliability, reassembly, primitive flow control

• Network Control Program (NCP)
  – Provided connections, flow control
  – Assumed reliable network: IMPs
  – Used by programs like telnet, mail, file transfer

• Wanted to connect multiple networks
  – Not all reliable, different formats, etc…
THE ARPA NETWORK
DEC 1969
Abb. 4 ARPA NETwork, topologische Karte. Stand Juni 1974.
TCP/IP Introduced

- Vint Cerf, Robert Kahn build protocol to replace NCP
- Initial design: single protocol providing a unified reliable pipe
TCP/IP Introduced

- Vint Cerf, Robert Kahn build protocol to replace NCP
- Initial design: single protocol providing a unified reliable pipe
- Different requirements soon emerged, and the two were separated
  - IP: basic datagram service among hosts
  - TCP: reliable transport
  - UDP: unreliable multiplexed datagram service
David D. Clark, “The design Philosophy of the DARPA Internet Protocols”, 1988

• Primary goal: multiplexed utilization of existing interconnected networks
• Other goals:
  – Communication continues despite loss of networks or gateways
  – Support a variety of communication services
  – Accommodate a variety of networks
  – Permit distributed management of its resources
  – Be cost effective
  – Low effort for host attachment
  – Resources must be accountable
Internet Protocol

IP runs on all hosts and routers

- Provides addressing: how we name nodes in an IP network
- Provides forwarding: how routers move packets based on the destination address
- Later: routing: how routers build forwarding rules
IP’s Service Model

- Connectionless (datagram-based)
IP’s Service Model

- Connectionless (datagram-based)
- Best-effort delivery (unreliable service)
  - packets may be lost
  - packets may be delivered out of order
  - duplicate copies of packets may be delivered
  - packets may be delayed for a long time
IP’s Service Model

- Connectionless (datagram-based)
- Best-effort delivery (unreliable service)
  - packets may be lost
  - packets may be delivered out of order
  - duplicate copies of packets may be delivered
  - packets may be delayed for a long time
- It’s the lowest common denominator
  - A network that delivers no packets fits the bill!
  - All these can be dealt with above IP (if probability of delivery is non-zero…)
IP Version 4: Each address is a 32-bit number:

128.148.16.7

10000000 10010100 00010000 00000111

128.148.16.7
IP Addressing

IP Version 4: Each address is a 32-bit number: 128.148.16.7

10000000 10010100 00010000 00000111

128.148.16.7

Notation
- Write each byte ("octet") as a decimal number
- This is called "dotted decimal" or "dotted quad" notation
An IP address identifies...

- Who a host is: A unique number
- Where it is on the Internet

128.148.16.7
An IP address identifies...

- Who a host is: A unique number
- Where it is on the Internet
- Networks are allocated ranges of IP addresses by global authority (ICANN)

*ICANN (Internet Corporation for Assigned Names and Numbers)
IP Addressing

An IP address identifies...

• Who a host is: A unique number
• Where it is on the Internet
• Networks are allocated ranges of IP addresses by global authority (ICANN)
  – Further subdivided by regions, ISPs, organizations...

*ICANN (Internet Corporation for Assigned Names and Numbers)
An IP address identifies…

- Who a host is: A unique number
- Where it is on the Internet

- Networks are allocated ranges of IP addresses by global authority (ICANN)
  - Further subdivided by regions, ISPs, organizations


*ICANN (Internet Corporation for Assigned Names and Numbers)
IP Addressing

Brown owns the range: 128.148.xxx.xxx

Network part
Identifies Brown (to the Internet)

Host part
Denotes individual hosts within the Brown Network

16 BITS

2^16 = 65K HOSTS
IP Addressing

A network can designate IP addresses for its own hosts within its address range.

For 128.148.xxx.xxx:
IP Addressing

A network can designate IP addresses for its own hosts within its address range.

For 128.148.xxx.xxx:

- 0 - 2^55
IP Addressing

A network can designate IP addresses for its own hosts within its address range.

For 128.148.xxx.xxx:

```
10000000 10010100 xxxxxxxx xxxxxxxx
```
A network can designate IP addresses for its own hosts within its address range.

For 128.148.xxx.xxx:

Brown uses the prefix 128.148.0.0/16
A network can designate IP addresses for its own hosts within its address range.

For 128.148.xxx.xxx:

\[
10000000 \ 10010100 \ xxxxxxxx \ xxxxxxxx
\]

Brown uses the prefix 128.148.0.0/16

Some other ways to write this:

128.148/16
128.148.0.0 + subnet mask 255.255.0.0
Common prefixes

1.2.0.0/16
00000001 00000010 xxxxxxxx xxxxxxxx
≈ 65K hosts

8.0.0.0/8
00001000 xxxxxxxx xxxxxxxx xxxxxxxx
≈ 16M hosts

123.10.1.0/24
01111011 00001010 00000001 xxxxxxxx
≈ 256 hosts

201.112.10.200/30
11001001 01110000 00001010 110010xx
≈ 4 hosts
Example

How many addresses are in the network 192.1.0.0/20?

\[
\begin{array}{c}
1100 \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\end{array}
\]

\[
2^{12} = 4096 \text{ hosts}
\]
How do we move packets between networks?
Consider the network 1.2.1.0/24:
Consider the network 1.2.1.0/24:

- For IP, communicating on same network is easy—this is the link-layer’s job!
Consider the network 1.2.1.0/24:

- For IP, communicating on same network is easy—this is the link-layer’s job!
- Need to map IP addresses to MAC addresses (more on this later) <ARP>
IP forwarding

Consider the network 1.2.1.0/24:

• For IP, communicating on same network is easy—this is the link-layer’s job!

• Need to map IP addresses to MAC addresses (more on this later)

• How to reach an address outside this network?
Consider the network 1.2.1.0/24:

- For IP, communicating on same network is easy—this is the link-layer’s job!
- Need to map IP addresses to MAC addresses (more on this later)

To reach other networks, send packets to a router, which forwards IP packets to other networks
Forwarding IP packets
Forwarding IP packets

1.2.1.2

1.2.1.3

1.2.1.200

1.2.1.1

IF0

IF1

IF2
Forwarding IP packets

1.2.1.2
1.2.1.3
1.2.1.200
1.2.1.0/24

1.2.2.100
1.2.2.105
1.2.2.0/24

1.2.1.1
1.2.2.1
IF0
IF1
IF2
Forwarding IP packets

To more networks (ie, Internet)

1.2.1.2
1.2.1.3
1.2.1.200

1.2.1.1
1.2.2.1

1.2.2.100
1.2.2.105
Forwarding IP packets

Src: 1.2.1.3
Dst: 1.2.2.100

To more networks (ie, Internet)
Forwarding IP packets

Src: 1.2.1.3
Dst: 1.2.2.100

To more networks (ie, Internet)
Forwarding IP packets

To more networks (i.e., Internet)

Prefix | Interface
--- | ---
1.2.1.0/24 | IF1
1.2.2.0/24 | IF2
* | IF0

Src: 1.2.1.3
Dst: 1.2.2.100
...
Forwarding IP packets

- **Src:** 1.2.1.3
- **Dst:** 1.2.2.100

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>&lt;everything else&gt;</td>
<td>IF0</td>
</tr>
</tbody>
</table>
What about the rest?

How to reach networks that aren’t directly connected?

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>&lt;everything else&gt;</td>
<td>IF0</td>
</tr>
</tbody>
</table>
What about the rest?

- Need IP of another router that knows about other networks
- This “next hop” IP must be reachable locally!

<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>
What about the rest?

- Need IP of another router that knows about other networks
- This "next hop" IP must be reachable locally!
- "Default" => 0.0.0.0/0 refers to every address
  - Also called a gateway

<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>
The forwarding table

- Exploits hierarchical structure of addresses: know how to reach networks, not individual hosts

<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>

- Table is keyed on a network prefix, not a whole address
- Select best prefix with longest prefix matching (more on this later)
A forwarding table

# 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
The IPv4 Header

<table>
<thead>
<tr>
<th>Bit</th>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>31 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Version</td>
<td>IHL</td>
<td>TOS</td>
<td>Total length</td>
<td>Identification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Source address</td>
<td>Destination address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Options</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

- 0-40 bytes for Options
- Up to 65536 bytes for Options
- 20 bytes for Header checksum
The IPv4 Header

Defined by RFC 791
RFC (Request for Comment): defines network standard
Important fields

- **Version**: 4 for IPv4 packets, 6 for IPv6
- **Destination address**: used for forwarding
- **TTL (time-to-live)**: decremented each hop
  - Can prevent forwarding loops (and do other stuff…)
- **Checksum**: computed over header (very weak!)
- **Protocol identifier**: describes what’s in the packet
  - 6: TCP, 17: UDP, 1: ICMP, …
  - Defines the type of the payload
Less important fields

- Header length: in 32-bit units
  - >5 implies use of IP options
  - Almost all routers ignore IP options
Less important fields

• Header length: in 32-bit units
  – >5 implies use of IP options
  – Almost all routers ignore IP options

• Fragmentation
  – Network can fragment a packet if next link requires a small frame
  – Most routers don’t fragment (or reassemble fragments)
Less important fields

- **Header length**: in 32-bit units
  - >5 implies use of IP options
  - Almost all routers ignore IP options

- **Fragmentation**
  - Network can fragment a packet if next link requires a small frame
  - Most routers don’t fragment (or reassemble fragments)

- **We won’t talk about…**
  - Type of Service (TOS): basic traffic classification
  - Identifier: might have special meaning on some networks
Forwarding mechanics

When an IP packet arrives at a host/router:
Forwarding mechanics

When an IP packet arrives at a host/router:

- Is it valid? Verify checksum over header
Forwarding mechanics

When an IP packet arrives at a host/router:

• Is it valid? Verify checksum over header

• **Is it for me?** If dest IP == your address, send to OS
Forwarding mechanics

When an IP packet arrives at a host/router:

- **Is it valid?** Verify checksum over header
- **Is it for me?** If dest IP == your address, send to OS
- If not, where should it go?
  - Consult forwarding table => find next hop
  - Decrement TTL
  - Send packet to next hop
Traceroute

- When TTL reaches 0, router may send back an error
  - ICMP TTL exceeded
- If it does, we can identify a path used by a packet!
Coming up…

• ARP: Mapping IPs to MAC addresses
• How are addresses assigned?
• NAT: When it gets complicated
• Routing algorithms: how to build forwarding tables
LOCALHOST: 5000

A

192.168.0.2

FORWARDING TABLE

1.2.1.0 - LOCAL
1.2.1.1 : 5001
1.2.3.2 : 5001

B

192.168.0.1

1.2.3.1

C

1.2.3.2

D

192.168.0.3

5003
Coming up…

- ARP: Mapping IPs to MAC addresses
- How are addresses assigned?
- NAT: When it gets complicated
- Routing algorithms: how to build forwarding tables

Fill out the group preference survey for the IP project (announcement soon) by tomorrow (Feb 18) by 11:59PM
Putting it all together...
Putting it all together…

- The more connected a router becomes, the more complex its forwarding table... and the more it may change!

- **Routing algorithms**: routers exchange path information to their forwarding tables (more on this later)
Putting it all together…

• The more connected a router becomes, the more complex its forwarding table… and the more it may change!

• **Routing algorithms:** routers exchange path information to their forwarding tables (more on this later)
Goal: find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>&lt;everything else&gt;</td>
<td>IF0</td>
</tr>
</tbody>
</table>

1.2.2.100

00000001.00000010.00000010.01100100

?= 
Goal: find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>&lt;everything else&gt;</td>
<td>IF0</td>
</tr>
</tbody>
</table>

1.2.2.100: 00000001.00000010.00000010.01100100
1.2.1.0/24: 00000001.00000010.00000001.xxxxxxxx

?=xxxxxxxx
Goal: find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>&lt;everything else&gt;</td>
<td>IF0</td>
</tr>
</tbody>
</table>

1.2.2.100 00000001.00000010.00000010.01100100
1.2.1.0/24 00000001.00000010.00000001.xxxxxxxx
1.2.2.0/24 00000001.00000010.00000010.xxxxxxxx
Goal: find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>&lt;everything else&gt;</td>
<td>IF0</td>
</tr>
</tbody>
</table>

1.2.2.100 00000001.00000010.00000010.01100100

1.2.1.0/24 00000001.00000010.00000001.xxxxxxxx

1.2.2.0/24 00000001.00000010.00000010.xxxxxxxx
Goal: find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>0.0.0.0/0</td>
<td>IF0</td>
</tr>
</tbody>
</table>

1.2.2.100 00000001.00000010.00000010.01100100
?=  1.2.1.0/24 00000001.00000010.00000001.xxxxxxxx
1.2.2.0/24 00000001.00000010.00000010.xxxxxxxx
Goal: find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>0.0.0.0/0</td>
<td>IF0</td>
</tr>
</tbody>
</table>

1.2.2.100: 00000001.00000010.00000010.01100100
1.2.1.0/24: 00000001.00000010.00000001.xxxxxxxx
1.2.2.0/24: 00000001.00000010.00000010.xxxxxxxx
0.0.0.0/0: xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx
**Goal:** find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>0.0.0.0/0</td>
<td>IF0</td>
</tr>
</tbody>
</table>

1.2.2.100  00000001.00000010.00000010.01100100  Output: IF2
1.2.1.0/24  00000001.00000010.00000001.xxxxxxxx
1.2.2.0/24  00000001.00000010.00000010.xxxxxxxx
0.0.0.0/0   xxxxxxxxx.xxxxxxxxx.xxxxxxxxx.xxxxxxxxx

**Longest Prefix Matching (LPM):** can represent entire IP space in (small) table!
Goal: find the most specific (ie, longest) prefix matching the destination

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</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>0.0.0.0/0</td>
<td>IF0</td>
</tr>
</tbody>
</table>

How to reach 1.2.2.100?

1.2.2.100 \(\text{00000001.00000010.00000010.01100100}\) \(?=\)
1.2.1.0/24 \(\text{00000001.00000010.00000001.xxxxxxxx}\)
1.2.2.0/24 \(\text{00000001.00000010.00000010.xxxxxxxx}\)
0.0.0.0/0  \(\text{xxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx}\)

Output: IF2

Longest Prefix Matching (LPM): can represent entire IP space in (small) table!
Some ISP

Brown
128.148.0.0/16

Customer 2
1.3.0.0/16

Customer 3
5.6.128.0/20

Brown’
128.148.100.0/24

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.148.0.0/16</td>
<td>IF1</td>
</tr>
<tr>
<td>1.3.0.0/16</td>
<td>IF2</td>
</tr>
<tr>
<td>5.6.128.0/20</td>
<td>IF3</td>
</tr>
<tr>
<td>128.148.100.0/24</td>
<td>IF4</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>8.0.0.2</td>
</tr>
</tbody>
</table>
Some ISP

Brown
128.148.0.0/16

Customer 2
1.3.0.0/16

Customer 3
5.6.128.0/20

Brown’
128.148.100.0/24

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.148.0.0/16</td>
<td>IF1</td>
</tr>
<tr>
<td>1.3.0.0/16</td>
<td>IF2</td>
</tr>
<tr>
<td>5.6.128.0/20</td>
<td>IF3</td>
</tr>
<tr>
<td>128.148.100.0/24</td>
<td>IF4</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>8.0.0.2</td>
</tr>
</tbody>
</table>

Dst: 128.148.105.207

Dst: 128.148.100.104
show route table inet.0 active-path

inet.0: 866991 destinations, **13870153** routes (866991 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Metric</th>
<th>Last Updated</th>
<th>Via</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td><em>[Static/5]</em> 5w0d 19:43:09</td>
<td>&gt; to 12.0.1.1 via em0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0.0.0/24</td>
<td><em>[BGP/170]</em> 1d 10:24:47, localpref 100, from 12.122.83.238</td>
<td>AS path: 7018 3356 13335 I, validation-state: valid</td>
<td>&gt; to 12.0.1.1 via em0.0</td>
<td></td>
</tr>
<tr>
<td>1.0.4.0/22</td>
<td><em>[BGP/170]</em> 1d 10:24:47, localpref 100, from 12.122.83.238</td>
<td>AS path: 7018 3356 4826 38803 I, validation-state: valid</td>
<td>&gt; to 12.0.1.1 via em0.0</td>
<td></td>
</tr>
<tr>
<td>1.0.4.0/24</td>
<td><em>[BGP/170]</em> 1d 10:24:47, localpref 100, from 12.122.83.238</td>
<td>AS path: 7018 3356 4826 38803 I, validation-state: valid</td>
<td>&gt; to 12.0.1.1 via em0.0</td>
<td></td>
</tr>
<tr>
<td>1.0.5.0/24</td>
<td><em>[BGP/170]</em> 1d 10:24:47, localpref 100, from 12.122.83.238</td>
<td>AS path: 7018 3356 4826 38803 I, validation-state: valid</td>
<td>&gt; to 12.0.1.1 via em0.0</td>
<td></td>
</tr>
<tr>
<td>1.0.6.0/24</td>
<td><em>[BGP/170]</em> 1d 10:24:47, localpref 100, from 12.122.83.238</td>
<td>AS path: 7018 3356 4826 38803 I, validation-state: valid</td>
<td>&gt; to 12.0.1.1 via em0.0</td>
<td></td>
</tr>
</tbody>
</table>
How to avoid loops?

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>31 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>IHL</td>
<td>TOS</td>
<td>Total length</td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Fragment offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 20 bytes
- 0-40 bytes
- Up to 65536 bytes
How to avoid loops?

**TTL (Time to Live):** Decrement by 1 at each hop, send back error at 0
How to avoid loops?

**TTL (Time to Live):** Decrement by 1 at each hop, send back error at 0

---

<table>
<thead>
<tr>
<th>Version</th>
<th>IHL</th>
<th>TOS</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Fragment offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td>Source address</td>
<td>Options</td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

**traceroute:** tool to send packets with increasing TTLs => can learn about network paths!
Traceroute example

[deemer@Warsprite ~]$ traceroute -q 1 google.com
traceroute to google.com (142.251.40.174), 30 hops max, 60 byte packets
  1  router1-nac.linode.com (207.99.1.13)  0.621 ms
  2  if-0-1-0-0-0.gw1.cjj1.us.linode.com (173.255.239.26)  0.499 ms
  3  72.14.222.136 (72.14.222.136)  0.949 ms
  4  72.14.222.136 (72.14.222.136)  0.919 ms
  5  108.170.248.65 (108.170.248.65)  1.842 ms
  6  lga25s81-in-f14.1e100.net (142.251.40.174)  1.812 ms
Traceroute example

[deemer@Warsprite ~]$ traceroute -q 1 amazon.co.uk
traceroute to amazon.co.uk (178.236.7.220), 30 hops max, 60 byte packets
 1  router2-nac.linode.com (207.99.1.14) 0.577 ms
 2  if-11-1-0-1-0.gw2.cjj1.us.linode.com (173.255.239.16) 0.461 ms
 3  ix-et-2-0-2-0.tcore3.njy-newark.as6453.net (66.198.70.104) 1.025 ms
 4  be3294.ccr41.jfk02.atlas.cogentco.com (154.54.47.217) 2.938 ms
 5  be2317.ccr41.lon13.atlas.cogentco.com (154.54.30.186) 69.725 ms
 6  be2350.rcr21.b023101-0.lon13.atlas.cogentco.com (130.117.51.138) 69.947 ms
 7  a100-row.demarc.cogentco.com (149.11.173.122) 71.639 ms
 8  150.222.15.28 (150.222.15.28) 78.217 ms
 9  150.222.15.21 (150.222.15.21) 84.383 ms
10  *
11  150.222.15.4 (150.222.15.4) 74.529 ms
    ...