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

Start of network layer

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

- **Physical Layer**: Service: move bits to other node across link.
- **Link Layer**: Service: move frames to other node across link. May add reliability, medium access control.
- **Network Layer**: Service: move packets to any other node in the network. Internet Protocol (IP).
- **Transport Layer**: Service: multiplexing applications. Reliable byte stream to other node (TCP), Unreliable datagram (UDP).
- **Application Layer**: Service: user-facing application. Application-defined messages.
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 (eg. 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

• 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

• 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

• 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
- 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)
• 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

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

- 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

10000000 10010100 xxxxxxxxxx xxxxxxxxxx

Network part
Identifies Brown (to the Internet)

Host part
Denotes individual hosts within the Brown Network
A network can designate IP addresses for its own hosts within its address range.

For 128.148.xxx.xxx:

10000000 10010100 xxxxxxxxxxx xxxxxxxxxx

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

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Binary</th>
<th>First 8 Bits</th>
<th>Next 8 Bits</th>
<th>Remaining 24 Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.0.0/16</td>
<td>00000001</td>
<td>00000010</td>
<td>xxxxxxxx</td>
<td>xxxxxxxx</td>
</tr>
<tr>
<td>8.0.0.0/8</td>
<td>00000100</td>
<td>xxxxxxxx</td>
<td>xxxxxxxx</td>
<td>xxxxxxxx</td>
</tr>
<tr>
<td>123.10.1.0/24</td>
<td>01111011</td>
<td>00001010</td>
<td>00000001</td>
<td>xxxxxxxx</td>
</tr>
<tr>
<td>201.112.10.200/30</td>
<td>11001001</td>
<td>01110000</td>
<td>00001010</td>
<td>110010xx</td>
</tr>
</tbody>
</table>
Example

How many addresses are in the network 192.1.0.0/20?
How do we move packets between networks?
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

Src: 1.2.1.3
Dst: 1.2.2.100

To more networks (ie, Internet)

IF0

IF1

IF2

1.2.1.1

1.2.2.1

1.2.2.100

1.2.2.105

1.2.1.3

1.2.1.200

1.2.1.200
Forwarding IP packets

Src: 1.2.1.3
Dst: 1.2.2.100

To more networks (ie, Internet)
Forwarding IP packets

Prefix: 1.2.1.3  
Interface: IF1

Src: 1.2.1.3  
Dst: 1.2.2.100

To more networks (ie, Internet)

Prefix  Interface
---  ---
1.2.1.2  IF0
1.2.1.3  IF1
1.2.1.200  IF0
1.2.1.200  IF2
1.2.2.100
1.2.2.105
1.2.2.105
Forwarding IP packets

Src: 1.2.1.3
Dst: 1.2.2.100

To more networks (ie, Internet)

<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>

To more networks (ie, Internet)
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 is a network prefix, not a whole address
• Select best prefix with *longest prefix matching* (more on this later)
# ip route
127.0.0.8 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

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

• 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:

• **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

Fill out the group preference survey for the IP project (announcement soon) by tomorrow (Feb 18) by 11:59PM
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

How to reach 1.2.2.100?

1.2.2.100
1.2.1.0/24
1.2.2.0/24
0.0.0.0/0

Prefix | Interface
--- | ---
1.2.1.0/24 | IF1
1.2.2.0/24 | IF2
0.0.0.0/0 | IF0

Output: IF2

Longest Prefix Matching (LPM): can represent entire IP space in (small) table!
rvreviews@route-server.ip.att.net>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>Route</th>
<th>Type</th>
<th>Active/Last</th>
<th>Update Time</th>
<th>Local Pref</th>
<th>AS Path</th>
<th>Validation State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>[Static/5]</td>
<td>5w0d 19:43:09</td>
<td>1d 10:24:47</td>
<td>100</td>
<td>7018 3356 13335 I</td>
<td>valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0.0.0/24</td>
<td>[BGP/170]</td>
<td>1d 10:24:47</td>
<td>1d 10:24:47</td>
<td>100</td>
<td>7018 3356 4826 38803 I</td>
<td>valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0.4.0/22</td>
<td>[BGP/170]</td>
<td>1d 10:24:47</td>
<td>1d 10:24:47</td>
<td>100</td>
<td>7018 3356 4826 38803 I</td>
<td>valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0.5.0/24</td>
<td>[BGP/170]</td>
<td>1d 10:24:47</td>
<td>1d 10:24:47</td>
<td>100</td>
<td>7018 3356 4826 38803 I</td>
<td>valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0.6.0/24</td>
<td>[BGP/170]</td>
<td>1d 10:24:47</td>
<td>1d 10:24:47</td>
<td>100</td>
<td>7018 3356 4826 38803 I</td>
<td>valid</td>
</tr>
</tbody>
</table>
How to avoid loops?

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

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

<table>
<thead>
<tr>
<th>Hop</th>
<th>Address</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>router1-nac.linode.com (207.99.1.13)</td>
<td>0.621 ms</td>
</tr>
<tr>
<td>2</td>
<td>if-0-1-0-0-0.gw1.cjj1.us.linode.com (173.255.239.26)</td>
<td>0.499 ms</td>
</tr>
<tr>
<td>3</td>
<td>72.14.222.136 (72.14.222.136)</td>
<td>0.949 ms</td>
</tr>
<tr>
<td>4</td>
<td>72.14.222.136 (72.14.222.136)</td>
<td>0.919 ms</td>
</tr>
<tr>
<td>5</td>
<td>108.170.248.65 (108.170.248.65)</td>
<td>1.842 ms</td>
</tr>
<tr>
<td>6</td>
<td>lga25s81-in-f14.1e100.net (142.251.40.174)</td>
<td>1.812 ms</td>
</tr>
</tbody>
</table>
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
          ... 