CSCI-1680
Network Layer:
IP & Forwarding

Rodrigo Fonseca

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

- Network layer: Internet Protocol (v4)
- Forwarding
- Next 2 classes: Routing
Internet Protocol Goal

• **How to connect everybody?**
  – New global network or connect existing networks?

• **Glue lower-level networks together:**
  – allow packets to be sent between any pair or hosts

• **Wasn’t this the goal of switching?**
Inter-networking Challenges

• **Heterogeneity**
  – Different addresses
  – Different service models
  – Different allowable packet sizes

• **Scaling**

• **Congestion control**
How would you design such a protocol?

• **Circuits or packets?**
  – Predictability

• **Service model**
  – Reliability, timing, bandwidth guarantees

• **Any-to-any**
  – Finding nodes: naming, routing
  – Maintenance (join, leave, add/remove links,…)
  – Forwarding: message formats
IP’s Decisions

• **Packet switched**
  – Unpredictability, statistical multiplexing

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

• **Any-to-any**
  – Common message format
  – Separated routing from forwarding
  – Naming: uniform addresses, hierarchical organization
  – Routing: hierarchical, prefix-based (longest prefix matching)
  – Maintenance: delegated, hierarchical
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
- Replace NCP
- Initial design: single protocol providing a unified reliable pipe
  - Could support any application
- Different requirements soon emerged, and the two were separated
  - IP: basic datagram service among hosts
  - TCP: reliable transport
  - UDP: unreliable *multiplexed* datagram service
An excellent read

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 Protocol running on all hosts \textit{and routers}
- Routers are \textit{present in all networks they join}
- Uniform addressing
- Forwarding/Fragmentation
- Complementary:
  - Routing, Error Reporting, Address Translation
How does this work?

- **Routers are present in all networks they join**
  - Nodes only know how to talk on the networks they are in
- **What about addresses?**
  - IP addresses mean nothing for the L2 networks!
  - Need a mapping
- **Forwarding**
  - Decide whether to deliver locally, or to another router in the same network
  - Addresses are allocated in a way to make this work
IP Protocol

• **Provides addressing and forwarding**
  – Addressing is a set of conventions for naming nodes in an IP network
  – Forwarding is a local action by a router: passing a packet from input to output port

• **IP forwarding finds output port based on destination address**
  – Also defines certain conventions on how to handle packets (e.g., fragmentation, time to live)

• **Contrast with routing**
  – Routing is the process of determining how to map packets to output ports (topic of next two lectures)
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…).
Format of IP addresses

- Globally unique (or made seem that way)
  - 32-bit integers, read in groups of 8-bits:
    128.148.32.110

- Hierarchical organization:
  - Assign blocks of contiguous addresses to the same parts of the network
  - All hosts in the same network share the same prefix
  - Networks can have different sizes (different prefix sizes)
  - Addresses have network and host parts
Prefix Notation

• **Significant bits + mask**
  • E.g. all nodes which share prefix 128.148.0.0:
    - 16 MSB matter
    - Mask is: 11111111 11111111 00000000 00000000
    - Or 255.255.0.0
    - Or /16

• **Could say**
  - 128.148
  - 128.148/16
  - 128.148.0.0/16
  - 128.148.0.0 netmask 255.255.0.0
Forwarding Tables

• Exploit hierarchical structure of addresses: need to know how to reach *networks*, not hosts

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>212.31.32.*</td>
<td>0.0.0.0</td>
</tr>
<tr>
<td>18.<em>.</em>.*</td>
<td>212.31.32.5</td>
</tr>
<tr>
<td>128.148.<em>.</em></td>
<td>212.31.32.4</td>
</tr>
<tr>
<td>Default</td>
<td>212.31.32.1</td>
</tr>
</tbody>
</table>

• Keyed by network portion, not entire address
• Next address should be local: router knows how to reach it directly* (we’ll see how soon)
IP Forwarding Table

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>212.31.32/24</td>
<td>0.0.0.0</td>
</tr>
<tr>
<td>18/8</td>
<td>212.31.32.5</td>
</tr>
<tr>
<td>128.148/16</td>
<td>212.31.32.4</td>
</tr>
<tr>
<td>128.148.128/17</td>
<td>212.31.32.8</td>
</tr>
<tr>
<td>0/0</td>
<td>212.31.32.1</td>
</tr>
</tbody>
</table>

• Forwarding is done by *longest prefix matching*
  – More on this later
Obtaining IP Addresses

• Blocks of IP addresses allocated hierarchically
  – ISP obtains an *address block*, may subdivide
  
  ISP: 128.35.16/20  \(10000000 \ 00100011 \ 00010000 \ 00000000\)
  Client 1: 128.35.16/22  \(10000000 \ 00100011 \ 00010000 \ 00000000\)
  Client 2: 128.35.20/22  \(10000000 \ 00100011 \ 00010100 \ 00000000\)
  Client 3: 128.35.24/21  \(10000000 \ 00100011 \ 00011000 \ 00000000\)

• Blocks restricted to powers of 2

• Global allocation: ICANN, /8’s *(ran out!)*

• Regional registries: ARIN, RIPE, APNIC, LACNIC, AFRINIC
How does this really work?

H1 -> H2: H2.ip & H1.mask != H1.subnet => no direct path
### R1’s Forwarding Table

<table>
<thead>
<tr>
<th>Network</th>
<th>Subnet Mask</th>
<th>Next Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.96.34.0</td>
<td>255.255.255.128</td>
<td>128.96.34.1</td>
</tr>
<tr>
<td>128.96.34.128</td>
<td>255.255.255.128</td>
<td>128.96.34.130</td>
</tr>
<tr>
<td>128.96.33.0</td>
<td>255.255.255.0</td>
<td>128.96.34.129</td>
</tr>
</tbody>
</table>

![Diagram](image.png)
## IP v4 packet format

<table>
<thead>
<tr>
<th>Position</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td>vers</td>
<td>version number (4 bits)</td>
</tr>
<tr>
<td></td>
<td>hdr len</td>
<td>header length (16 bits)</td>
</tr>
<tr>
<td></td>
<td>TOS</td>
<td>type of service (8 bits)</td>
</tr>
<tr>
<td></td>
<td>Total Length</td>
<td>total length (16 bits)</td>
</tr>
<tr>
<td></td>
<td>Identification</td>
<td>identification (16 bits)</td>
</tr>
<tr>
<td></td>
<td>Fragment offset</td>
<td>fragment offset (24 bits)</td>
</tr>
<tr>
<td></td>
<td>TTL</td>
<td>time to live (8 bits)</td>
</tr>
<tr>
<td></td>
<td>Protocol</td>
<td>protocol number (8 bits)</td>
</tr>
<tr>
<td></td>
<td>hdr checksum</td>
<td>header checksum (16 bits)</td>
</tr>
<tr>
<td></td>
<td>Source IP address</td>
<td>source IP address (32 bits)</td>
</tr>
<tr>
<td></td>
<td>Destination IP address</td>
<td>destination IP address (32 bits)</td>
</tr>
<tr>
<td></td>
<td>Options</td>
<td>options</td>
</tr>
<tr>
<td></td>
<td>Padding</td>
<td>padding</td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td>data (variable length)</td>
</tr>
</tbody>
</table>
IP header details

• Forwarding based on destination address
• TTL (time-to-live) decremented at each hop
  – Originally was in seconds (no longer)
  – Mostly prevents forwarding loops
  – Other cool uses…
• Fragmentation possible for large packets
  – Fragmented in network if crossing link w/ small frame
  – MF: more fragments for this IP packet
  – DF: don’t fragment (returns error to sender)
• Following IP header is “payload” data
  – Typically beginning with TCP or UDP header
Other fields

• Version: 4 (IPv4) for most packets, there’s also 6
• Header length: in 32-bit units (>5 implies options)
  – 4 bits * 4 bytes = 64KiB max
• Type of service (won’t go into this)
• Protocol identifier (TCP: 6, UDP: 17, ICMP: 1, …)
• Checksum over the header
Translating IP to lower level addresses or... How to reach these *local* addresses?

- **Map IP addresses into physical addresses**
  - E.g., Ethernet address of destination host
  - or Ethernet address of next hop router

- **Techniques**
  - Encode physical address in host part of IP address (IPv6)
  - Each network node maintains lookup table (IP->phys)
ARP – *address resolution protocol*

- Dynamically builds table of IP to physical address bindings for a *local network*
- Broadcast request if IP address not in table
- All learn IP address of requesting node (broadcast)
- Target machine responds with its physical address
- Table entries are discarded if not refreshed
**ARP Ethernet frame format**

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware type = 1</td>
<td>ProtocolType = 0x0800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLen = 48</td>
<td>PLen = 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceHardwareAddr (bytes 0–3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceHardwareAddr (bytes 4–5)</td>
<td>SourceProtocolAddr (bytes 0–1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceProtocolAddr (bytes 2–3)</td>
<td>TargetHardwareAddr (bytes 0–1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TargetHardwareAddr (bytes 2–5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TargetProtocolAddr (bytes 0–3)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Why include source hardware address?**
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
Network Address Translation (NAT)

• Despite CIDR, it’s still difficult to allocate addresses ($2^{32}$ is only 4 billion)
• We’ll talk about IPv6 later
• 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?
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>

20-byte IP header
(protocol = 1—ICMP)

depends on type/code

Types include:
- echo, echo reply, destination unreachable, time exceeded, ..

See http://www.iana.org/assignments/icmp-parameters
**Example: Time Exceeded**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>20-byte IP header (protocol = 1—ICMP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 11</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>unused</td>
</tr>
</tbody>
</table>

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

- **Code usually 0 (TTL exceeded in transit)**
- **Discussion: traceroute**
Example: Can’t Fragment

- Sent if DF=1 and packet length > MTU
- What can you use this for?
- Path MTU Discovery
  - Can do binary search on packet sizes
  - But better: base algorithm on most common MTUs
Coming Up

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

# 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>

# 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