CSCI-1680 - Computer Networks

Network Layer:
IP & Forwarding

Chen Avin
• IP out today. Your job:
  – Find partners, get setup with Github
  – Implement IP forwarding and DV routing
  – Get started NOW (ok, after class)

• HW1 due today
Today

• **Network layer: Internet Protocol (v4)**
• **Forwarding**
  – Addressing
  – Fragmentation
  – ARP
  – DHCP
  – NATs
• **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?**
Internetworking 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

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

- **Any-to-any**
  - Common message format
  - Separated routing from forwarding (Data & Control Plane)
  - Naming: uniform addresses, hierarchical organization
  - Routing: hierarchical, prefix-based (longest prefix matching)
  - Maintenance: delegated, hierarchical
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 and *routers*
- Routers are present in all networks they join
- Uniform addressing
- Forwarding/Fragmentation
- Complementary:
  - Routing, Error Reporting, Address Translation
The Internet network layer

host, router network layer functions:

- **IP protocol**
  - addressing conventions
  - datagram format
  - packet handling conventions

- **ICMP protocol**
  - error reporting
  - router "signaling"

- **routing protocols**
  - path selection
  - RIP, OSPF, BGP

- **forwarding table**

transport layer: TCP, UDP

link layer

physical layer

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…)
IP addressing

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

- Hierarchical: network + host

- Originally, routing prefix embedded in address

  - Class A (8-bit prefix), B (16-bit), C (24-bit)
  - Routers need only know route for each network
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)
Classed Addresses

• Hierarchical: network + host
  – Saves memory in backbone routers (no default routes)
  – Originally, routing prefix embedded in address
  – Routers in same network must share network part

• Inefficient use of address space
  – Class C with 2 hosts (2/255 = 0.78% efficient)
  – Class B with 256 hosts (256/65535 = 0.39% efficient)
  – Shortage of IP addresses
  – Makes address authorities reluctant to give out class B’s

• Still too many networks
  – Routing tables do not scale

• Routing protocols do not scale
IP addressing: CIDR

**CIDR:** Classless InterDomain Routing
- subnet portion of address of arbitrary length
- address format: `a.b.c.d/x`, where `x` is # bits in subnet portion of address

```
11001000  00010111  00010000  00000000

200.23.16.0/23
```
Classless Addressing

Route aggregation with CIDR
Longest prefix matching

**longest prefix matching**

when looking for forwarding table entry for given destination address, use **longest** address prefix that matches destination address.

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010*** *******</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 *******</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011*** *******</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

**examples:**

DA: 11001000 00010111 00010110 **10100001**

which interface?

DA: 11001000 00010111 00011000 **10101010**

which interface?
Packet Format

- Version (4): currently 4
- Hlen (4): number of 32-bit words in header
- TOS (8): type of service (not widely used)
- Length (16): number of bytes in this datagram
- Ident (16): used by fragmentation
- Flags/Offset (16): used by fragmentation
- TTL (8): number of hops this datagram has traveled
- Protocol (8): demux key (TCP=6, UDP=17)
- Checksum (16): of the header only
- DestAddr & SrcAddr (32)
Fragmentation & Reassembly

• Each network has maximum transmission unit (MTU)

• Strategy
  – Fragment when necessary (MTU < size of datagram)
  – Source tries to avoid fragmentation (why?)
  – Re-fragmentation is possible
  – Fragments are self-contained datagrams
  – Delay reassembly until destination host
  – No recovery of lost fragments
• Ethernet MTU is 1,500 bytes
• PPP MTU is 576 bytes
  – R2 must fragment IP packets to forward them
Fragmentation Example (cont)

- IP addresses plus ident field identify fragments of same packet
- MF (more fragments bit) is 1 in all but last fragment
- Fragment offset multiple of 8 bytes
  - Multiply offset by 8 for fragment position original packet

<table>
<thead>
<tr>
<th>Start of header</th>
<th>Identi = x</th>
<th>Offset = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1400 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start of header</th>
<th>Identi = x</th>
<th>Offset = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>512 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start of header</th>
<th>Identi = x</th>
<th>Offset = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>64 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start of header</th>
<th>Identi = x</th>
<th>Offset = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>376 bytes</td>
</tr>
</tbody>
</table>
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 Packet Format

<table>
<thead>
<tr>
<th>Position</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HardwareType</td>
<td>type of physical network (e.g., Ethernet)</td>
</tr>
<tr>
<td>8</td>
<td>ProtocolType</td>
<td>type of higher layer protocol (e.g., IP)</td>
</tr>
<tr>
<td>16</td>
<td>HLEN &amp; PLEN</td>
<td>length of physical and protocol addresses</td>
</tr>
<tr>
<td>31</td>
<td>Operation</td>
<td>request or response</td>
</tr>
<tr>
<td>0–3</td>
<td>SourceHardwareAddr</td>
<td>(bytes 0–3)</td>
</tr>
<tr>
<td>4–5</td>
<td>SourceHardwareAddr</td>
<td>(bytes 4–5)</td>
</tr>
<tr>
<td>2–3</td>
<td>SourceProtocolAddr</td>
<td>(bytes 2–3)</td>
</tr>
<tr>
<td>2–5</td>
<td>TargetHardwareAddr</td>
<td>(bytes 2–5)</td>
</tr>
<tr>
<td>0–3</td>
<td>TargetProtocolAddr</td>
<td>(bytes 0–3)</td>
</tr>
</tbody>
</table>

- HardwareType: type of physical network (e.g., Ethernet)
- ProtocolType: type of higher layer protocol (e.g., IP)
- HLEN & PLEN: length of physical and protocol addresses
- Operation: request or response
- Source/Target Physical/Protocol addresses
DHCP: Dynamic Host Configuration Protocol

**goal:** allow host to *dynamically* obtain its IP address from network server when it joins network
- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

**DHCP overview:**
- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg
DHCP client-server scenario

DHCP server

arriving DHCP client needs address in this network

223.1.1.0/24

223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4

223.1.2.0/24

223.1.2.1
223.1.2.9
223.1.2.2

223.1.3.0/24

223.1.3.1
223.1.3.2

223.1.3.27

DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover
- src: 0.0.0.0, 68
- dest: 255.255.255.255, 67
- yiaddr: 0.0.0.0
- transaction ID: 654

arriving client

DHCP offer
- src: 223.1.2.5, 67
- dest: 255.255.255.255, 68
- yiaddr: 223.1.2.4
- transaction ID: 654
- lifetime: 3600 secs

DHCP request
- src: 0.0.0.0, 68
- dest: 255.255.255.255, 67
- yiaddr: 223.1.2.4
- transaction ID: 655
- lifetime: 3600 secs

DHCP ACK
- src: 223.1.2.5, 67
- dest: 255.255.255.255, 68
- yiaddr: 223.1.2.4
- transaction ID: 655
- lifetime: 3600 secs
DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)
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?
NAT: network address translation

rest of Internet

local network (e.g., home network) 10.0.0/24

138.76.29.7

10.0.0.1

10.0.0.2

10.0.0.3

all datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

rest of Internet

10.0.0/24

IPv4 private addresses

<table>
<thead>
<tr>
<th>IANA-reserved private IPv4 network ranges</th>
<th>Start</th>
<th>End</th>
<th>No. of addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-bit Block (/8 prefix, 1 × A)</td>
<td>10.0.0.0</td>
<td>10.255.255.255</td>
<td>16 777 216</td>
</tr>
<tr>
<td>20-bit Block (/12 prefix, 16 × B)</td>
<td>172.16.0.0</td>
<td>172.31.255.255</td>
<td>1 048 576</td>
</tr>
<tr>
<td>16-bit Block (/16 prefix, 256 × C)</td>
<td>192.168.0.0</td>
<td>192.168.255.255</td>
<td>65 536</td>
</tr>
</tbody>
</table>
ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer “above” IP:
  - ICMP msgs carried in IP datagrams
- **ICMP message**: type, code plus first 8 bytes of IP datagram causing error

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>
Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL = 1
  - second set has TTL = 2, etc.
  - unlikely port number

- when $n$th set of datagrams arrives to $n$th router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address

- when ICMP messages arrives, source records RTTs

**stopping criteria:**

- UDP segment eventually arrives at destination host
- destination returns ICMP “port unreachable” message (type 3, code 3)
- source stops
Coming Up

• Routing: how do we fill the routing tables?
  – Intra-domain routing
  – Inter-domain routing