CS118 Discussion 1A, Week 6

Zengwen Yuan
Dodd Hall 78, Friday 10:00—10:50 a.m.
Outline

• Network Layer
  • Overview: data v.s. control plane
  • IPv4/IPv6, DHCP, NAT
• Project 2 spec
• Midterm review
Network layer: overview

- Basic functions for network layer
  - Forwarding/Routing
- Network service model
  - Guaranteed delivery
  - Guaranteed delivery w/ bounded delay
  - In-order packet delivery
  - Guaranteed minimal bandwidth
Network layer: overview

- Connection v.s. connection-less delivery
  - circuit switch/packet switch
- Network layer protocols
  - Addressing and fragmentation: IPv4, IPv6
  - Routing: RIP, OSPF, BGP, DVMRP, PIM
  - Others: DHCP, ICMP, NAT
IPv4 Header

- **Header length**: 4-byte unit
- **Length**: 1-byte unit
- **Fragmentation**: id + MF/DF + offset (8-byte unit)
- **TTL**: time to live
- **Checksum**
  - Is it redundant?
  - Why is it just checksum for header?
- **Protocol**: identifies the upper layer protocol
- **Source and destination IP addresses**
IP address

- Globally recognizable identifier
- IPv4: 0.0.0.0~255.255.255.255
  - Most IP addresses are globally unique
  - Exception — why?
- Network id, host id
- CIDR address
IP address classes


<table>
<thead>
<tr>
<th>Class</th>
<th>1st Octet Decimal</th>
<th>1st Octet High Order Bits</th>
<th>Network/Host ID (N=Network, H=Host)</th>
<th>Default Subnet Mask</th>
<th>Number of Networks</th>
<th>Hosts per Network (Usable Addresses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 – 126*</td>
<td>0</td>
<td>N.H.H.H</td>
<td>255.0.0.0</td>
<td>126 (2^7 – 2)</td>
<td>16,777,214 (2^24 – 2)</td>
</tr>
<tr>
<td>B</td>
<td>128 – 191</td>
<td>10</td>
<td>N.N.H.H</td>
<td>255.255.0.0</td>
<td>16,382 (2^14 – 2)</td>
<td>65,534 (2^16 – 2)</td>
</tr>
<tr>
<td>C</td>
<td>192 – 223</td>
<td>110</td>
<td>N.N.N.H</td>
<td>255.255.255.0</td>
<td>2,097,150 (2^21 – 2)</td>
<td>254 (2^8 – 2)</td>
</tr>
<tr>
<td>D</td>
<td>224 – 239</td>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td>Reserved for Multicasting</td>
</tr>
<tr>
<td>E</td>
<td>240 – 254</td>
<td>1111</td>
<td></td>
<td></td>
<td></td>
<td>Experimental; used for research</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Private Networks</th>
<th>Subnet Mask</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10.0.0.0</td>
<td>255.0.0.0</td>
<td>10.0.0.0 - 10.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>172.16.0.0 - 172.31.0.0</td>
<td>255.240.0.0</td>
<td>172.16.0.0 - 172.31.255.255</td>
</tr>
<tr>
<td>C</td>
<td>192.168.0.0</td>
<td>255.255.0.0</td>
<td>192.168.0.0 - 192.168.255.255</td>
</tr>
</tbody>
</table>
Hierarchical addressing

• subnet: a portion of addressing space
  • extend bits from the network id
  • <network address>/<subnet mask>
• route aggregation
CIDR address

- a.b.c.d/x
  - x: # bits in network ID portion of the address
  - address: a.b.c.d, network mask: $2^32 - 2^{(32-x)}$

CIDR   11001000 00010111 00010000 00000000
IP prefix 200.23.16.0/23
netmask 11111111 11111111 11111111 00000000
255.255.254.0
IP fragmentation and reassembly

- MTU: maximum transmission unit
- identifier
- flag bit: three bit
  - DF (Do not Fragment) = 0
  - MF (More Fragments) = 0?
- offset
Switching

• Longest prefix matching

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00011000 **********</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00010*** **********</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 0001**** **********</td>
<td>2</td>
</tr>
<tr>
<td>******** ******** ******** ********</td>
<td>3</td>
</tr>
</tbody>
</table>

• Linear lookup
DHCP: Dynamic Host Configuration Protocol

- Dynamically allocates the following info to a host
  - IP address for the host
  - IP address for default router
  - Subnet mask
  - IP address for DNS caching resolver
- Allows address reuse
DHCP: operations

- Host broadcasts “DHCP discovery” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- Host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

Important example on Chapter 4 slides 45—46!
NAT (network address translation)

- Depletion of IPv4 addresses — short-term solution
  - IP tunneling?
- Use private IP addresses
- Side-benefit: security
- How to achieve?
  - <public IP:port> — <private IP:port> mapping
NAT: detail

- outgoing packets:
  - replace (source IP address, source port #) of every outgoing packet to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination address
  - remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair

- incoming packets:
  - replace (destination NAT IP address, destination port #) of every incoming packet with corresponding (source IP address, port #) stored in NAT table
NAT: downside

- Increased complexity
- Single point of failure
- Cannot run services inside a NAT box
IPv6

IPv6 Header Format (RFC 2460)
IPv6/IPv4 differences

- Fixed-length 40 byte header
  - Length field excludes header
  - Header Length field eliminated
- Address length: 128 bits
- Priority: usage yet to be finalized
- Flow Label: identify packets in same flow
- Next header: identify upper layer protocol for data
- Options: outside of the basic header, indicated by Next Header field
- Header Checksum: removed
IPv6 address format (optional)


- Can skip leading zeros of each word:
  2607:F010:3f9:0:0:0:4:1

- Can skip one sequence of zero words (compressed representation), e.g., 2607:f010:3f9::4:1

- Can leave the last 32 bits in dot-decimal:
  2607:f010:3f9::0.4.0.1

- Can specify a prefix by /length: 2607:f010:3f9::/64
Special IPv6 addresses (optional)

- ::/128 - Unspecified
- ::1/128 - Loopback
- ::ffff:0:0/96 - IP4-mapped address
- 2002::/16 - 6to4
- ff00::/8 - Multicast
- fe80::/10 - Link-Local Unicast
Routing: concepts

• Global or decentralized information?

  • global: all routers have complete topology, link cost info

  • algorithm?
Routing: concepts

• Global or decentralized information?
  • global: all routers have complete topology, link cost info
  • “link state” algorithms
Routing: concepts

• Global or decentralized information?
  • global: all routers have complete topology, link cost info
  • “link state” algorithms
  • decentralized: router knows physically-connected neighbors, link costs to neighbors; iterative process of computation, exchange of info with neighbors
  • algorithm?
Routing: concepts

• Global or decentralized information?
  
  • global: all routers have complete topology, link cost info
  
  • “link state” algorithms
  
  • decentralized: router knows physically-connected neighbors, link costs to neighbors; iterative process of computation, exchange of info with neighbors
  
  • “distance vector” algorithms
Link state routing

- Dijkstra’s algorithm
  - net topology, link costs known to all nodes
  - computes least cost paths from one node (‘source”) to all other nodes
  - iterative: after k iterations, know least cost path to k destinations
Link state routing: algorithm

1 Initialization:
2 \[ N' = \{u\} \]
3 for all nodes \( v \)
4     if \( v \) adjacent to \( u \)
5         then \( D(v) = c(u,v) \)
6     else \( D(v) = \infty \)
7
8 Loop
9     find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
10    add \( w \) to \( N' \)
11    update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \):
12        [Link cost update heuristic from Dijkstra algo.]
13 until all nodes in \( N' \)

\( c(x, y) \): link cost from node \( x \) to \( y \); \( c(x, y) = \infty \) if not direct neighbors
\( D(v) \): current value of cost of path from source to destination \( v \)
\( p(v) \): predecessor node along path from source to \( v \)
\( N' \): set of nodes whose least cost path definitively known
Link state routing: algorithm

1  Initialization:
2     \( N' = \{ u \} \)
3     for all nodes \( v \)
4        if \( v \) adjacent to \( u \)
5           then \( D(v) = c(u,v) \)
6        else \( D(v) = \infty \)
7
8  Loop
9     find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
10    add \( w \) to \( N' \)
11    update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \):
12       \( D(v) = \min( D(v), D(w) + c(w,v) ) \)
13    until all nodes in \( N' \)

\( c(x, y) \): link cost from node \( x \) to \( y \); \( c(x, y) = \infty \) if not direct neighbors
\( D(v) \): current value of cost of path from source to destination \( v \)
\( p(v) \): predecessor node along path from source to \( v \)
\( N' \): set of nodes whose least cost path definitively known
Link state routing: example

• Using link state routing to setup a forwarding table for node u
Let's work it out

<table>
<thead>
<tr>
<th>N'</th>
<th>D(v), p(v)</th>
<th>D(w), p(w)</th>
<th>D(x), p(x)</th>
<th>D(y), p(y)</th>
<th>D(z), p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>2, u</td>
<td>5, u</td>
<td>1, u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>ux</td>
<td>2, u</td>
<td>4, x</td>
<td></td>
<td>2, x</td>
<td>∞</td>
</tr>
<tr>
<td>uxy</td>
<td>2, u</td>
<td>3, y</td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyv</td>
<td></td>
<td>3, y</td>
<td></td>
<td>4, y</td>
<td></td>
</tr>
<tr>
<td>uxyvw</td>
<td></td>
<td></td>
<td></td>
<td>4, y</td>
<td></td>
</tr>
<tr>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
</tbody>
</table>
Let’s work it out

<table>
<thead>
<tr>
<th>N’</th>
<th>D(v), p(v)</th>
<th>D(w), p(w)</th>
<th>D(x), p(x)</th>
<th>D(y), p(y)</th>
<th>D(z), p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>2, u</td>
<td>5, u</td>
<td>1, u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>ux</td>
<td>2, u</td>
<td>4, x</td>
<td></td>
<td>2, x</td>
<td>∞</td>
</tr>
<tr>
<td>uxy</td>
<td></td>
<td>3, y</td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyv</td>
<td></td>
<td>3, y</td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyvw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyvwx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Project 2 Spec

• Questions?

• Two major parts
  • packet format design
  • message handling