Last Lecture: Network Layer

1. Design goals and issues ✔
2. Basic Routing Algorithms & Protocols ✔
   ▪ Packet Forwarding
   ▪ Shortest-Path Algorithms
   ▪ Routing Protocols
3. Addressing, fragmentation and reassembly
4. Internet Routing Protocols and Inter-networking
5. Router design
6. Congestion Control, Quality of Service
7. More on the Internet’s Network Layer
This Lecture: Network Layer

1. Design goals and issues
2. Basic Routing Algorithms & Protocols
3. Addressing, Fragmentation and reassembly ✔
   - Hierarchical addressing
   - Address allocation & CIDR
   - IP fragmentation and reassembly
4. Internet Routing Protocols and Inter-networking
5. Router design
6. Congestion Control, Quality of Service
7. More on the Internet’s Network Layer
1. IP Addressing

- Dotted-quad notation: here’s timberlake.cse’s IP

```
128  205  36  8

10000000 11001101 00100100 00001000
```

- Theoretically, up to $2^{32} \approx 4$ billion hosts
- Practically, about 768 millions (Jul 2010, ISC Survey), still huge!

- Routing table with 768M entries? No no.
Hierarchical Addressing: Rough Idea

- Each “network” assigned a prefix
- Foreign routers’ routing tables only need an entry for the entire “network”
  - The entry points to the network’s “gateway(s)”
# Subnet Mask: Extracting the Network Prefix

**Address**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>205</td>
<td>36</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>11001101</td>
<td>00100100</td>
<td>00001000</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>11111111</td>
<td>00000000</td>
<td></td>
</tr>
</tbody>
</table>

**Mask**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Scalability Improved

- **Routing tables are smaller** (but still too big)
- **No need to update the routers when new host added**
  - E.g., adding a new host 5.6.7.213 on the right
  - Doesn’t require adding a new forwarding-table entry

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.2.3.4</td>
<td>1.2.3.7</td>
<td>1.2.3.156</td>
<td>5.6.7.8</td>
</tr>
<tr>
<td>LAN 1</td>
<td>host</td>
<td>host</td>
<td>...</td>
<td>host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5.6.7.9</td>
<td>5.6.7.212</td>
</tr>
<tr>
<td>LAN 2</td>
<td>host</td>
<td>host</td>
<td>...</td>
<td>host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.6.7.213</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.2.3.0/24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.6.7.0/24</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

forwarding table
Address Allocation

- How to partition the address space into “blocks”
- Who gets which block?
# Classful Allocation (The Old Way)

<table>
<thead>
<tr>
<th>Class</th>
<th>Network ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td>Host ID</td>
</tr>
<tr>
<td>Class B</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Class C</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>Class D</td>
<td>1110</td>
<td>Multicast Addresses</td>
</tr>
<tr>
<td>Class E</td>
<td>1111</td>
<td>Reserved for experiments</td>
</tr>
</tbody>
</table>

This is why dotted-quad notation is used.
Classless Inter-Domain Routing (CIDR)

Use two 32-bit numbers to represent a network. Network number = IP address + Mask

IP Address: 12.4.0.0   IP Mask: 255.254.0.0

Address: 00001100 00000100 00000000 00000000

Mask: 11111111 11111110 00000000 00000000

Network Prefix for hosts

Written as 12.4.0.0/15
CIDR: Reduce Routing Table Sizes

- About 350K entries to date

- Send me anything with addresses beginning 200.23.16.0/20
- Send me anything with addresses beginning 199.31.0.0/16

Internet

Fly-By-Night-ISP

ISP-Rs

Organization 0
200.23.16.0/23

Organization 1
200.23.18.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23
(BGP) Routing Table Size Growth
Scalability: Address Aggregation

Provider is given 201.10.0.0/21

201.10.0.0/22  201.10.4.0/24  201.10.5.0/24  201.10.6.0/23

Routers in the rest of the Internet just need to know how to reach 201.10.0.0/21. The provider can direct the IP packets to the appropriate customer.
Multi-homed customer with 201.10.6.0/23 has two providers. Other parts of the Internet need to know how to reach these destinations through both providers.
ISP-R-Us has a more specific route to Organization 1

Requires routers to do *longest prefix match*, per packet, every few nanosecond.
2. IP Fragmentation and Reassembly

- A packet may hit networks with different MTUs

- Fragmentation needed at networks whose MTUs are smaller than the packet

- Reassemble the packet after getting out
Where to do Reassembly

- At end nodes or routers?

- **At routers:**
  - Con: How much buffer space required at routers?
  - Con: What if routes in network change? Or there are multiple paths to the same destination?

- **At end (receiving) nodes**
  - Pro: avoids unnecessary work where large packets are fragmented multiple times
  - Pro: at routers, less buffer space & less computation
  - Con: if any fragment missing, retransmit entire packet through entire path, wasting bandwidth
  - *TCP/IP takes this approach*
IP Packet Format

- **IP protocol version number**
- **Header length (bytes)**
- **"type" of data**
- **Max number remaining hops (decremented at each router)**
- **Upper layer protocol to deliver payload to**
- **Total datagram length (bytes)**
- **For fragmentation/reassembly**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ver</td>
<td>IP protocol version number</td>
</tr>
<tr>
<td>head.</td>
<td>Header length (bytes)</td>
</tr>
<tr>
<td>type of service</td>
<td>&quot;type&quot; of data</td>
</tr>
<tr>
<td>length</td>
<td></td>
</tr>
<tr>
<td>16-bit identifier</td>
<td></td>
</tr>
<tr>
<td>flgs</td>
<td></td>
</tr>
<tr>
<td>fragment offset</td>
<td></td>
</tr>
<tr>
<td>time to live</td>
<td></td>
</tr>
<tr>
<td>upper layer</td>
<td></td>
</tr>
<tr>
<td>header checksum</td>
<td></td>
</tr>
<tr>
<td>32 bit source IP address</td>
<td></td>
</tr>
<tr>
<td>32 bit destination IP address</td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>(variable length, typically a TCP or UDP segment)</td>
</tr>
<tr>
<td>E.g. timestamp, record route taken, specify list of routers to visit.</td>
<td></td>
</tr>
</tbody>
</table>
Fragmentation Related Fields

- **Length**
  - Length of IP fragment

- **Identifier**
  - To match up with other fragments

- **Flags**
  - Don’t fragment flag
  - More fragments flag

- **Fragment offset**
  - Where this fragment lies in entire IP datagram
  - Measured in 8 octet units (13 bit field)
IP Fragmentation Example #1

MTU = 4000

Length = 3820, M=0

IP Header    IP Data
IP Fragmentation Example #2

- IP Header
- Data

Length = 3820, M=0

3800 bytes

MTU = 2000

Length = 2000, M=1, Offset = 0

1980 bytes

Length = 1840, M=0, Offset = 1980

1820 bytes

SUNY at Buffalo; CSE 489/589 – Modern Networking Concepts; Fall 2010; Instructor: Hung Q. Ngo
IP Fragmentation Example #3

MTU = 1500

Length = 2000, M=1, Offset = 0

Length = 1840, M=0, Offset = 1980

Length = 1480 bytes

Length = 1500, M=1, Offset = 0

Length = 1500, M=1, Offset = 1980

Length = 520, M=1, Offset = 1480

Length = 1480 bytes

Length = 360, M=0, Offset = 3460

Length = 500 bytes

SUNY at Buffalo; CSE 489/589 – Modern Networking Concepts; Fall 2010; Instructor: Hung Q. Ngo
IP Reassembly

- Fragments might arrive out-of-order
  - Don’t know how much memory required until receive final fragment
- Some fragments may be duplicated
  - Keep only one copy
- Some fragments may never arrive
  - After a while, give up entire process
Fragmentation and Reassembly Concepts

- **Decentralized**: Every network can choose MTU
- **Connectionless**
  - Each (fragment of a) packet contains full routing information
  - Fragments travel independently
- **Best effort**
  - Fail by dropping packet
  - Destination can give up on reassembly
  - No need to signal sender that failure occurred
- **E2E principle**
  - Reassembly at endpoints
- **These are key networking principles!**
Fragmentation is Harmful

- Uses resources poorly
  - Forwarding costs per packet
  - Best if we can send large chunks of data
  - Worst case: packet just bigger than MTU
- Poor end-to-end performance
- Solution: *Path MTU discovery* protocol

- Common theme in system design
  - Assure correctness by implementing complete protocol
  - Optimize common cases to avoid full complexity