Internet Protocol: Addressing and Lookup

Srinidhi Varadarajan
IP Addressing: introduction

- **IP address**: 32-bit identifier for host, router *interface*
- **interface**: connection between host, router and physical link
  - router’s typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with interface, not host, router

223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1 1
IP Addressing

- **IP address:**
  - network part (high order bits)
  - host part (low order bits)

- **What’s a network?**
  (from IP address perspective)
  - device interfaces with same network part of IP address
  - can physically reach each other without intervening router

network consisting of 3 IP networks
(for IP addresses starting with 223, first 24 bits are network address)
IP Addressing

How to find the networks?

- Detach each interface from router, host
- create “islands of isolated networks

Interconnected system consisting of six networks
### IP Addresses

Given notion of “network”, let’s re-examine IP addresses:

“class-full” addressing:

<table>
<thead>
<tr>
<th>Class</th>
<th>Network</th>
<th>Host</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>host</td>
<td>1.0.0.0 to 127.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>network</td>
<td>128.0.0.0 to 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td>network</td>
<td>192.0.0.0 to 223.255.255.255</td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td>multicast address</td>
<td>224.0.0.0 to 239.255.255.255</td>
</tr>
</tbody>
</table>

32 bits
IP addressing: CIDR

- **classful addressing:**
  - inefficient use of address space, address space exhaustion
  - e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network

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

![Network and Host Parts](image)

```
11001000 00010111 00010000 00000000
200.23.16.0/23
```
IP addresses: how to get one?

Hosts (host portion):

- hard-coded by system admin in a file
- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address: “plug-and-play”
  - host broadcasts “DHCP discover” msg
  - DHCP server responds with “DHCP offer” msg
  - host requests IP address: “DHCP request” msg
  - DHCP server sends address: “DHCP ack” msg
IP addresses: how to get one?

Network (network portion):

- get allocated portion of ISP’s address space:

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>11001000 00010111 00010000 00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000 00010111 00010100 00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>...</td>
<td>......</td>
<td>..</td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000 00010111 00011110 00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>
Hierarchical addressing allows efficient advertisement of routing information:

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

ISP-R-Us

Fly-By-Night-ISP

“Send me anything with addresses beginning 200.23.16.0/20”

Internet

“Send me anything with addresses beginning 199.31.0.0/16”
Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1

Organization 0
- 200.23.16.0/23

Organization 2
- 200.23.20.0/23

Organization 7
- 200.23.30.0/23

Organization 1
- 200.23.18.0/23

ISP's-R-Us

Fly-By-Night-ISP

Internet

"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"
IP addressing: the last word...

**Q:** How does an ISP get block of addresses?

**A:** ICANN: Internet Corporation for Assigned Names and Numbers

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes
Getting a datagram from source to dest.

IP datagram:

<table>
<thead>
<tr>
<th>misc fields</th>
<th>source IP addr</th>
<th>dest IP addr</th>
<th>data</th>
</tr>
</thead>
</table>

- datagram remains unchanged, as it travels source to destination
- addr fields of interest here

```
378| 123 | 223.1.1.1   |
367| 223.1.2.2  |
357| 223.1.3.2  |
```

Routing table in A:

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

Diagram showing network connections with IP addresses and routing table.
Getting a datagram from source to dest.

Starting at A, given IP datagram addressed to B:

- look up net. address of B
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected
Getting a datagram from source to dest.

| misc fields | 223.1.1.1 | 223.1.2.3 | data |

Starting at A, dest. E:
- look up network address of E
- E on different network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- continued.....
Getting a datagram from source to dest.

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E
- E on same network as router’s interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!!
# IP datagram format

**IP protocol version number**
- 32 bits
- “type” of data

**Header length**
- (32 bit words)
- max number remaining hops (decremented at each router)

**Upper layer protocol to deliver payload to**
- 32 bit source IP address
- 32 bit destination IP address

**32 bits**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>IP protocol version number</td>
</tr>
<tr>
<td>type/service</td>
<td>“Type” of data</td>
</tr>
<tr>
<td>length</td>
<td>IP datagram length (bytes)</td>
</tr>
<tr>
<td>identifier</td>
<td>16-bit identifier</td>
</tr>
<tr>
<td>flgs</td>
<td>Flags for fragmentation/reassembly</td>
</tr>
<tr>
<td>fragment</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>offset</td>
<td>Internet checksum</td>
</tr>
<tr>
<td>time to live</td>
<td>Time to live</td>
</tr>
<tr>
<td>upper layer</td>
<td>Upper layer Protocol to deliver payload to</td>
</tr>
<tr>
<td>source IP address</td>
<td>32 bit source IP address</td>
</tr>
<tr>
<td>destination IP address</td>
<td>32 bit destination IP address</td>
</tr>
<tr>
<td>options (if any)</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>data</td>
<td>Data (variable length, typically a TCP or UDP segment)</td>
</tr>
</tbody>
</table>

E.g. timestamp, record route taken, specify list of routers to visit.
IP Fragmentation & Reassembly

- Network links have MTU (max. transfer size) - largest possible link-level frame.
  - Different link types, different MTUs
- Large IP datagram divided (“fragmented”) within net
  - One datagram becomes several datagrams
  - “reassembled” only at final destination
  - IP header bits used to identify, order related fragments

Fragmentation:
in: one large datagram
out: 3 smaller datagrams

Reassembly
IP Fragmentation and Reassembly

One large datagram becomes several smaller datagrams

- **Original Datagram:**
  - Length: 4000
  - ID: x
  - Offset: 0

- **First Fragment:**
  - Length: 1500
  - ID: x
  - Offset: 0

- **Second Fragment:**
  - Length: 1500
  - ID: x
  - Offset: 1480

- **Third Fragment:**
  - Length: 1040
  - ID: x
  - Offset: 2960
ICMP: Internet Control Message Protocol

- used by hosts, routers, gateways 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>
Router Architecture Overview

Two key router functions:
- run routing algorithms/protocol (RIP, OSPF, BGP)
- *switching* datagrams from incoming to outgoing link
Input Port Functions

Decentralized switching:
- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at ‘line speed’
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Physical layer: bit-level reception
Data link layer:
Input Port Queuing

- Fabric slower that input ports combined -> queueing may occur at input queues
- **Head-of-the-Line (HOL) blocking**: queued datagram at front of queue prevents others in queue from moving forward
- *queuing delay and loss due to input buffer overflow*

![Input Port Queuing Diagram]

- **output port contention at time t**: only one red packet can be transferred
- **green packet experiences HOL blocking**
Three types of switching fabrics

- **Memory**
- **Bus**
- **Crossbar**
Switching Via Memory

First generation routers:
- packet copied by system’s (single) CPU
- speed limited by memory bandwidth (2 bus crossings per datagram)

Modern routers:
- input port processor performs lookup, copy into memory
- Cisco Catalyst 8500
Switching Via Bus

- datagram from input port memory to output port memory via a shared bus
- **bus contention**: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)
Switching Via An Interconnection Network

- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network
Output Ports

- **Buffering** required when datagrams arrive from fabric faster than the transmission rate
- **Scheduling discipline** chooses among queued datagrams for transmission
Output port queuing

- buffering when arrival rate via switch exceeds output line speed
- *queuing (delay) and loss due to output port buffer overflow!*
IPv6

- **Initial motivation:** 32-bit address space completely allocated by 2008.

- **Additional motivation:**
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - new “anycast” address: route to “best” of several replicated servers

- **IPv6 datagram format:**
  - fixed-length 40 byte header
  - no fragmentation allowed
**IPv6 Header (Cont)**

*Priority:* identify priority among datagrams in flow

*Flow Label:* identify datagrams in same “flow.”

(concept of “flow” not well defined).

*Next header:* identify upper layer protocol for data

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td>payload len</td>
<td>next hdr</td>
<td>hop limit</td>
</tr>
</tbody>
</table>

| source address | (128 bits) |
| destination address | (128 bits) |

**data**

32 bits
Other Changes from IPv4

- **Checksum**: removed entirely to reduce processing time at each hop
- **Options**: allowed, but outside of header, indicated by “Next Header” field
- **ICMPv6**: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?

- Two proposed approaches:
  - Dual Stack: some routers with dual stack (v6, v4) can “translate” between formats
  - Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers
Dual Stack Approach
Tunneling

IPv6 inside IPv4 where needed