IP datagram format

- IP protocol version number
- Header length (bytes)
- "Type" of data
- 16-bit identifier
- Time to live (decremented at each router)
- Upper layer protocol to deliver payload to

<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>fragment offset for fragmentation/reassembly</td>
</tr>
<tr>
<td>length</td>
<td>Total datagram length (bytes)</td>
</tr>
<tr>
<td>identifier</td>
<td>16-bit identifier</td>
</tr>
<tr>
<td>flags</td>
<td>Flags</td>
</tr>
<tr>
<td>offset</td>
<td>Offset</td>
</tr>
<tr>
<td>time to live</td>
<td>Upper layer header checksum</td>
</tr>
<tr>
<td>upper layer</td>
<td>Options (if any)</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>data</td>
<td>Data (variable length, typically a TCP or UDP segment)</td>
</tr>
</tbody>
</table>

**How much overhead with TCP?**

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer
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
**IP Fragmentation and Reassembly**

**Example**

- **4000 byte datagram**
- **MTU = 1500 bytes**

One large datagram becomes several smaller datagrams

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>185</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1040</td>
<td>x</td>
<td>0</td>
<td>370</td>
</tr>
</tbody>
</table>

1480 bytes in data field

offset = 1480/8
• **IP address:** 32-bit identifier for host, router *interface*

• **interface:** connection between host/router and physical link

router’s typically have multiple interfaces
host typically has one interface

IP addresses associated with each interface

223.1.1.1 = 11011111 00000001 00000001 00000001
**Subnets**

- **IP address:**
  - subnet part (high order bits)
  - host part (low order bits)
- **What’s a subnet?**
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router

```
223.1.1.1 223.1.1.2 223.1.1.3
223.1.2.1 223.1.2.9 223.1.3.27
223.1.2.1 223.1.2.2
```

network consisting of 3 subnets
Recipe
To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.
Subnets

How many?
**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
Q: How does a *host* get an IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network-configuration->tcp/ip->properties
  - UNIX: /etc/rc.config

- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address from a server
  - “plug-and-play”
DHCP overview:
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
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

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
**Q:** How does *network* get subnet part of IP addr?

**A:** gets allocated portion of its provider 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

Fly-By-Night-ISP

- "Send me anything with addresses beginning 200.23.16.0/20"

ISPs-R-Us

- "Send me anything with addresses beginning 199.31.0.0/16"

Internet
Hierarchical addressing: more specific routes

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

ISP-R-Us

"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

Organization 1 200.23.18.0/23

ISP-R-Us

"Send me anything with addresses beginning 200.23.16.0/20"

Fly-By-Night-ISP

Organization 0 200.23.16.0/23

Organization 2 200.23.20.0/23

Organization 7 200.23.30.0/23

Internet
**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