CS 43: Computer Networks

16: Network Layer, IP

November 10, 2020
The Network Layer!

- **Application**: the application (e.g., the Web, Email)
- **Transport**: end-to-end connections, reliability
- **Network**: routing
- **Link (data-link)**: framing, error detection
- **Physical**: 1’s and 0’s/bits across a medium (copper, the air, fiber)
Network Layer Functions

• **Forwarding:** move packets from router’s input to appropriate router output
  – Look up in a table

• **Routing:** determine route taken by packets from source to destination.
  – Populating the table
IP Datagrams

- IP Datagrams are like a letter
  - Totally self-contained
  - Include all necessary addressing information
  - No advanced setup of connections or circuits

<table>
<thead>
<tr>
<th>Version</th>
<th>HLen</th>
<th>DSCP/ECN</th>
<th>Datagram Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Flags</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any, usually not)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data (variable len: typically TCP/UDP segment)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### IP Datagram Format

**IP protocol version number**

**Header length (in 32-bit words)**

**Version**

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>19</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td><strong>HLen</strong></td>
<td><strong>Type of Service</strong></td>
<td><strong>Datagram Length</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Flags</td>
<td>Offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source IP Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any, usually not)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IP Datagram Format

Fragmentation/ reassembly: Identifier, Flags, Offset
### IP Datagram Format

- **Version**: 4 bits
- **HLen**: 4 bits
- **Type of Service**: 8 bits
- **Datagram Length**: 16 bits
- **Identifier**: 16 bits
- **Flags**: 3 bits
- **Offset**: 13 bits
- **TTL**: 8 bits
- **Protocol**: 8 bits
- **Header Checksum**: 16 bits
- **Source IP Address**: 32 bits
- **Destination IP Address**: 32 bits
- **Options (if any, usually not)**
- **Data**: Variable length

- **max number remaining hops** (decremented at each router)
- **upper layer protocol** to deliver payload to TCP/UDP
**IP Datagram Format**

- **Version**
- **HLen**
- **Type of Service**
- **Datagram Length**
- **Identifier**
- **Flags**
- **Offset**
- **TTL**
- **Protocol**
- **Checksum**
- **Source IP Address**
- **Destination IP Address**
- **Options (if any, usually not)**
- **Data**

- max number remaining hops (decremented at each router)
- upper layer protocol to deliver payload to
- e.g. timestamp, record route taken, specify list of routers to visit.
### IP Datagrams

#### how much overhead?
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

<table>
<thead>
<tr>
<th>Version</th>
<th>HLen</th>
<th>DSCP/ECN</th>
<th>Datagram Length</th>
<th>Identifier</th>
<th>Flags</th>
<th>Offset</th>
<th>TTL</th>
<th>Protocol</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>19</td>
<td>24</td>
<td>31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Source IP Address**
- **Destination IP Address**
- **Options (if any, usually not)**
- **Data (variable len: typically TCP/UDP segment)**
IP Datagrams

Addresses must be unique on the network!

<table>
<thead>
<tr>
<th>Source endpoint.</th>
<th>Final destination endpoint.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>19</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>HLen</td>
<td>DSCP/ECN</td>
<td>Datagram Length</td>
<td>Identifier</td>
<td>Flags</td>
<td>Offset</td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Protocol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Source IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Options (if any, usually not)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data (variable len: typically TCP/UDP segment)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s in a name?

- **Host name:** web.cs.swarthmore.edu
  - **Domain:** registrar for each top-level domain (e.g., .edu)
  - **Host name:** local administrator assigns to each host

- **IP addresses:** 130.58.68.164
  - **Prefixes:** ICANN, regional Internet registries, and ISPs
  - **Hosts:** static configuration, or dynamic using DHCP

- **MAC addresses:** D8:D3:85:94:5F:1E
  - **OIDs:** assigned to vendors by the IEEE
  - **Adapters:** assigned by the vendor from its block
IP Addressing

- IP: 32-bit addresses
  - Usually written in dotted notation, e.g. 192.168.21.76
  - Each number is a byte
  - Stored in Big Endian order (network byte order)
IP Addresses

- $2^{32} = 4,294,967,296$ possible addresses.

- In the early 80’s, that’s a lot!
  - Population was ~4.5 billion.

- Now...not so much.
  - Population > 7 billion.
IP Prefixes

• Addresses are allocated in blocks called prefixes to organizations.

• Addresses in an N-bit prefix have the same top N bits.

• If an organization has an IP/N prefix, it can allocate $2^{32-N}$ addresses to end hosts on its network.

Network Prefix Length = N

Host Address Bits: 32-N
IP Prefixes

• Written in IP address/length notation

• Address is the lowest address in the allocated block. Length is prefix in bits.

• E.g. 128.13.0.0/16 is 128.13.0.0 to 128.13.255.255
  Read as: ”128.13.0.0 slash 16” prefix.
IP Prefixes

How would we express the following prefix?

```
00010010 00011111 00000000 xxxxxxxxxx
```

Network Prefix Length = N
Host Address Bits: 32-N

/24
Network Interfaces

- **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 or two interfaces (e.g., wired Ethernet, wireless 802.11)

- **IP addresses associated with each interface**
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
  - On the same link layer

223.1.1.1 = 11011111 00000001 00000001 00000001
223 1 1 1 1
Who gets an address? How many?

• Back in the old days, you called up Jon Postel
  – “How many addresses do you need?”
  – “Here you go! I may have rounded a bit.”
Assigning Addresses

• **IANA** – Internet Assigned Numbers Authority
  – (Run by Jon Postel until 1988)
  – Now a part of ICANN

• **ICANN**: Internet Corporation for Assigned Names and Numbers
  – Manages IP addresses, DNS, resolves disputes

---

**Diagram:**

- **IANA**
- **ARIN (US, CANADA)**
- **APNIC (Asia Pacific)**
- **RIPE (Europe)**
- **LACNIC (Latin America)**
- **AfriNIC (Africa)**

Flow from **IANA** to **ARIN**, **APNIC**, **RIPE**, **LACNIC**, and **AfriNIC**. From **ISPs** to **Customers**.
Who gets an address? How many?

- **Classful Addressing**
  - Class A: 8-bit prefix, 24 bits for hosts (16,777,216)
  - Class B: 16-bit prefix, 16 bits for hosts (65,536)
  - Class C: 24-bit prefix, 8 bits for hosts (256)
Classes of IP Addresses

**Class A**

- **Network**: 0-8
- **Host**: 16-31

Example: MIT 18.*.*.*

**Class B**

- **Network**: 0-16
- **Host**: 24-31

Example: NEU 129.10.*.*

**Class C**

- **Network**: 0-24
- **Host**: 24-31

Example: 216.63.78.*
CIDR

• Classless Inter-Domain Routing
  – Prefix (subnet) length is no longer fixed
  – (Can be division of bits rather than just 8/24, 16/16, and 24/8)
CIDR

• Classless Inter-Domain Routing
  – Prefix (subnet) length is no longer fixed
  – Address blocks come with a **subnet mask**
Classless Inter-Domain Routing (CIDR)

IP Address : 12.4.0.0     IP Mask: 255.254.0.0

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

• Classless Interdomain Routing
  – Prefix (subnet) length is no longer fixed
  – Address blocks come with a **subnet mask**

• Subnet mask written in two ways:
  – Dotted decimal: 255.255.240.0
  – /20
  – Both mean:
  
  
  11111111  11111111  11110000  00000000

  /20
CIDR

• Addresses divided into two pieces:
  – Prefix portion (network address)
  – Host portion

• Given an IP address and mask, we can determine:
  – The prefix (network address) by ANDing
  – The broadcast address by ORing inverted mask
Why might a device care about its “Network or Subnet Address”? 

- Answers the question: is the destination on the same subnet as me?

- Address + subnet mask $\rightarrow$ Network address

- If destination is on same network:
  - Send directly to them

- Else:
  - Send to gateway router
Network Address (Subnet Address)

IP Address & subnet mask -> Network Address

- E.g., 230.8.1.3/18 /18 => mask is 255.255.192.0

```plaintext
11100110 00001000 00000001 00000011
```

IP address

```plaintext
11111111 11111111 11000000 00000000
```

/Subnet mask
Network Address (Subnet Address)

- E.g., 230.8.1.3/18 /18 => mask is 255.255.192.0

Network address advertised by router: 230.8.0.0
Broadcast Address

- E.g., 230.8.1.3/18

\[
\begin{array}{cccccc}
11100110 & 00001000 & 00000001 & 00000011 \\
11111111 & 11111111 & 11000000 & 00000000 \\
00000000 & 00000000 & 00111111 & 11111111 \\
\end{array}
\]

- IP address
- /18 Subnet mask
- Complement of the subnet mask
Broadcast Address

- E.g., 230.8.1.3/18

<table>
<thead>
<tr>
<th>IP address</th>
<th>Complement of the subnet mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>11100110 00001000 00000001 00000011</td>
<td>00000000 00000000 00111111 11111111</td>
</tr>
</tbody>
</table>
Broadcast Address

- E.g., 230.8.1.3/18

\[ \begin{array}{cccc}
11100110 & 00001000 & 00000001 & 00000011 \\
00000000 & 00000000 & 00111111 & 11111111 \\
11100110 & 00001000 & 00111111 & 11111111 \\
\end{array} \]

Broadcast address: 230.8.63.255
Datagram forwarding table

<table>
<thead>
<tr>
<th>dest address</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-range 1</td>
<td>3</td>
</tr>
<tr>
<td>address-range 2</td>
<td>2</td>
</tr>
<tr>
<td>address-range 3</td>
<td>2</td>
</tr>
<tr>
<td>address-range 4</td>
<td>1</td>
</tr>
</tbody>
</table>

4 billion IP addresses, try to aggregate table entries

IP destination address in arriving packet’s header
Routers exchange state (we’ll save the what and when for later). They decide, for each destination, how to get there, and build a lookup structure for their forwarding table. What should they build?

A. A list – scan for the destination.

B. A hash table – look up the destination.

C. A tree – Follow branches that lead to the destination.

D. Some other software structure.

E. We can’t do this in software, we need special hardware.
Look-up Algorithm

- Protocol: ATM (Virtual Circuits), Ethernet (Flat addresses)
  - Mechanism: Exact Match
  - Techniques: Direct lookup, Hash Tables, Binary Trees
- Protocol: IPv4, IPv6
  - Mechanism: Longest Prefix Match
  - Techniques: Prefix Trees, TCAM (Ternary Content Addressable Memories)
## Datagram forwarding table

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>200.23.16.* through 200.23.23.*</td>
<td>0</td>
</tr>
<tr>
<td>200.23.24.0 through 200.23.24.255</td>
<td>1</td>
</tr>
<tr>
<td>200.23.25.* through 200.23.31.*</td>
<td>2</td>
</tr>
<tr>
<td>Otherwise (default gateway)</td>
<td>3</td>
</tr>
</tbody>
</table>
### Datagram forwarding table

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111</td>
<td>2</td>
</tr>
<tr>
<td>Otherwise (default gateway)</td>
<td>3</td>
</tr>
</tbody>
</table>
Longest prefix matching

In a forwarding table entry, use the **longest address prefix** that matches destination address.

<table>
<thead>
<tr>
<th>Destination IP Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;upper 16 bit&gt; 00010*** ********************</td>
<td>0</td>
</tr>
<tr>
<td>&lt;upper 16 bit&gt; 00011000  **********</td>
<td>1</td>
</tr>
<tr>
<td>&lt;upper 16 bit&gt; 00011***  **********</td>
<td>2</td>
</tr>
<tr>
<td>Otherwise (default gateway)</td>
<td>3</td>
</tr>
</tbody>
</table>

**DA:** <upper 16 bits> 00011000  10101010
DA: <upper 16 bits> 00010110  10100001

which interface?
Router architecture overview

- high-level view of generic router architecture:

  - router input ports: (different from TCP ports!!)
    these are physical inputs/outputs to the router

  - high-speed switching fabric

  - routing processor

  - router output ports

  - forwarding data plane (hardware) operates in nanosecond timeframe

  - routing, management control plane (software) operates in millisecond time frame

  - crossbar
Input port functions

- **Physical layer:**
  - Bit-level reception

- **Link layer:**
  - E.g., Ethernet (chapter 6)

- **Decentralized switching:**
  - Using header field values, lookup output port using forwarding table in input port memory ("match plus action")
Longest prefix matching

In a forwarding table entry, use the longest address prefix that matches destination address.

<table>
<thead>
<tr>
<th>Destination IP Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;upper 16 bit&gt; 00010*** ***********</code></td>
<td>0</td>
</tr>
<tr>
<td><code>&lt;upper 16 bit&gt; 00011000 ***********</code></td>
<td>1</td>
</tr>
<tr>
<td><code>&lt;upper 16 bit&gt; 00011*** ***********</code></td>
<td>2</td>
</tr>
<tr>
<td>Otherwise (default gateway)</td>
<td>3</td>
</tr>
</tbody>
</table>
Binary Prefix Tree

• Store the prefixes as a tree
  – Prefixes “spelled out” following a path from the root
  – One bit for each level of the tree
  – Some nodes correspond to valid prefixes
  – ... which have next-hop interfaces in a table

• When a packet arrives
  – Traverse the tree based on the destination address
  – Stop upon reaching the longest matching prefix

<table>
<thead>
<tr>
<th>Prefix Range-1</th>
<th>0*</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix Range-2</td>
<td>00*</td>
<td>2</td>
</tr>
<tr>
<td>Prefix Range-3</td>
<td>11*</td>
<td>3</td>
</tr>
</tbody>
</table>

Depth = W
Degree = 2
Stride = 1 bit
Multi-bit Prefix Tree

- Store the prefixes as a tree: 4-ary tree
  - k bits for each level of the tree

<table>
<thead>
<tr>
<th>Prefix Range</th>
<th>Prefix</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix Range-1</td>
<td>111*</td>
<td>1</td>
</tr>
<tr>
<td>Prefix Range-2</td>
<td>10*</td>
<td>2</td>
</tr>
<tr>
<td>Prefix Range-3</td>
<td>1010*</td>
<td>3</td>
</tr>
<tr>
<td>Prefix Range-4</td>
<td>10101*</td>
<td>4</td>
</tr>
</tbody>
</table>

Depth = W/k
Degree = 2^k
Stride = k bits
Even Faster Lookups

• Can use special hardware
  – Content Addressable Memories (CAMs)
  – Allows look-ups on a key rather than flat address

• Huge innovations in the mid-to-late 1990s
  – After CIDR was introduced (in 1994)
  – ... and longest-prefix match was a major bottleneck
Hierarchical Addressing: Route Aggregation

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

ISPs-R-Us

“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16”

Internet
Hierarchical Addressing: Route Aggregation

“Send me anything with addresses beginning 200.23.16.0/20” translates to the following:

\[
200.23.16.0/20 = \begin{array}{cccc}
200 & 23 & 16 & 0 \\
11001000 & 00010111 & 00010000 & 00000000
\end{array}
\]

/20 Prefix

\[
200.23.16.0 = \begin{array}{cccc}
200 & 23 & 16 & 0 \\
11001000 & 00010111 & 00010000 & 00000000
\end{array}
\]

/20 Prefix

\[
200.23.31.255 = \begin{array}{cccc}
200 & 23 & 16 & 0 \\
11001000 & 00010111 & 00011111 & 11111111
\end{array}
\]

Range represented by the /20 prefix

/20 prefix contains the range of IP addresses that match the first 20 bits, and can have any value for the remaining 12 bits in the range of:

[first 20 bits] 0000 00000000
[first 20 bits] 1111 11111111

A total of \(2^{12} = 4,096\) IP addresses
Route aggregation in Fly-By-Night ISP

Fly-By-Night-ISP

200.23.16.0/20 = 11001000 00010111 00010000 00000000

Individual Organizations: All of these organizations IP addresses lie within Fly-by-Night’s /20 prefix (first 20 bits are the same)

- they more specifically match on the three more bits to form a /23 prefix (first 23 bits of all IP addresses within their organization are the same).
- The last 9 (32-23) bits provide $2^9 = 512$ unique IP addresses within each organization.

/23 prefixes

200.23.16.0/23 = 11001000 00010111 00010000 00000000
200.23.18.0/23 = 11001000 00010111 00010001 00000000
200.23.20.0/23 = 11001000 00010111 00010010 00000000
200.23.30.0/23 = 11001000 00010111 00011110 00000000
What should we do if organization 1 decides to switch to ISPs-R-Us?

“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16”
What should we do if organization 1 decides to switch to ISPs-R-Us?

A. Move 200.23.18.0/23 to ISPs-R-Us (and break up Fly-By-Night’s /20 block).
B. Give new addresses to Organization 1 (and force them to change all their addresses).
C. Some other solution.
ISPs-R-Us has a more specific route to Organization 1

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

“Send me anything with addresses beginning 200.23.16.0/20”
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

ISPs-R-Us

Fly-By-Night-ISP

Longest prefix matching!

“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”
Longest Prefix Matching at Router 1

Now, when an incoming packet addressed with destination address 200.23.18.5 arrives – this address belongs to Organization 1 and the packet will be matched using longest prefix matching and will be routed to ISPs-R-Us rather than the Fly-by-Night ISP.

<table>
<thead>
<tr>
<th>dest address</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>200.23.16.0/20 = 11001000 00010111 00010000 00000000</td>
<td>(to Fly-by-Night ISP)</td>
</tr>
<tr>
<td>199.31.0.0/16 = 11000111 00011111 00000000 00000000</td>
<td>(to ISPs-R-Us)</td>
</tr>
<tr>
<td>200.23.18.0/23 = 11001000 00010111 00010010 00000000</td>
<td>(to ISPs-R-Us)</td>
</tr>
</tbody>
</table>