IP Addressing
Evolution of Internet Structure

- Internet c. 1990
  - Tree structure, centered around one backbone
  - National Science Foundation (NSF) funded

End users

National backbone

Service provider networks

Berkeley

BARRNET Regional

PARC

Berkeley

NCAR

WestNet Regional

UA

MidNet Regional

UNM

UNL

KU

ISU
An Old Internet ISP Map
A New Internet Map
Another Internet Map
Evolution of Internet Structure

- **Today**
  - Multiple backbone service providers
  - Arbitrary graph structure

![Diagram showing the evolution of Internet Structure with various service providers and regional networks.](attachment://image.png)
Problems of Scale

- Main problems
  - Inefficient address allocation
  - Too many networks for routing
IPv4 Address Model

- Properties
  - 32-bit address
  - Hierarchical
    - Network, subnet, host hierarchy
  - Maps to logically unique network adaptor
    - Exceptions: service request splitting for large web servers

- Three Class Model

<table>
<thead>
<tr>
<th>Class A:</th>
<th>0</th>
<th>Network (7 bits)</th>
<th>Host (24 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B:</td>
<td>1 0</td>
<td>Network (14 bits)</td>
<td>Host (16 bits)</td>
</tr>
<tr>
<td>Class C:</td>
<td>1 1 0</td>
<td>Network (21 bits)</td>
<td>Host (8 bits)</td>
</tr>
</tbody>
</table>
# IPv4 Address Model

<table>
<thead>
<tr>
<th>Class</th>
<th>Network ID</th>
<th>Host ID</th>
<th># of Addresses</th>
<th># of Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 + 7 bit</td>
<td>24 bit</td>
<td>$2^{24}-2$</td>
<td>126</td>
</tr>
<tr>
<td>B</td>
<td>10 + 14 bit</td>
<td>16 bit</td>
<td>$65,536 - 2$</td>
<td>$2^{14}$</td>
</tr>
<tr>
<td>C</td>
<td>110 + 21 bit</td>
<td>8 bit</td>
<td>$256 - 2$</td>
<td>$2^{21}$</td>
</tr>
<tr>
<td>D</td>
<td>1110 + Multicast Address</td>
<td></td>
<td>IP Multicast</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td>Future Use</td>
<td></td>
</tr>
</tbody>
</table>
Basic Datagram Forwarding with IP

- Hosts and routers maintain forwarding tables
  - List of <prefix, next hop> pairs
    - IP = 69.2.1.2  = 01000101 00000010 00000001 00000010
    - 24-bit prefix  = 69.2.1.0/24
      = 01000101 00000010 00000000 00000000
  - Often contains a default route
    - Pass unknown destination to provider ISP
  - Simple and static on hosts, edge routers
    - Complex and dynamic on core routers
Basic Datagram Forwarding with IP

Packet forwarding
- Compare network portion of address with <network/host, next hop> pairs in table
  - Send directly to host on same network
  - Send to indirectly (via router on same network) to host on different network
- Use ARP to get hardware address of host/router
IPv4 Address Model

- **IP addresses**
  - Host in class A network
    - 56.0.78.100
    - [www.usps.gov](http://www.usps.gov)
  - Host in class B network
    - 128.174.252.1
    - [www.cs.uiuc.edu](http://www.cs.uiuc.edu)
  - Host in class C network
    - 198.182.196.56
    - [www.linux.org](http://www.linux.org)

- **Questions**
  - What networks should be allocated to a company with 1000 machines?
  - What about a company with 100 machines?
  - What about a company with 2 machines that plans to grow rapidly?
Problems of Scale

- Pressure mostly on class B networks
  - Most companies plan to grow beyond 255 machines
  - Renumbering is time consuming and can interrupt service
  - Approximately 16,000 class B networks available

- Class B networks aren’t very efficient
  - Few organizations have $O(10,000)$ machines
  - More likely use $O(1,000)$ of the 65,000 addresses

- Scaling problems with alternatives
  - Multiple class C networks
    - Routing tables don’t scale
  - Protocols do not scale beyond $O(10,000)$ networks
IP Address Hierarchy
Evolution

- Began with class based system
  - Subnetting within an organization
    - Network can be broken into smaller networks
    - Recognized only within the organization
    - Implemented by packet switching
    - Smaller networks called subnets

Class A:

```
0 | Network (7 bits) | Host (24 bits)
```

Class B:

```
1 0 | Network (14 bits) | Host (16 bits)
```

Class C:

```
1 1 0 | Network (21 bits) | Host (8 bits)
```
Subnetting

- **Simple IP**
  - All hosts on the same network must have the same *network* number

- **Assumptions**
  - Subnets are close together
    - Look like one network to distant routers

- **Idea**
  - Take a single IP network number
  - Allocate the IP addresses to several physical networks (subnets)

- **Subnetting**
  - All hosts on the same network must have the same *subnet* number
Subnetting

- Enables a domain to further partition address space into smaller networks
  - Subdivide host id into subnet ID + host ID
  - Subnet mask

- Only routers in the domain interpret subnet mask
  - Other routers treat IP address as normal class A, B or C address
Subnet Example

- Consider
  - A domain with a class B address
  - 135.104.*

- Without subnetting
  - Every router in the domain needs to know how to route to every host

- However
  - the domain itself is likely organized as a hierarchy of physical networks
Subnet Example

Solution

- Partition the 65,536 address in the class B network
  - 256 subnets each with 256 addresses
  - Subnet mask: 255.255.255.0

- If 135.104.5.{1,2,3} are all on the same physical network reachable from router 135.105.4.1
  - There only needs to be one routing entry for 135.104.5.* pointing to 135.105.4.1 as next hop
Subnetting

- Normal IP

Class B:

```
1 0
Network (14 bits) Host (16 bits)
```

- Typical subnetting example
  - Use first byte of host as subnet number

Class B:

```
1 0
Network (14 bits) Subnet (8 bits) Host (8 bits)
```

- Atypical example
  - Non-contiguous 6-bit subnet number

Class B:

```
1 0
Network (14 bits)
```

Non-contiguous 6-bit subnet number.
Subnetting

- The subnet mask specifies the bits of network and subnet addresses
- Routing table entries carry both addresses and subnet masks

Class B:

```
1 0  Network (14 bits)    Host (16 bits)
```

Class B:

```
1 0  Network (14 bits)    Subnet (8 bits)    Host (8 bits)
```

Subnet Mask:

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
```
Subnetting – Host 1

Subnet Mask: 255.255.255.128
Subnet Number: 128.174.142.128

Host 1: 128.174.142.200
1 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 0 0 0

Subnet Mask 255.255.255.128
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0

Subnet # 128.174.142.128
1 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0
Subnetting – Host 3

Subnet Number: 128.174.141.0
Subnet Mask: 255.255.255.0

Host 3: 128.174.141.3

Subnet Mask 255.255.255.0

Subnet # 128.174.141.0
Subnetting - Example

Subnet Mask: 255.255.255.128
Subnet Number: 128.174.142.128

Subnet Mask: 255.255.255.0
Subnet Number: 128.174.141.0

H1
128.174.142.200

H3
128.174.141.3

R1
Subnet Mask: 255.255.255.128
Subnet Number: 128.174.142.0

R2
Subnet Mask: 255.255.255.0
Subnet Number: 128.174.141.0

H2
128.174.142.27
Send from H1 to H3

At H1:
- Compute (H3 AND H1’ s subnet mask)
  - 128.174.141.3 AND 255.255.255.128
  - = 128.174.141.0 (≠ 128.174.142.128)

- If result == H1’ s subnet number
  - H3 and H1 are on the same subnet
- else
  - route through appropriate router
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

Example Table from R2

- 196 = 1100 0100
- 128 = 1000 0000
- 141 = 1000 1101
- 142 = 1000 1110
- 145 = 1001 0001
- 196 = 1100 0100

Next hop
- 128.174.142.196
- 128.174.142.95
- 128.174.141.137
- 128.174.145.18
- 131.126.244.15
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

Example Table from R2
- **Next hop**
  - 128.174.142.196
  - 128.174.142.95
  - 128.174.141.137
  - 128.174.145.18
  - 131.126.244.15

\[196 = 1100 0100\]
\[128 = 1000 0000\]
\[141 = 1000 1101\]
\[142 = 1000 1110\]
\[145 = 1001 0001\]
\[196 = 1100 0100\]
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

Example Table from R2

- Next hop
  - 128.174.142.196
  - 128.174.142.95
  - 128.174.141.137
  - 128.174.145.18
  - 131.126.244.15

- 95 = 0101 1111
- 128 = 1000 0000
- 141 = 1000 1101
- 142 = 1000 1110
- 145 = 1001 0001
- 196 = 1100 0100
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

Example Table from R2

- Next hop
  - 128.174.142.196
  - 128.174.142.95
  - 128.174.141.137
  - 128.174.145.18
  - 131.126.244.15

95 = 0101 1111
128 = 1000 0000
141 = 1000 1101
to Interface 1
142 = 1000 1110
145 = 1001 0001
196 = 1100 0100
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

- Example Table from R2
  - Next hop
    - 128.174.142.196
    - 128.174.142.95
    - 128.174.141.137
    - 128.174.145.18
    - 131.126.244.15

137 = 1000 1001
128 = 1000 0000
141 = 1000 1101
142 = 1000 1110
145 = 1001 0001
196 = 1100 0100
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

- **Example Table from R2**
  - **Next hop**
    - 128.174.142.196
    - 128.174.142.95
    - 128.174.141.137
    - 128.174.145.18
    - 131.126.244.15

  - 137 = 1000 1001
  - 128 = 1000 0000
  - 141 = 1000 1101
  - 142 = 1000 1110
  - 145 = 1001 0001
  - 196 = 1100 0100
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

### Example Table from R2

- **Next hop**
  - 128.174.142.196
  - 128.174.142.95
  - 128.174.141.137
  - 128.174.145.18
  - 131.126.244.15

18 = 0001 0010
128 = 1000 0000
141 = 1000 1101
142 = 1000 1110
145 = 1001 0001
196 = 1100 0100
# Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

- **Example Table from R2**
  - **Next hop**
    - 128.174.142.196
    - 128.174.142.95
    - 128.174.141.137
    - 128.174.145.18
    - 131.126.244.15

  - $18 = 0001\ 0010$
  - $128 = 1000\ 0000$
  - $141 = 1000\ 1101$
  - $142 = 1000\ 1110$
  - $145 = 1001\ 0001$
  - $196 = 1100\ 0100$

  to R3
Routing with Subnetting

Example Table from R2

- Next hop
  - 128.174.142.196
  - 128.174.142.95
  - 128.174.141.137
  - 128.174.145.18
  - 131.126.244.15

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>
# Routing with Subnetting

**Example Table from R2**

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

- **Next hop**
  - 128.174.142.196
  - 128.174.142.95
  - 128.174.141.137
  - 128.174.145.18
  - 131.126.244.15

- **Subnet Values**
  - 15 = 0000 1111
  - 128 = 1000 0000
  - 141 = 1000 1101
  - 142 = 1000 1110
  - 145 = 1001 0001
  - 196 = 1100 0100

- **To R3**
Routing with Subnetting

<table>
<thead>
<tr>
<th>Subnet #</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.174.141.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.174.142.0</td>
<td>255.255.255.128</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.174.142.128</td>
<td>255.255.255.128</td>
<td>R1</td>
</tr>
<tr>
<td>128.174.0.0</td>
<td>255.255.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>Default</td>
<td>0.0.0.0</td>
<td>R3</td>
</tr>
</tbody>
</table>

- Example Table from R2
  - Next hop
    - 128.174.142.196 to R1
    - 128.174.142.95 to Interface 1
    - 128.174.141.137 to Interface 0
    - 128.174.145.18 to R3
    - 131.126.244.15 to R3
Subnetting

Notes
- Non-contiguous subnets are difficult to administer
- Multiple subnets on one physical network
  - Must be routed through router

Pros
- Helps address consumption
- Helps reduce routing table size
The Crisis

- Fixed 32-bit address space for IPv4
- Network allocation based on Classic A, B, C Model
- Central allocation authority
  - Randomly assigning addresses
- Problems
  - Router table explosion
  - Address space exhaustion
Classless Interdomain Routing (CIDR)

- **CIDR/Supernetting**
  - Problem with subnetting
    - Allows hierarchy within organizations
    - Does not reduce class B address space pressure
  - Solution
    - Aggregate routes in routing tables
    - Eliminate class notation
    - Generalize subnet notion
    - Allow only contiguous subnet masks
    - Specify network by \(<\text{network #}, \# \text{ of bits in subnet mask}>\)
    - Equivalent to \(<\text{network #}, \# \text{ of hosts}>\)
    - Blocks of class C networks can now be treated as one network
CIDR

- Route aggregation
  - Use contiguous blocks of Class C addresses
    - Example:
      - 192.4.16 – 192.4.31
      - 20 bit subnet mask
    - Block size must be a power of 2
      - Network number may be any length

192.4.16.0

```
11000000000000000100000010000000000000000
```

192.4.31.0

```
1100000000000000010000001111100000000000
```

Subnet Mask

```
1111111111111111111111111111100000000000
```
CIDR is similar to subnetting

- Trend is for increasing amounts of overlap in routing table entries
- Example: 128.174.142.200
  - Matches second, third and fourth lines
  - Route to entry with longest match
CIDR

Subnet: 128.174.141.0
\[100000000010101110100011010000000000\]

Subnet Mask length = 24 (255.255.255.0)
\[111111111111111111111111111000000000\]

Host: 128.174.142.200
\[1000000000101011101000111011001000\]

Resulting Subnet Number: 128.174.142.0 (≠ 128.174.141.0)
\[10000000001010111010001110000000000\]

Subnet: 128.174.142.192
\[100000000010101110100011101100000000\]

Subnet Mask length = 27 (255.255.255.224)
\[1111111111111111111111111110000000\]

Host: 128.174.142.200
\[1000000000101011101000111011001000\]

Resulting Subnet Number: 128.174.142.192 (≠ 128.174.142.192)
\[10000000001010111010001110110000000\]
CIDR

Subnet: 128.174.142.128

Resulting Subnet Number: 128.174.142.128 (= 128.174.142.128)

Subnet Mask length = 25 255.255.255.192

Host: 128.174.142.200

Resulting Subnet Number: 128.174.142.128 (= 128.174.142.128)

Subnet: 128.174.0.0

Resulting Subnet Number: 128.174.0.0 (= 128.174.0.0)

Subnet Mask length = 16 255.255.0.0
CIDR

- Subnetting
  - Share one address (network number) across multiple physical networks

- Supernetting
  - Aggregate multiple addresses (network numbers) for one physical network
CIDR

- Allows hierarchical development
  - Assign a block of addresses to a regional provider
    - Ex: 128.0.0.0/9 to BARRNET
  - Regional provider subdivides address and hands out block to sub-regional providers
    - Ex: 128.132.0.0/16 to Berkeley
  - Sub-regional providers can divide further for smaller organizations
    - Ex: 128.132.32.0/1 to Berkeley Computer Science Department
Pros and Cons

- Provides a fast easy solution
- Was not intended to be permanent
- Multihomed sites cannot benefit from aggregation
- Not backward compatible
IPv6

- History
  - Next generation IP (AKA IPng)
  - Intended to extend address space and routing limitations of IPv4
    - Requires header change
    - Attempted to include everything new in one change
  - IETF moderated
    - Based on Simple Internet Protocol Plus (SIPP)
IPv6

- Wish list
  - 128-bit addresses
  - Multicast traffic
  - Mobility
  - Real-time traffic/quality of service guarantees
  - Authentication and security
  - Autoconfiguration for local IP addresses
  - End-to-end fragmentation
  - Protocol extensions

- Smooth transition!

- Note
  - Many of these functionalities have been retrofit into IPv4
IPv6 Addresses

- 128-bit
  - 3.4 x 10^38 addresses (as compared to 4 x 10^9)
- Classless addressing/routing (similar to CIDR)

- Address notation
  - String of eight 16-bit hex values separated by colons
    - 5CFA:0002:0000:0000:CF07:1234:5678:FFCD
  - Set of contiguous 0’s can be elided
    - 5CFA:0002::0000:CF07:1234:5678:FFCD

- Address assignment
  - Provider-based
  - geographic

```
010  Region ID  Provider ID  Subscriber ID  Subnet  Host
    3     m      n       o      p  125-m-n-o-p
```
### IPv4 Packet Format

- **20 Byte minimum**
- **Mandatory fields are not always used**
  - e.g. fragmentation
- **Options are an unordered list of (name, value) pairs**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>version</td>
</tr>
<tr>
<td>8</td>
<td>hdr len</td>
</tr>
<tr>
<td>16</td>
<td>TOS</td>
</tr>
<tr>
<td></td>
<td>length</td>
</tr>
<tr>
<td></td>
<td>ident</td>
</tr>
<tr>
<td></td>
<td>flags</td>
</tr>
<tr>
<td></td>
<td>offset</td>
</tr>
<tr>
<td></td>
<td>TTL</td>
</tr>
<tr>
<td></td>
<td>protocol</td>
</tr>
<tr>
<td></td>
<td>checksum</td>
</tr>
<tr>
<td></td>
<td>source address</td>
</tr>
<tr>
<td></td>
<td>destination address</td>
</tr>
<tr>
<td></td>
<td>options (variable)</td>
</tr>
<tr>
<td></td>
<td>pad (variable)</td>
</tr>
</tbody>
</table>
### IPv6 Packet Format

- 40 Byte minimum
- Mandatory fields (almost) always used
- Strict order on options reduces processing time
  - No need to parse irrelevant options

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>priority</td>
<td>flow label</td>
<td></td>
</tr>
<tr>
<td>payload length</td>
<td>next header</td>
<td>hop limit</td>
<td></td>
</tr>
<tr>
<td>source address 4 words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address 4 words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>options (variable number, usually fixed length)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IPv6 Packet Format

- Version
  - 6
- Priority and Flow Label
  - Support service guarantees
  - Allow “fair” bandwidth allocation
- Payload Length
  - Header not included
- Next Header
  - Combines options and protocol
  - Linked list of options
  - Ends with higher-level protocol header (e.g. TCP)
- Hop Limit
  - TTL renamed to match usage
IPv6 Extension Headers

- Must appear in order
  - Hop-by-hop options
    - Miscellaneous information for routers
  - Routing
    - Full/partial route to follow
  - Fragmentation
    - IP fragmentation info
  - Authentication
    - Sender identification
  - Encrypted security payload
    - Information about contents
  - Destination options
    - Information for destination
IPv6 Extension Headers

- **Hop-by-Hop extension**
  - Length is in bytes beyond mandatory 8

  *Diagram with a table showing fields next header, length, and type*

- **Jumbogram option (packet longer than 65,535 bytes)**
  - Payload length in main header set to 0

  *Diagram with a table showing next header, length, and type*

Payload length in bytes
## IPv6 Extension Headers

### Routing extension
- Up to 24 “anycast” addresses target AS’ s/providers
- Next address tracks current target
- Strict routing requires direct link
- Loose routing allows intermediate nodes

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>next header</td>
<td>0</td>
<td># of addresses</td>
<td>next address</td>
</tr>
<tr>
<td>strict/loose routing bitmap</td>
<td>1 – 24 addresses</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IPv6 Extension Headers

- **Fragmentation extension**
  - Similar to IPv4 fragmentation
    - 13-bit offset
    - Last fragment mark (M)
  - Larger fragment identification field
IPv6 Extension Headers

- Authentication extension
  - Designed to be very flexible
  - Includes
    - Security parameters index (SPI)
    - Authentication data

- Encryption Extension
  - Called encapsulating security payload (ESP)
  - Includes an SPI
  - All headers and data after ESP are encrypted
IPv6 Design Controversies

- Address length
  - 8 byte
    - Might run out in a few decades
    - Less header overhead
  - 16 byte
    - More overhead
    - Good for foreseeable future
  - 20 byte
    - Even more overhead
    - Compatible with OSI
  - Variable length
IPv6 Design Controversies

- Hop limit
  - 65,535
    - 32 hop paths are common now
    - In a decade, we may see much longer paths
  - 255
    - Objective is to limit lost packet lifetime
    - Good network design makes long paths unlikely
      - Source to backbone
      - Across backbone
      - Backbone to destination
IPv6 Design Controversies

- Greater than 64KB data
  - Good for supercomputer/high bandwidth applications
  - Too much overhead to fragment large data packets

- 64 KB data
  - More compatible with low-bandwidth lines
  - 1 MB packet ties up a 1.5MBps line for more than 5 seconds
  - Inconveniences interactive users
IPv6 Design Controversies

- Keep checksum
  - Removing checksum from IP is analogous to removing brakes from a car
    - Light and faster
    - Unprepared for the unexpected

- Remove checksum
  - Typically duplicated in data link and transport layers
  - Very expensive in IPv4
IPv6 Design Controversies

- Mobile hosts
  - Direct or indirect connectivity
    - Reconnect directly using canonical address
    - Use home and foreign agents to forward traffic
  - Mobility introduces asymmetry
    - Base station signal is strong, heard by mobile units
    - Mobile unit signal is weak and susceptible to interference, may not be heard by base station
IPv6 Design Controversies

- **Security**
  - **Where?**
    - Network layer
      - A standard service
    - Application layer
      - No viable standard
      - Application susceptible to errors in network implementation
      - Expensive to turn on and off
  - **How?**
    - Political import/export issues
    - Cryptographic strength issues
Network Address Translation (NAT)

- Kludge (but useful)
- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)
NAT Illustration

Operation: S wants to talk to D:
- Create S-SN mapping
- Replace S with SN for outgoing packets
- Replace SN with S for incoming packets
What if we only have few (or just one) IP address?

- Use NAPT (Network Address Port Translator)
- NAPT translates:
  - <Paddr1, portA> to <Gaddr, portB>
  - potentially thousands of simultaneous connections with one global IP address
Problems with NAT

- Hides the internal network structure
  - some consider this an advantage
- Multiple NAT hops must ensure consistent mappings
- Some protocols carry addresses
  - e.g., FTP carries addresses in text
  - what is the problem?
- Encryption
NAT: Network Address Translation

- **Approach**
  - Assign one router a global IP address
  - Assign internal hosts local IP addresses

- **Change IP Headers**
  - IP addresses (and possibly port numbers) of IP datagrams are replaced at the boundary of a private network
  - Enables hosts on private networks to communicate with hosts on the Internet
  - Run on routers that connect private networks to the public Internet
NAT: Network Address Translation

- Outgoing packet
  - Source IP address (private IP) replaced by global IP address maintained by NAT router

- Incoming packet
  - Destination IP address (global IP of NAT router) replaced by appropriate private IP address

What address do the remote hosts respond to?

NAT router caches translation table:
(source IP address, port #) ➔
(NAT IP address, new port #)
NAT: Network Address Translation

- Benefits: local network uses just one (or a few) IP address as far as outside world is concerned
  - No need to be allocated range of addresses from ISP
    - Just one IP address is used for all devices
  - Can change addresses of devices in local network without notifying outside world
  - Can change ISP without changing addresses of devices in local network
  - Devices inside local net not explicitly addressable, visible by outside world (a security plus)
NAT: Network Address Translation

1: host 10.0.0.1 sends datagram to 128.119.40, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

<table>
<thead>
<tr>
<th>NAT translation table</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WAN side addr</td>
<td>LAN side addr</td>
</tr>
<tr>
<td>138.76.29.7, 5001</td>
<td>10.0.0.1, 3345</td>
</tr>
<tr>
<td>……</td>
<td>……</td>
</tr>
</tbody>
</table>

3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345
NAT: Network Address Translation

- Address Pooling
  - Corporate network has many hosts
  - Only a small number of public IP addresses

- NAT solution
  - Manage corporate network with a private address space
  - NAT, at boundary between corporate network and public Internet, manages a pool of public IP addresses
  - When a host from corporate network sends an IP datagram to a host in public Internet, NAT picks a public IP address from the address pool, and binds this address to the private address of the host
NAT: Network Address Translation

- Load balancing
  - Balance the load on a set of identical servers, which are accessible from a single IP address

- NAT solution
  - Servers are assigned private addresses
  - NAT acts as a proxy for requests to the server from the public network
  - NAT changes the destination IP address of arriving packets to one of the private addresses for a server
  - Balances load on the servers by assigning addresses in a round-robin fashion
NAT: Consequences

- 16-bit port-number field
  - 60,000 simultaneous connections with a single LAN-side address!

- End-to-end connectivity
  - NAT destroys universal end-to-end reachability of hosts on the Internet
  - A host in the public Internet often cannot initiate communication to a host in a private network
  - The problem is worse, when two hosts that are in different private networks need to communicate with each other
NAT: Consequences

- Performance
  - Modifying the IP header by changing the IP address requires that NAT boxes recalculate the IP header checksum
  - Modifying port number requires that NAT boxes recalculate TCP checksum

- Fragmentation
  - Datagrams fragmented before NAT device must not be assigned different IP addresses or different port numbers
NAT: Consequences

IP address in application data

- Applications often carry IP addresses in the payload of the application data
- No longer work across a private-public network boundary
- Hack: Some NAT devices inspect the payload of widely used application layer protocols and, if an IP address is detected in the application-layer header or the application payload, translate the address according to the address translation table