IP Address Classes
(Some are Obsolete)

<table>
<thead>
<tr>
<th>Class</th>
<th>Network ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>8-16</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>16-24</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td>24-32</td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1111</td>
<td></td>
</tr>
</tbody>
</table>

- **Class A**: Network ID
- **Class B**: 2 octets (16 bits)
- **Class C**: 3 octets (24 bits)
- **Class D**: Multicast Addresses
- **Class E**: Reserved for experiments

Subnetting
- Add another layer to hierarchy
- Variable length subnet masks
  - Could subnet a class B into several chunks

Important Concepts
- Hierarchical addressing critical for scalable system
  - Don’t require everyone to know everyone else
  - Reduces number of updates when something changes
  - Interaction with routing tables
- Sub-netting simplifies network management
  - Break up the network into smaller chunks
  - Managed internally in network
Outline

- CIDR addressing
- IP protocol
- IPv6
- NATs

IP Address Problem (1991)

- Address space depletion
  - Suppose you need $2^{16} + 1$ addresses?
  - In danger of running out of classes A and B
    - Class C too small for most domains
    - Very few class A – very careful about using them
    - Class B – greatest problem
  - Class B sparsely populated
    - But people refuse to give it back
  - Large forwarding tables
    - 2 Million possible class C groups

Classless Inter-Domain Routing (CIDR) – RFC1338

- Arbitrary split between network & host part of address \(\rightarrow\) more efficient use of address space
  - Do not use classes to determine network ID
  - Use common part of address as network identifier
  - E.g., addresses 192.4.16 - 192.4.31 have the first 20 bits in common. Thus, we use these 20 bits as the network number \(\rightarrow\) 192.4.16/20
  - Merge forwarding entries \(\rightarrow\) smaller tables
    - Use single entry for range in forwarding tables
    - Combined forwarding entries when possible
      - "Adjacent" in address space and same egress
CIDR Example

- Network is allocated 8 class C chunks, 200.10.0.0 to 200.10.7.255
  - Move 3 bits of class C address to host address
  - Network address is 21 bits: 201.10.0.0/21
- Replaces 8 class C routing entries with 1 entry
- But how do routers know size of network address?
  - Routing protocols must carry prefix length with address

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</th>
<th>00010111</th>
<th>00010000</th>
<th>00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000</td>
<td>00010111</td>
<td>00010000</td>
<td>00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000</td>
<td>00010111</td>
<td>00010010</td>
<td>00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000</td>
<td>00010111</td>
<td>00010100</td>
<td>00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000</td>
<td>00010111</td>
<td>00011110</td>
<td>00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>

IP Addresses: How to Get One?

- How does an ISP get block of addresses?
  - From Regional Internet Registries (RIRs)
    - ARIN (North America, Southern Africa), APNIC (Asia-Pacific), RIPE (Europe, Northern Africa), LACNIC (South America)
- How about a single host?
  - Hard-coded by system admin in a file
    - 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

CIDR Illustration

Provider is given 201.10.0.0/21

Provider

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

- Longest prefix match!!

Outline

- CIDR addressing
  - Forwarding example
- IP protocol
- IPv6
- NATs

Host Routing Table Example

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.2.209.100</td>
<td>0.0.0.0</td>
<td>255.255.255.255</td>
<td>eth0</td>
</tr>
<tr>
<td>128.2.0.0</td>
<td>0.0.0.0</td>
<td>255.255.0.0</td>
<td>lo</td>
</tr>
<tr>
<td>0.0.0.0</td>
<td>128.2.254.36</td>
<td>0.0.0.0</td>
<td>eth0</td>
</tr>
</tbody>
</table>

- From "netstat -rn"
- Host 128.2.209.100 when plugged into CS ethernet
- Dest 128.2.209.100 → routing to same machine
- Dest 128.2.0.0 → other hosts on same ethernet
- Dest 127.0.0.0 → special loopback address
- Dest 0.0.0.0 → default route to rest of Internet
  - Main CS router: gigrouter.net.cs.cmu.edu (128.2.254.36)

Routing to the Network

- Packet to 10.1.1.3 arrives
- Path is R2 → R1 → H1 → H2
Routing Within the Subnet

- Packet to 10.1.1.3
- Matches 10.1.0.0/23

Routing table at R2

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0</td>
<td>provider</td>
<td>10.1.16.1</td>
</tr>
<tr>
<td>10.1.8.0/24</td>
<td>10.1.8.1</td>
<td>10.1.8.1</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>10.1.2.1</td>
<td>10.1.2.1</td>
</tr>
<tr>
<td>10.1.0.0/23</td>
<td>10.1.2.2</td>
<td>10.1.2.2</td>
</tr>
</tbody>
</table>

• Packet to 10.1.1.3
• Matches 10.1.1.1/31
• Longest prefix match

Routing table at R1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0</td>
<td>provider</td>
<td>10.1.2.1</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>10.1.0.1</td>
<td>10.1.0.1</td>
</tr>
<tr>
<td>10.1.1.0/24</td>
<td>10.1.1.1</td>
<td>10.1.1.1</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>10.1.2.2</td>
<td>10.1.2.2</td>
</tr>
</tbody>
</table>

Outline

- CIDR addressing
- IP protocol
- IPv6
- NATs

IP Service Model

- Low-level communication model provided by Internet
- Datagram
  - Each packet self-contained
    - All information needed to get to destination
    - No advance setup or connection maintenance
  - Analogous to letter or telegram
IPv4 Header Fields

- **Version**: IP Version
  - 4 for IPv4
- **HLen**: Header Length
  - 32-bit words (typically 5)
- **TOS**: Type of Service
  - Priority information
- **Length**: Packet Length
  - Bytes (including header)
- **Header format can change with versions**
  - First byte identifies version
- **Length field limits packets to 65,535 bytes**
  - In practice, break into much smaller packets for network performance considerations

IP Delivery Model

- **Best effort service**
  - Network will do its best to get packet to destination
- **Does NOT guarantee**:
  - Any maximum latency or even ultimate success
  - Sender will be informed if packet doesn’t make it
  - Packets will arrive in same order sent
  - Just one copy of packet will arrive
- **Implications**
  - Scales very well
  - Higher level protocols must make up for shortcomings
  - Reliably delivering ordered sequence of bytes → TCP
  - Some services not feasible
  - Latency or bandwidth guarantees

IP Fragmentation

- **Every network has own Maximum Transmission Unit (MTU)**
  - Largest IP datagram it can carry within its own packet frame
  - E.g., Ethernet is 1500 bytes
  - Don’t know MTUs of all intermediate networks in advance
- **IP Solution**
  - When hit network with small MTU, router fragments packet
  - Destination host reassembles the paper – why?
Fragmentation Related Fields

- Length
  - Length of IP fragment
- Identification
  - 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

IP Fragmentation Example #2

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
  - Loss of a fragment
- Path MTU discovery protocol → determines minimum MTU along route
  - Uses ICMP error messages
- Common theme in system design
  - Assure correctness by implementing complete protocol
  - Optimize common cases to avoid full complexity
Internet Control Message Protocol (ICMP)

- Short messages used to send error & other control information
- Examples
  - Ping request / response
    - Can use to check whether remote host reachable
  - Destination unreachable
    - Indicates how packet got & why couldn’t go further
  - Flow control
    - Slow down packet delivery rate
  - Redirect
    - Suggest alternate routing path for future messages
  - Router solicitation / advertisement
    - Helps newly connected host discover local router
  - Timeout
    - Packet exceeded maximum hop limit

IP MTU Discovery with ICMP

- Typically send series of packets from one host to another
- Typically, all will follow same route
  - Routes remain stable for minutes at a time
- Makes sense to determine path MTU before sending real packets
- Operation
  - Send max-sized packet with “do not fragment” flag set
  - If encounters problem, ICMP message will be returned
    - “Destination unreachable: Fragmentation needed”
    - Usually indicates MTU problem encountered
IP MTU Discovery with ICMP

- When successful, no reply at IP level
  - "No news is good news"
  - Higher level protocol might have some form of acknowledgement

Important Concepts

- Base-level protocol (IP) provides minimal service level
  - Allows highly decentralized implementation
  - Each step involves determining next hop
  - Most of the work at the endpoints
- ICMP provides low-level error reporting

- IP forwarding → global addressing, alternatives, lookup tables
- IP addressing → hierarchical, CIDR
- IP service → best effort, simplicity of routers
- IP packets → header fields, fragmentation, ICMP

Outline

- CIDR addressing
- IP protocol
- IPv6
- NATs

IPv6

- "Next generation" IP.
  - Most urgent issue: increasing address space.
    - 128 bit addresses
  - Simplified header for faster processing:
    - No checksum (why not?)
    - No fragmentation (?)
  - Support for guaranteed services: priority and flow id
  - Options handled as "next header"
    - reduces overhead of handling options
IPv6 Addressing

- Do we need more addresses? Probably, long term
  - Big panic in 90s: “We’re running out of addresses!”
- 128 bit addresses provide space for structure (good!)
  - Hierarchical addressing is much easier
  - Assign an entire 48-bit sized chunk per LAN – use Ethernet addresses
  - Different chunks for geographical addressing, the IPv4 address space,
  - Perhaps help clean up the routing tables - just use one huge chunk per ISP and one huge chunk per customer.

IPv6 Autoconfiguration

- Serverless (“Stateless”). No manual config at all.
  - Only configures addressing items, NOT other host things
  - If you want that, use DHCP.
- Link-local address
  - 1111 1110 10 :: 64 bit interface ID (usually from Ethernet addr)
  - (fe80::/64 prefix)
  - Uniqueness test (“anyone using this address?”)
  - Router contact (solicit, or wait for announcement)
  - Contains globally unique prefix
  - Usually: Concatenate this prefix with local ID → globally unique IPv6 ID
- DHCP took some of the wind out of this, but nice for “zero-conf” (many OSes now do this for both v4 and v6)

Fast Path versus Slow Path

- Common case: Switched in silicon (“fast path”)
  - Almost everything
- Weird cases: Handed to CPU (“slow path”, or “process switched”)
  - Fragmentation
  - TTL expiration (traceroute)
  - IP option handling
- Slow path is evil in today’s environment
  - “Christmas Tree” attack sets weird IP options, bits, and overloads router.
  - Developers can’t (really) use things on the slow path for data flow
    - Slows down their traffic
    - If it became popular, they’d be in the soup!

IPv6 Header Cleanup

- Different options handling
- IPv4 options: Variable length header field. 32 different options.
  - Rarely used
  - No development / many hosts/routers do not support
  - Worse than useless: Packets w/options often even get dropped!
  - Processed in “slow path”.
- IPv6 options: “Next header” pointer
  - Combines “protocol” and “options” handling
  - Next header: “TCP”, “UDP”, etc.
  - Extensions header: Chained together
  - Makes it easy to implement host-based options
  - One value “hop-by-hop” examined by intermediate routers
    - Things like “source route” implemented only at intermediate hops
IPv6 Header Cleanup

- No checksum
- Why checksum just the IP header?
  - Efficiency: If packet corrupted at hop 1, don’t waste b/w transmitting on hops 2..N.
  - Useful when corruption frequent, b/w expensive
  - Today: Corruption rare, b/w cheap

IPv6 Fragmentation Cleanup

- IPv4:
  - Large MTU
- IPv6:
  - Small MTU
  - Router must fragment
  - Discard packets, send ICMP “Packet Too Big”
    - Similar to IPv4 "Don’t Fragment" bit handling
  - Sender must support Path MTU discovery
  - Receive “Packet too Big” messages and send smaller packets
  - Increased minimum packet size
    - Link must support 1280 bytes;
    - 1500 bytes if link supports variable sizes
  - Reduced packet processing and network complexity.
  - Increased MTU a boon to application writers
  - Hosts can still fragment - using fragmentation header. Routers don’t deal with it any more.

Migration from IPv4 to IPv6

- Interoperability with IP v4 is necessary for gradual deployment.

- Alternative mechanisms:
  - Dual stack operation: IP v6 nodes support both address types
  - Translation:
    - Use form of NAT to connect to the outside world
    - NAT must not only translate addresses but also translate between IPv4 and IPv6 protocols
  - Tunneling: tunnel IP v6 packets through IP v4 clouds

Outline

- CIDR addressing
- IP protocol
- IPv6
- NATs
**Altering the Addressing Model**

- **Original IP Model:** Every host has unique IP address
- **Implications**
  - Any host can communicate with any other host
  - Any host can act as a server
    - Just need to know host ID and port number
  - No secrecy or authentication – complicates security
  - Packet traffic observable by routers and by LAN-connected hosts
  - Possible to forge packets
    - Use invalid source address
    - Easy to address hosts

**Private Network Accessing Public Internet**

- Don’t have enough IP addresses for every host in organization
- **Security**
  - Don’t want every machine in organization known to outside world
  - Want to control or monitor traffic in / out of organization

**Reducing IP Addresses**

- Most machines within organization are used by individuals
  - For most applications, act as clients
  - Small number of machines act as servers for entire organization
    - E.g., mail server, web, ...
  - All traffic to outside passes through firewall

(Most) machines within organization don’t need actual IP addresses!

**Network Address Translation (NAT)**

- Within Organization
  - Assign every host an unregistered IP address
    - IP addresses 10/8 & 192.168/16 unassigned
  - Route within organization by IP protocol, can do subnetting, …
  - Firewall
    - Does not let any packets from internal node escape
    - Outside world does not need to know about internal addresses
NAT: Opening Client Connection

- Client 10.2.2.2 wants to connect to server 198.2.4.5:80
- OS assigns ephemeral port (1000)
- Connection request intercepted by firewall
  - Maps client to port of firewall (5000)
  - Creates NAT table entry

NAT: Client Request

- Firewall acts as proxy for client
  - Intercepts message from client and marks itself as sender

NAT: Server Response

- Firewall acts as proxy for client
  - Acts as destination for server messages
  - Relabels destination to local addresses

NAT: Enabling Servers

- Use port mapping to make servers available
  - Manually configure NAT table to include entry for well-known port
  - External users give address 243.4.4.80
  - Requests forwarded to server
NAT Considerations

- NAT has to be consistent during a session.
  - Set up mapping at the beginning of a session and maintain it during the session
    - Recall 2nd level goal 1 of Internet: Continue despite loss of networks or gateways
    - What happens if your NAT reboots?
    - Recycle the mapping that the end of the session
      - May be hard to detect
  - NAT only works for certain applications.
    - Some applications (e.g. ftp) pass IP information in payload
    - Need application level gateways to do a matching translation
    - Breaks a lot of applications.
      - Example: Let’s look at FTP
  - NAT is loved and hated
    - Breaks many apps (FTP)
    - Inhibits deployment of new applications like p2p (but so do firewalls!)
      - Little NAT boxes make home networking simple.
      - Saves addresses. Makes allocation simple.